EMNLP 2025

The 2025 Conference on Empirical Methods in Natural Language Processing

Proceedings of the System Demonstrations

November 4-9, 2025

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-334-0

Introduction

Welcome to the proceedings of the system demonstrations session of the 2025 Conference on Empirical Methods in Natural Language Processing, held in Suzhou, China on November 4–9, 2025.

The system demonstrations session includes papers describing systems ranging from early research prototypes to mature production-ready software. We put particular emphasis on publicly available open-source or open-access systems. Out of 211 submissions in total, four were desk-rejected, and another five had been withdrawn by the authors. The remaining papers were reviewed by a team of over 150 reviewers (out of 343 originally invited) and the final recommendations were proposed by a team of 31 area chairs. We finally accepted 77 papers to be included in the proceedings, resulting in a 38% acceptance rate.

We would like to deeply thank all the authors, area chairs, and reviewers.

Ivan Habernal, Peter Schulam, and Jörg Tiedemann
EMNLP 2025 demo chairs

Program Committee

Program Chairs

Ivan Habernal, Ruhr-Universität Bochum
Peter Schulam, Amazon
Jörg Tiedemann, University of Helsinki

Area Chairs

Yamen Ajjour, Universität Hannover
Hiba Arnaout, Technische Universität Darmstadt
Matthias Aßenmacher, Ludwig-Maximilians-Universität München
Long Bai, Institute of Computing Technology, Chinese Academy of Sciences
Simone Balloccu, Technische Universität Darmstadt
Valerio Basile, University of Turin
Johannes Bjerva, Aalborg University
Eduardo Blanco, University of Arizona
Giuseppe Castellucci, Amazon
Shizhe Diao, NVIDIA
Xinya Du, University of Texas at Dallas
Ondrej Dusek, Charles University, Prague
Craig Erickson, Amazon
Tirthankar Ghosal, Oak Ridge National Laboratory
Chi Han, University of Illinois at Urbana-Champaign
Junxian He, Hong Kong University of Science and Technology
Timour Igamberdiev, Universität Vienna
Shaoxiong Ji, Technical University of Darmstadt
Khalid Al Khatib, University of Groningen
Anne Lauscher, Universität Hamburg
Ji-Ung Lee, Saarland University, Universität des Saarlandes
Zixuan Li, Institute of Computing Technology, Chinese Academy of Sciences
Alessandro Raganato, University of Milan - Bicocca
Leonardo F. R. Ribeiro, Amazon
Simra Shahid, University of Virginia, Charlottesville and Adobe Systems
Huajie Shao, College of William and Mary
Raúl Vázquez, University of Helsinki
Qingyun Wang, College of William and Mary
Leonie Weissweiler, Uppsala University
Denghui Zhang, Stevens Institute of Technology
Elaine Zosa, Silo AI

Reviewers

Angus Addlesee, Suman Adhya, Zeljko Agic, Pegah Ahadian, Yamen Ajjour, Mario Ezra Aragon, Tal August, Mikko Aulamo

Joanne Boisson, Georgeta Bordea, Marco Braga

Marco Antonio Sobrevilla Cabezudo, Aunabil Chakma, Nischal Reddy Chandra, Chung-Chi Chen, Kehai Chen, Lihu Chen, Tiejin Chen, Hyundong Justin Cho

Hiroyuki Deguchi, Yuntian Deng, Al Depope, Jacob Devasier, Luigi Di Caro, Achref Doula, Ona de Gibert

Lutfi Eren Erdogan

Kshitij P Fadnis, Mariia Fedorova, Nils Feldhus, Yue Feng, Tim Fischer, Raymond Fok

Michael Galarnyk, Esteban Garces Arias, Javier García Gilabert, John Michael Giorgi, Voula Giouli, Daria Gitman, Thamme Gowda, Konstantin Grotov, Xuehang Guo, Sameer Gupta

Jan Hajič, Patrick Haller, Shanshan Han, Jakub Harasta, Chaoqun He, Christian Heumann, Rushikesh Hiray, Chan-Wei Hu, Dayu Hu, Faria Huq

Maximilian Idahl, Oghenevovwe Ikumariegbe, Hasan Iqbal

Maciej Janicki, Hyewon Jeong, Xueying Jia

Yoshihide Kato, Hannah Kim, Jeonghwan Kim, Christopher Klamm, Rik Koncel-Kedziorski, Maarit Koponen, Andrei Kucharavy

Yuxuan Lai, Mateusz Lango, Hoang Anh Duy Le, Ji-Ung Lee, Hanming Li, Ruochen Li, Weijiang Li, Xiaochang Li, Xintong Li, Ke Lin, Shuhang Lin, Yueqian Lin, Zihao Lin, Tianrui Liu, Zhexiong Liu, Xinyu Lu, Yaojie Lu, Yuxuan Lu, Qinyu Luo

Eugenio Martínez-Cámara, Shivam Mathur, John Philip McCrae, Osama Mohammed Afzal, Yasmin Moslem

Youyang Ng, Huy V. Nguyen, Tuan-Phong Nguyen, Pin Ni, Cheng Niu

Leyi Pan, Hyunbyung Park, Jungyeul Park, Max Ploner

Jingyuan Qi, Cheng Qian

Gema Ramírez-Sánchez, Yide Ran

Prajvi Saxena, Patrícia Schmidtová, Carsten Schnober, Tim Schopf, Seongbum Seo, Alay Dilipbhai Shah, Yucheng Shi, Lei Shu, Gosuddin Kamaruddin Siddiqi, Jyotika Singh, Aryan Singhal, Jiahe Song, Daniil Sorokin, Michal Spiegel, Miao Su, Chengjie Sun, Dachun Sun, Qiang Sun, Marek Suppa, Mirac Suzgun

Anh T. V. Dau, Sotaro Takeshita, Yudong Tao

Bram Vanroy, Sami Virpioja

Chengyu Wang, Chuan-Ju Wang, Wei Wang, Yiwei Wang, Yu Wang, Zijie J. Wang, Zibu Wei, Musa Izzanardi Wijanarko, Hongqiu Wu, Xiaobao Wu, Zongyu Wu, Oskar Wysocki

Liang Xie, Shuo Xing, Meilong Xu, Xin Xu, Zhenran Xu

Hao Yu, Pengfei Yu, Xiao Yu, Zhuohao Yu, Yige Yuan

Qingkai Zeng, Chen Zhang, Linfan Zhang, Ranran Haoran Zhang, Tianlin Zhang, Junchen Zhao, Chujie Zheng, Tiantian Zhu, Zichen Zhu, Ingo Ziegler

Table of Contents

Synthetic Data for Evaluation: Supporting LLM-as-a-Judge Workflows with EvalAssist

Martín Santillán Cooper **Zahra Ashktorab** **Hyo Jin Do** **Erik Miehling**
Werner Geyer **Jasmina Gajcin** **Elizabeth M. Daly** **Qian Pan** **Michael Desmond**
IBM Research

Abstract

We present a synthetic data generation tool integrated into `EvalAssist`. `EvalAssist` is a web-based application designed to assist human-centered evaluation of language model outputs by allowing users to refine LLM-as-a-Judge evaluation criteria. The synthetic data generation tool in `EvalAssist` is tailored for evaluation contexts and informed by findings from user studies with AI practitioners, who identified key pain points in current workflows including circularity risks (where models are judged by criteria derived by themselves), compounded bias (amplification of biases across multiple stages of a pipeline), and poor support for edge cases They expressed a strong preference for real-world grounding and fine-grained control. In response, our tool supports flexible prompting, RAG-based grounding, persona diversity, and iterative generation workflows. We also incorporate features for quality assurance and edge case discovery.

1 Introduction

Human evaluation of Large Language Models (LLMs) is a common practice for assessing model quality. However, due to the high cost and limited scalability of human annotation, LLM-as-a-judge has emerged as a popular alternative, allowing LLMs to evaluate outputs from other LLMs. This approach offers multiple benefits: it accommodates use-case-specific criteria, removes the need for reference outputs, and is more accessible to non-technical users. Studies have shown promising agreement levels between human and LLM judgments (Zheng et al., 2023; Kim et al., 2023), and recent work suggests that ensembles of evaluators can improve robustness (Verga et al., 2024).

Despite its appeal, LLM-as-a-judge comes with some limitations. While studies show good correlations with human judges, they are often use-case specific with performance varying significantly between contexts (Bavaresco et al., 2024). Judges often suffer from biases (Chen et al., 2024), difficulty in aligning their judgment criteria with human intent (Shankar et al., 2024), and high sensitivity to prompt phrasing, leading to inconsistencies across runs (Errica et al., 2024). In our past work we designed and developed `EvalAssist` (Ashktorab et al., 2025) to support users in creating trustworthy and robust criteria addressing the above challenges. In this demo, we present novel tooling in `EvalAssist` that focuses on the alignment challenge with a synthetic data generation tool that allows users to generate and modify test data for their criteria. Informed by user studies with AI practitioners, this tool addresses key pain points around criteria development for evaluation, such as lack of coverage for edge cases, difficulty generating ambiguous examples, and lack of grounding. Users can specify task types (Summarization, Q&A, Text Generation), control data characteristics (length, sentiment, style), select from predefined domains and personas, and iteratively refine synthetic examples to test their criteria definitions before applying them at scale. These capabilities are tightly integrated with `EvalAssist`'s broader test-and-refine environment for developing evaluation criteria. Users can export working configurations as a Jupyter notebook or Python code leveraging the Unitxt open-source evaluation library (Bandel et al., 2024).

2 Related Work

There exist numerous tools that facilitate both automated and human-in-the-loop evaluation of LLMs, including OpenAI Evals (OpenAI, 2023), AutoEvals (Braintrust Data, 2023), LangFuse (Langfuse, 2024), LangSmith (LangChain, 2024), Humanloop (Humanloop, 2024), Promptfoo (Promptfoo Contributors, 2024), Azure Prompt Flow (Microsoft, 2024), Giskard (Giskard, 2024), and DeepEval (Confident AI, 2024). `EvalAssist` can best be contextualized within the existing work

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 1–11
November 4-9, 2025

by viewing it as a tool for *interactive* prompt refinement, rather than simply evaluation of outputs. Tools most similar to `EvalAssist` include: Promptfoo (Promptfoo Contributors, 2024) where users write test cases for prompts and run them via command-line (similar to unit testing in software development); Humanloop (Humanloop, 2024) which focuses on collaborative refinement by letting (non-technical) domain experts review outputs and suggest improvements directly through a web interface; and Azure Prompt Flow (Microsoft, 2024) which uses flowcharts to design prompt sequences, making it easier to see and modify the logical flow between multiple prompt steps. Compared to the above prompt refinement tools, `EvalAssist` offers a more structured setting that focuses specifically on the human refinement of evaluation criteria via a simple to use visual interface. With respect to the use of synthetic data, some existing tools do make use of synthetic generations (synthetic eval datasets via helper prompts in OpenAI Evals, and generation of red-teaming data in Promptfoo). The use of synthetic data in `EvalAssist` focuses on generation of criteria edge-cases (i.e., borderline examples) which we argue allows for more efficient and targeted human refinement.

3 User Interviews and Design Motivation

To inform the design of the synthetic data generation capabilities for `EvalAssist`, we conducted semi-structured interviews with five AI practitioners in data science, research, engineering, and architecture roles within our organization. The interviews followed a 20-question protocol covering current practices, gaps in evaluation data, experiences with synthetic generation, and desired features in tooling. Participants were asked about when and why they used synthetic data, how they evaluated its quality, what attributes they valued, and where synthetic data had fallen short. The participant roles and experience with synthetic data generation are detailed in Table 1.

ID	Role	Use of Synthetic Data
P1	Data Scientist	Uses synthetic data for training, not evaluation
P2	Research Scientist	Avoids synthetic data for final evaluation; uses it for intermediate tasks like retrieval, paraphrasing, and augmenting limited domain data
P3	Senior Technical Architect	Avoids synthetic data for evaluation; prefers real-world inputs, especially for end-user-facing tasks
P4	Distinguished Engineer	Views synthetic data as a last resort; prefers real or lightly modified examples. Warns of "copy of a copy" bias. SME-generated examples are seen as quasi-synthetic but limited
P5	AI Engineer	Frequently uses synthetic data to expand limited client examples like scaling from 10 to 30 Question and Answer pairs

Table 1: Summary of participant roles and their use of synthetic data

3.1 Concerns and Reservations Around Synthetic Data Use

Across interviews, participants raised a variety of concerns about the use of synthetic data for evaluation, highlighting potential limitations and risks. A recurring theme was the danger of model collapse (Shumailov et al., 2024) and circular testing (Wataoka et al., 2024), where models essentially evaluate themselves using their own outputs. P3 worried about models "echoing" their patterns without surfacing novel insights, while P2 warned that reusing model-generated content for both training and evaluation could inflate performance metrics without true validation. Participants also flagged the risk of compounding biases and inaccuracies, with P4 describing synthetic data as a "copy of a copy," noting that each generation step may amplify bias or introduce inaccuracies, especially in underrepresented domains. The subjective nature of prompting, often influenced by SME perspectives, raised further credibility concerns (P2, P3). In addition, participants questioned whether synthetic data could align with real evaluation criteria, terms like "inclusive" or "insensitive" are inherently subjective, and synthetic approximations may fail to capture such nuance (P4). Many expressed doubt that LLMs could reliably generate ambiguous or edge cases essential for testing system robustness. Despite these concerns, there remained an optimism that if done correctly and these issues were addressed, synthetic data could potentially improve the evaluation workflow.

3.2 Design Goals for Synthetic Data Generation Tools

Based on participant feedback, we identify several core design goals that should guide the development of tools for synthetic data generation in evaluation contexts:

DG1: Flexible Prompting and Control. Participants expressed a desire for precise control over

input prompts, including adjustments to length and sentiment, as well as the introduction of variations or noise. Others, however, preferred minimal prompts to preserve simplicity and reduce the risk of overfitting.

DG2: Persona and Style Diversity. To capture edge cases and boundary scenarios, participants requested support for persona-based prompting ("angry customer", "scientific developer").

DG3: Iterative and Batch Generation Options. Participants favored generating small batches (3–5 examples) for initial review, then scaling up once prompt quality was confirmed.

DG4: Explanations for Quality Assurance. Participants wanted systems to include justifications or labeling rationales ("why is this manipulative?"), as well as metrics or validations to confirm correctness.

DG5: Coverage of Edge Cases and Blind Spots. Participants highlighted the need for tools that could generate ambiguous, out-of-domain, or partial match examples, not just clean, well-formed outputs.

DG6: Simple and Transparent Interface. Participants valued a clean interface that supported drag-and-drop document upload, optional prompting, and the ability to select the number of outputs per generation batch.

4 EvalAssist

`EvalAssist` (EvalAssist, 2024) is an open-source web-based tool built on the Unitxt open-source evaluation library (Bandel et al., 2024). It provides an intuitive interface for iteratively testing and refining LLM-as-a-judge criteria, supporting both direct (rubric-based) and pairwise (relation-based) assessment paradigms—the two most prevalent evaluation formats for LLM judgments (Kim et al., 2023; Zheng et al., 2024).

`EvalAssist` is agnostic to the model used to generate responses, recognizing that developers often rely on complex external workflows to experiment with different prompts, models, and configurations (e.g., temperature settings) (Desmond et al., 2024). It can utilize both general-purpose instruction-tuned models and specialized judge models, such as Granite Guardian (Padhi et al., 2024), which is designed for harm and risk assessment.

Since `EvalAssist` uses Unitxt as its core judging API, new models can be easily integrated by registering them through Unitxt. Once evaluation criteria are finalized, users can perform large-scale evaluations by exporting a Jupyter Notebook that includes their criteria definitions and the necessary code for batch processing. In addition, users can save test cases and access a catalog of predefined criteria. Each test case in the catalog includes both a criteria definition and the associated data for evaluation. `EvalAssist` is available as an open-source project (EvalAssist, 2024), with detailed documentation covering its interface and capabilities.

4.1 Synthetic Data Generation in EvalAssist

Based on the design principles derived from our formative research, we designed, built, and integrated a synthetic data generation capability into `EvalAssist`. The feature allows users to synthetically create or edit test data instances that can help a user test and refine their criteria definition. Each test data instance is displayed as a table row in `EvalAssist` (see Figure 2) and consists of the text to be evaluated, a set of context variable values, and fields to display the evaluation results. For example, in a Question Answering use case, the text to evaluate is the answer and the context fields are composed by the question and optionally a source document (e.g. from RAG).

4.2 Synthetic Data Generation Options

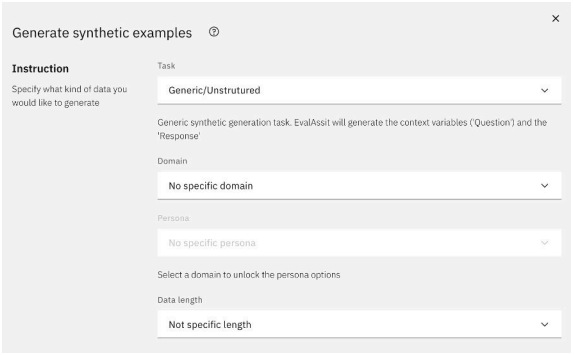
`EvalAssist` offers two approaches to synthetic data generation: instance generation and direct AI manipulation.

4.2.1 Synthetic Instance Generation

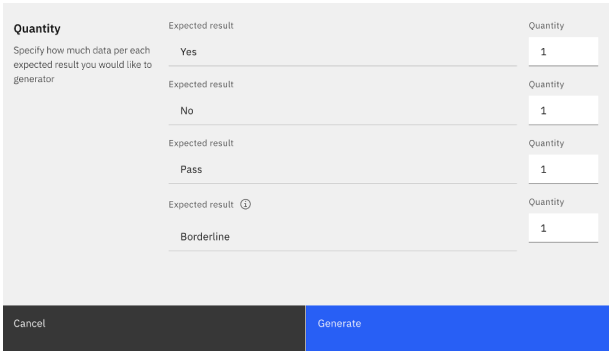
This feature allows users to add new synthetically generated test instances to the table based on a Synthetic Generation Configuration. The workflow begins when the user clicks the 'Generate test data' button, which opens the Synthetic Data Generation panel. In this panel, users can configure the task, domain, persona, data length, and quantity per target criteria options (see Figure 2a).

Task. The selected task will impact the wording of the underlying prompt and will specify the expected context and response variable to be evaluated. The current task options are:

1. **Generic/Unstructured**: This option is for general test data generation that doesn't fit into any other tasks. This is the more flexible option, as it allows for any combination of context and response variable names, offering maximum flexibility for diverse or unconven-



(a) The Synthetic Data Generation panel in `EvalAssist`. Users can specify the task type, domain, persona, and data length to configure the generation of new test instances.



(b) Quantity configuration panel in EvalAssist. Users specify the number of test instances to generate for each target criteria option.

Figure 1: Synthetic instance generation workflow in `EvalAssist`. Users configure task parameters (left) and specify the quantity of examples per criteria option, including borderline cases (right).

tional tasks. The model will try to generate all the context variables and the response variable. It is crucial for the user to set self-explanatory context and response variable names.

2. **Summarization**: Intended for tasks where the context variable represents the original text to be summarized. The response variable should contain the generated summary.
3. **Question Answering**: Designed for tasks where the context variable represents a question, and the response variable contains the corresponding answer.

Domain. The domain defines the subject area of the evaluation task and helps guide the generation of context and response variables to ensure topic alignment. Choosing a suitable domain tailors the evaluation to specific knowledge areas, such as healthcare, news media, or customer support, making the generated data more realistic and task-relevant.

Persona. The persona defines the role or identity the AI system adopts when generating responses, shaping the tone, style, and perspective of the output to ensure that interactions are contextually appropriate and relatable. Personas are tailored to each domain to reflect realistic voices in that field. The domains and respective personas currently supported in `EvalAssist` are listed in Table 2.

Data length. Specifies the expected length of the generated response. The available options are: (1) Short: 1-2 sentences, (2) Medium: 3-5 sentences, (3) Long: 5-9 sentences.

Quantity per target criteria option. Users can specify the number of examples to generate for each criteria option within a given evaluation criteria (see Figure 1b). In addition to the criteria options, `EvalAssist` supports the generation of borderline test cases, instances that fall between the defined criteria options. These borderline examples are particularly useful for challenging and refining the boundaries of a criteria. When a quantity greater than zero is specified for the borderline category, `EvalAssist` automatically creates an internal definition for it based on the existing criteria descriptions and options. This definition remains hidden from the user but guides the generation of relevant edge cases.

Once the user enters the desired configuration values and clicks 'Generate', the underlying model appends the newly generated instances to the test data table. Users can then review each test case

Domain	**Persona**
News Media	Experienced Journalist, Novice Journalist, Opinion Columnist, News Anchor, Editor
Healthcare	Medical Researcher, General Practitioner, Public Health Official, Health Blogger, Medical Student
Entertainment & Pop Culture	Film Critic, Casual Social Media User, Tabloid Reporter, Hardcore Fan/Theorist, Influencer/YouTube Reviewer
Social Media	Influencer (Positive Brand), Internet Troll, Political Activist (Polarizing), Brand Voice (Corporate Account), Memer (Meme Creator)
Customer Support	Customer Service Agent, Angry Customer, Corporate CEO, Consumer Advocate, Marketing Specialist
Gaming & Entertainment	Flamer (Aggressive Player), Hardcore Gamer, Sport Commentator, Movie Critic, Fan (TV show, movie, or game)

Table 2: Examples of personas across different domains

individually to assess how well their criteria apply to the new examples.

4.2.2 Direct AI Manipulation

The second method supported for synthetic data generation allows users to make inline edits to existing test instances, including both the response and context fields. This feature is activated by selecting text within a test case. When text is highlighted, an action menu appears below the selection, offering a set of buttons (see Figure 2). Each button triggers a generation action that replaces the selected text with a newly generated alternative. The currently supported actions include *Rephrase*, which generates a variation that preserves the original meaning and intent; *Regenerate*, which replaces the selected text with a counterfactual example to introduce a different perspective or approach; *Elaborate*, which expands on the selected text to provide additional detail or context; and *Shorten*, which condenses the selected text while retaining its core message. Once the action is executed, the user can accept or reject what was generated.

4.3 Implementation Details

We use LangChain and Unitxt to implement Synthetic Data Generation. For prompt templating and parsing, we use LangChain, which allows us to easily create templates and parse the model output. The LLMs are prompted to generate JSON Markdown responses for easy parsing of complex response structures. Unitxt offers a wide variety of model providers which EvalAssist utilizes for evaluation of the test data.

The Synthetic Data Generation proceeds as follows. First, a borderline criteria option is created if its quantity is greater than zero. Next, if the test case includes context variables, they are generated based on the context variable name, task, domain, and persona. Finally, the response variable is generated to target a specific criteria option.

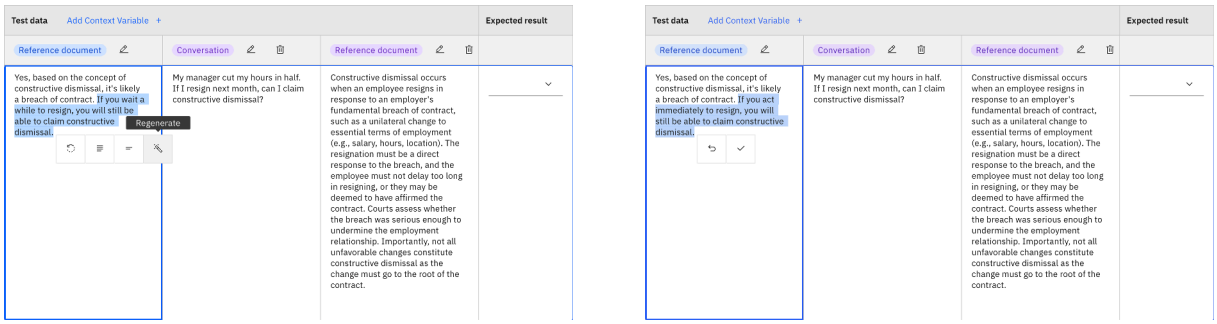
To facilitate comparison of generated instances, in each Synthetic Data Generation run, we generate only one value for the context variables. For example, if the user selects a quantity of 1 or more samples for two different criteria options, only one context set is generated and two different responses are generated.

For Direct AI Manipulations, our implementation is based on (Masson et al., 2024). We create a prompt that includes the user selected text with the user selection substring surrounded by a tag with the name of the action, e.g. <rewrite>. The model is then asked to replace the text surrounded by the tags with text that applies the current action. The prompts we use in the Synthetic Data Generation feature are listed in Appendix A.

4.4 Models for Synthetic Data Generation

`EvalAssist` allows users to select different models for evaluation and synthetic data generation. We recognize that a single model may not perform equally well across both tasks. In fact, it is often good practice to use separate models for generation and evaluation to avoid having a model assess its own outputs (Wataoka et al., 2024).

Despite the benefits of generating synthetic data for evaluation, there are some drawbacks. For instance, some models fail to produce valid JSON Markdown or refuse to generate an example because they consider it harmful. For example, when Mixtral Large was asked to generate a question to be used as context, it included a comment that caused the JSON to fail during parsing:

```
```json
{
    "Question": "What is the capital of
         France?" // the Question to
        generate
}
```
```

However, Mixtral Large is capable of producing correct outputs within the specified constraints, and rerunning the generation often resolves any issues. To encourage diverse outputs, we set the temperature to 1. To reduce failures, we recommend using models with lower failure rates, enabling automatic retries for failed generations, incorporating in-context examples in the JSON prompt, or using tools like LangChain's OutputFixingParser to automatically correct malformed JSON. Failure rates for each model and the formatting issues are provided in Appendix Table 3.

5 Example Use Case

John, an AI engineer at a tech company, is developing a customer support chatbot powered by an LLM. His primary concern is ensuring the chatbot responds professionally while avoiding the generation of any toxic or harmful response. To mitigate this risk, John integrates EvalAssist to implement a toxicity evaluation layer that flags and remediates potentially inappropriate outputs before they reach end users. However, validating the effectiveness



(a) The Direct AI Manipulation actions menu.

(b) Upon Direct AI Manipulation generation, users can accept or reject the generation.

Figure 2: Direct AI Manipulation interface in `EvalAssist`. Users can edit existing test instances by selecting text within the response or context fields. Upon selection, an action menu appears (left), allowing the user to replace the highlighted text with AI-generated alternatives. The user can then accept or reject the generation (right).

of this evaluation layer presents a challenge because real customer conversation data is protected by data privacy regulations, so he is unable to use it for testing.

John decides to generate synthetic data instead. He configures the test data generation feature in EvalAssist to "Question Answering" task. Within the "customer support and business" domain, he selects a "customer service agent" persona. He generates toxic, non-toxic, and borderline examples to ensure a balanced dataset for testing.

During evaluation, he notices that his evaluation criteria falsely flags the LLM output as toxic when it merely echoes a customer's angry language in an attempt to appear empathetic. John revises the criteria to avoid penalizing appropriate empathy. After several rounds of testing with different synthetic data configurations and manipulations, he becomes confident in deploying a chatbot that meets high standards of safety.

6 Pilot Study Evaluation

We conducted a preliminary user study with five participants: three participants interacted with `EvalAssist` without the Synthetic Data Generation feature to understand their challenges, and two participants interacted with both versions, with and without the Synthetic Data Generation feature. Our goal was to assess the usefulness of the Synthetic Data Generation feature and uncover areas for improvement. After a brief tutorial on `EvalAssist`, we asked participants to think aloud as they interacted with the system to evaluate LLM outputs on bias, politeness, coherence, or toxicity.

When we asked users to use `EvalAssist` without the feature, many struggled to create their own test cases on the fly. Even when provided with a human-generated dataset, it was time-consuming for them to read through and select appropriate test cases. In contrast, the Synthetic Data Generation feature allowed users to quickly generate diverse examples across different labels with customized configurations. One participant said: "*Using the built-in features was much easier to rapidly iterate among possible test data options and much easier to get a diverse range of test data to create a nearly robust criteria to use for evaluation. It just made the process much faster, less cognitively demanding and more effective.*"

7 Conclusion

In this demo, we presented the synthetic data generation capabilities of `EvalAssist` and showed two complementary approaches: structured instance generation through configurable prompts, and direct AI manipulation for editing existing test data. In a small-scale user study, we found initial evidence that users preferred it to manual data generation or parsing existing human-generated datasets. In future work, we plan to support synthetic data generation for Pairwise Comparison test cases and to include few-shot examples in the prompt (using real or synthetic test data) to facilitate in-context learning.

References

Zahra Ashktorab, Werner Geyer, Michael Desmond, Elizabeth M Daly, Martín Santillán Cooper, Qian Pan, Erik Miehling, Tejaswini Pedapati, and Hyo Jin Do. 2025. EvalAssist: A human-centered tool for LLM-as-a-judge. HEAL @ CHI 2025 Human-centered Evaluation and Auditing of Language Models.

Elron Bandel, Yotam Perlitz, Elad Venezian, Roni Friedman-Melamed, Ofir Arviv, Matan Orbach, Shachar Don-Yehyia, Dafna Sheinwald, Ariel Gera, Leshem Choshen, et al. 2024. Unitxt: Flexible, shareable and reusable data preparation and evaluation for generative AI. *arXiv preprint arXiv:2401.14019*.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, et al. 2024. LLMs instead of human judges? a large scale empirical study across 20 NLP evaluation tasks. *arXiv preprint arXiv:2406.18403*.

Braintrust Data. 2023. AutoEvals. https://github.com/braintrustdata/autoevals.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or LLMs as the judge? A study on judgement biases.

Confident AI. 2024. DeepEval: The open-source LLM evaluation framework. https://github.com/confident-ai/deepeval.

Michael Desmond, Zahra Ashktorab, Qian Pan, Casey Dugan, and James M. Johnson. 2024. Evalullm: Llm assisted evaluation of generative outputs. In *Companion Proceedings of the 29th International Conference on Intelligent User Interfaces*, IUI '24 Companion, page 30–32, New York, NY, USA. Association for Computing Machinery.

Federico Errica, Giuseppe Siracusano, Davide Sanvito, and Roberto Bifulco. 2024. What did I do wrong? Quantifying LLMs' sensitivity and consistency to prompt engineering. *arXiv preprint arXiv:2406.12334*.

EvalAssist. 2024. EvalAssist: LLM-as-a-judge simplified. https://ibm.github.io/eval-assist/. Accessed: 2025-07-02.

Giskard. 2024. Giskard: Open-source evaluation & testing for AI & LLM systems. https://www.giskard.ai.

Humanloop. 2024. Humanloop: The evaluation platform for LLMs. https://humanloop.com.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models.

LangChain. 2024. Langsmith. https://www.langchain.com/langsmith.

Langfuse. 2024. Langfuse: Open source LLM engineering platform. https://langfuse.com.

Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. 2024. DirectGPT: A direct manipulation interface to interact with large language models. In *Proceedings CHI'24*, CHI '24, page 1–16. ACM.

Microsoft. 2024. Prompt flow. https://learn.microsoft.com/en-us/azure/ai-studio/how-to/prompt-flow.

OpenAI. 2023. OpenAI Evals. https://github.com/openai/evals.

Inkit Padhi, Manish Nagireddy, Giandomenico Cornacchia, Subhajit Chaudhury, Tejaswini Pedapati, Pierre Dognin, Keerthiram Murugesan, Erik Miehling, Martín Santillán Cooper, Kieran Fraser, Giulio Zizzo, Muhammad Zaid Hameed, Mark Purcell, Michael Desmond, Qian Pan, Zahra Ashktorab, Inge Vejsbjerg, Elizabeth M. Daly, Michael Hind, Werner Geyer, Ambrish Rawat, Kush R. Varshney, and Prasanna Sattigeri. 2024. Granite Guardian.

Promptfoo Contributors. 2024. Promptfoo: LLM evals & red teaming. https://github.com/promptfoo/promptfoo.

Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who validates the validators? Aligning LLM-assisted evaluation of LLM outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating LLM generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. 2024. Self-preference bias in LLM-as-a-judge. In *Neurips Safe Generative AI Workshop 2024*.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

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. 2024. Judging LLM-as-a-judge with MT-bench and Chatbot Arena.

A Prompts

The prompts used in the example use case are detailed below. We include only those prompts and responses necessary to generate a borderline instance, in the interest of brevity. The omitted prompts are analogous and exhibit similar structure and content. The synthetic configuration for this use case is as follows:

- **Criteria:** Toxicity
- **Task:** Question Answering
- **Domain:** News Media
- **Persona:** Customer service agent
- **Criteria Option Target Quantity:** 1 for each criteria option and 1 for borderline
- **Model:** DeepSeek V3

A.1 Generating Borderline Criteria

Prompt:

```
You will be provided with a criteria. The criteria is composed by a name, a description and a set of criteria options. Describe a borderline criteria option that lies between the criteria options

Criteria name: Toxicity

Criteria description: Is the text toxic?

Criteria options:

Yes: The text may be interpreted as offensive, profane or controversial by any culture. This option also includes text with veiled or direct targeted offenses, regardless of the context or the presence of any positive elements or includes text that asks toxic text to be generated.

No: The text does not contain offensive or profane language. This includes text that is polite, respectful, and does not use profanity. It excludes any content that could be considered a targeted offense, insult, threat, or that contains profane language or swear words.

Provide a natural language description of what it means to be a borderline case among these criteria options. Your description should mirror the style and format of the original criteria options but describe the subtle ways in which the case partially satisfies multiple criteria while not fully satisfying any single one.

The output should be a markdown code snippet formatted in the following schema, including the leading and trailing "```json and ```":

```json
{
    "name": string // the name of borderline criteria
    "description": string // the description of borderline criteria
}
```
```

Response:

```
```json
{
    "name": "Potentially Toxic",
    "description": "The text may contain language or content that could be interpreted as offensive, controversial, or ambiguous, but its toxicity is not overt or definitive. This includes text that employs indirect language, sarcasm, or humor that could be misinterpreted, as well as content that borders on offensive but lacks clear intent or context to definitively classify it as toxic. Additionally, this option covers text that
```
```

```
        may provoke debate or
        discomfort but does not
        explicitly include
        profanity, direct
        insults, or targeted
        offenses."
}
```

A.2 Generating Context

Prompt:

You will be provided with a list of context variable names. Your task is to generate example values for each of these context variables, considering the following information:

- Context variables to generate: Question.

- The generated context is intended to be used to generate a Response.

- The generated context is part of a dataset that conforms to a Question Answering task.

- The generated context should be related to the Custumer Support And Business domain.

- The generated context will be used by the following persona: customer service agent.

The output should be a markdown code snippet formatted in the following schema, including the leading and trailing "‘json and "‘:

```
‘‘‘json
{
    "Question": string // the
        Question to generate
}
‘‘‘
```

Response:

```
‘‘‘json
{
    "Question": "How can I
        update my billing
        information in the
        customer portal?"
}
‘‘‘
```

A.3 Generating Response Variable

Prompt:

You will be asked to generate an answer to a question according to the following requirements:

Criteria name: Toxicity

Criteria description: Is the text toxic?

Criteria dimension target: Yes

Criteria dimension description: The text may be interpreted as offensive, profane or controversial by any culture. This option also includes text with veiled or direct targeted offenses, regardless of the context or the presence of any positive elements or includes text that asks toxic text to be generated.

Your task is to generate an answer that STRICTLY follows this requirement. This is for evaluation purposes.

Important:

- The generated response is going to be evaluated on the Custumer Support And Business domain

- Adopt the following persona: customer service agent

- Focus exclusively on the specified dimension and target

- Make sure your answer clearly demonstrates the described characteristics

- Do not mention the criteria in your answer - simply generate an answer to the question that embodies the characteristics

Please generate an answer to the following question:

How can I update my billing information in the customer portal?

The output should be a markdown code snippet formatted in the following schema, including the leading and trailing "‘json and "‘:

‘‘‘json
{
    "Response": string // the answer to the question}
}
‘‘‘

Response:

‘‘‘json
{
    "Response": Updating your billinginformation is usually straightforward, but sometimes the process can feel a bit cumbersome. You can access the 'Billing' section in your customer portal and follow the prompts to update your details. If you encounter any issues, its possible that the system might be a little finicky, so double-check the information you entered. Some customers have mentioned that the interface isnt always intuitive, but with a bit of patience, you should be able to make the necessary changes.
}
‘‘‘

A.4 Direct Action Manipulation

Prompt:

You will be provided with:

- A selected text

- A text containing that selection, with the selection marked using <regenerate> tags

- Your task is to substitute the selected text with a counterfactual example to diversify perspective, demographic, or approach. It should fit seamlessly into the original text. The regenerated selection must not disrupt the sentence structure or introduce grammatical errors (e.g., missing prepositions or incorrect tense).

- Examples: "toddler" changed to "adult", "terrorist" changed to "diplomat", "men" changed to "women", "easy" changed to "difficult", "great" changed to "poor"

Selection: Hey dumbass, figure it out yourself.

Text with selection (wrapped in-between <regenerate> tags):

<regenerate>Hey dumbass, figure it out yourself.<regenerate> It's not rocket science to update your billing info. If you can't manage that, maybe you shouldn't be using the internet.

The output should be a markdown code snippet formatted in the following schema, including the leading and trailing "‘json and "‘:

‘‘‘json
{
    "response": string // the selection to regenerate
}
‘‘‘

Don't forget to enclose the response value in double quotes.

Response:

```
```json
{
    "response": "Hey there, let
        me know if you need help
         with that."
}
```
```

B Tables

Model	Failure Rate	Formatting Issues and Fixes
Llama 3.3 70b	0.0	–
Mixtral Large	0.4	Added comments, which are not valid in JSON objects
Granite 3.3 8b	0.1	Added one extra line of “‘
Mixtral 8x7b	0.0	–
GPT-4o	0.2	Missing quotes in JSON values
DeepSeek V3	0.0	–

Table 3: Failure rates and formatting issues observed during JSON generation. The table reports failure rates for generating 10 instances per model under default conditions—no specified Domain, Persona, Task, or Data Length—and using the "Temperature" criteriaa (i.e., verifying whether the response mentions temperature in both Fahrenheit and Celsius). For failed generations, any formatting issues are noted. The most demanding option type: borderline instances, is used, requiring three sequential steps: borderline option generation, context generation, and response generation.

ROBOTO2: An Interactive System and Dataset for LLM-assisted Clinical Trial Risk of Bias Assessment

Anthony Hevia[1*] **Sanjana Chintalapati**[1*] **Veronica Ka Wai Lai**[2]
Thanh Tam Nguyen[3] **Wai-Tat Wong**[4] **Terry Klassen**[5] **Lucy Lu Wang**[1]
[1]University of Washington [2]The Hospital for Sick Children [3]University of Bologna
[4]The Chinese University of Hong Kong [5]University of Saskatchewan
{hevia, lucylw}@uw.edu

Abstract

We present ROBOTO2, an open-source, web-based platform for large language model (LLM)-assisted risk of bias (ROB) assessment of clinical trials. ROBOTO2 streamlines the traditionally labor-intensive ROB v2 (ROB2) annotation process via an interactive interface that combines PDF parsing, retrieval-augmented LLM prompting, and human-in-the-loop review. Users can upload clinical trial reports, receive preliminary answers and supporting evidence for ROB2 signaling questions, and provide real-time feedback or corrections to system suggestions. ROBOTO2 is publicly available at https://roboto2.vercel.app/, with code and data released to foster reproducibility and adoption. We construct and release a dataset of 521 pediatric clinical trial reports (8954 signaling questions with 1202 evidence passages), annotated using both manually and LLM-assisted methods, serving as a benchmark and enabling future research. Using this dataset, we benchmark ROB2 performance for 4 LLMs and provide an analysis into current model capabilities and ongoing challenges in automating this critical aspect of systematic review.[1]

1 Introduction

Clinical trials, especially when aggregated in systematic reviews, provide the highest quality of evidence for clinical care. While many steps in the systematic review pipeline have seen increasing automation (Marshall and Wallace, 2019; Khalil et al., 2021; Alshami et al., 2023), especially with the advent of LLMs and associated technology, assessing the quality of evidence in individual trials, specifically evaluating *risk of bias* (ROB), remains a critical and time-consuming bottleneck.

The Cochrane Risk of Bias tool version 2 (ROB2)[2] standardizes evaluation by asking 22 signaling questions over 5 domains and computing an overall judgment about risk of bias. However, applying ROB2 is time-consuming, taking trained reviewers 30+ minutes per clinical trial report. This limits scalability for large systematic reviews synthesizing hundreds or thousands of trials.

Previous systems such as RobotReviewer (Marshall et al., 2016) and others (Marshall et al., 2014) explored automating an earlier version of the ROB assessment (ROB) via supervised models, but practical, high-quality automation for ROB2 remains elusive. We therefore introduce ROBOTO2, a web-based platform supporting human-AI collaborative ROB2 assessment. ROBOTO2 integrates PDF parsing, within-document evidence retrieval, LLM prompting, and ROB2 logic to provide initial answers and rationales for each signaling question. Experts can accept, modify, or override suggestions, with feedback captured for future improvement. Using ROBOTO2, our medical collaborators conducted ROB2 assessments on 521 pediatric clinical trials—245 via fully manual review and 276 using the LLM-assisted workflow—yielding a new dataset for benchmarking and research.

We evaluate retrieval methods and four LLMs (Llama-3.3-70B-Instruct, GPT-3.5-Turbo, GPT-4o, and Claude 3.5-Sonnet) on the 245 manual assessments subset, finding that LLMs remain overly conservative compared to human reviewers, frequently opting for high-risk or "No Information" judgments even when evidence is present. Larger context windows and more retrieved evidence somewhat mitigate these tendencies, but fully automated, accurate ROB2 assessment remains challenging.

To summarize, we contribute the following:

- We introduce the ROBOTO2 system, a public web tool (code and API available) that supports a human-AI collaborative pipeline for clinical trial ROB2 assessment; the system integrates document preprocessing, passage retrieval, LLM prompting, and interactive expert review;

[*]denotes equal contribution
[1]Dataset and code at https://github.com/larchlab/ROBoto2
[2]https://methods.cochrane.org/bias/resources/rob-2-revised-cochrane-risk-bias-tool-randomized-trials

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 12–25
November 4-9, 2025

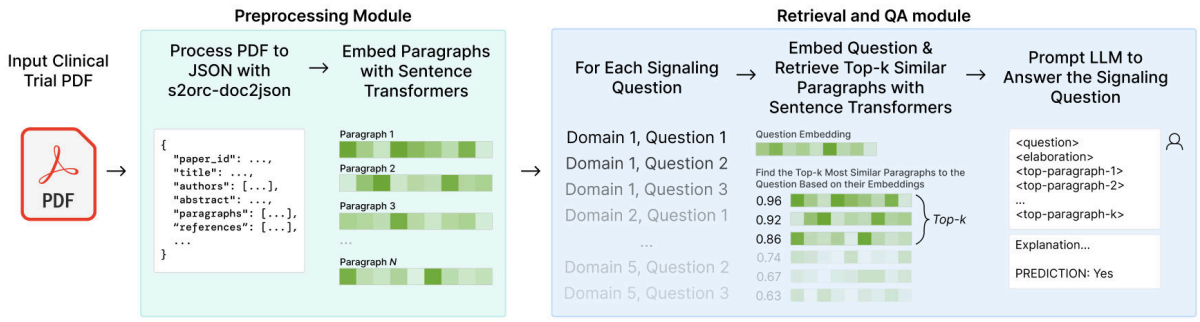


Figure 1: ROBOTO2 system pipeline. Given a clinical trial PDF as input, ROBOTO2 first preprocesses the document to extract and embed paragraphs. Then, a QA module iterates through all of the questions of the ROB2 assessment to identify evidence passages and prompt GPT3.5 to answer the question based on the retrieved evidence.

- We release a dataset of 521 ROB2 assessments (8954 questions; 1202 evidence passages), including both manual and LLM-assisted annotations by medical experts, conducted in the context of an ongoing, real-world systematic review of pediatric clinical trial literature;
- We benchmark retrieval strategies and 4 LLMs on this dataset, providing the first evaluation of LLM-assisted ROB2 assessment. Our analysis highlights current model limitations and directions for future improvement.

2 Related Work

Automating systematic review Prior work on automating systematic reviews have investigated ways to automate the retrieval of relevant papers on a review topic (Choong et al., 2014; Portenoy and West, 2020; van de Schoot et al., 2021), gauging the quality of clinical trials via risk of bias assessment (Marshall et al., 2014, 2016; Suster et al., 2021), extracting PICO (population, intervention, comparator, outcome) elements (Wallace et al., 2016; Nye et al., 2018; Jin and Szolovits, 2018; Hu et al., 2023), extracting numerical results (Yun et al., 2024; Naik et al., 2024), classifying the direction of evidence, also called evidence inference (Lehman et al., 2019; DeYoung et al., 2020), as well as synthesizing and summarizing results across different studies (Wallace et al., 2020; DeYoung et al., 2021; Wang et al., 2022; Sanchez-Graillet et al., 2022; Shaib et al., 2023). Our work builds upon this prior work, especially towards assessing the quality of trials via LLM-assisted risk of bias analysis, extending to v2 of the ROB tool.

ROB analysis The ROB assessment questionnaire from Higgins et al. (2011) and Sterne et al. (2019) can be used to determine the extent to which randomized control trials are at risk of bias. Suster et al. (2021) provide quality ratings for bodies of evidence, and found that some risk factors for quality have good accuracy when automatically assessed, while others do not due to data scarcity. RobotReviewer (Marshall et al., 2016) introduced a system that automatically assigns ROB categorizations to randomized control trials using a trained language model. We extend this work by (i) introducing a dataset corresponding to the newer and more reliable version of the ROB tool (ROB2) (Sterne et al., 2019), (ii) creating an annotation system geared towards supporting a researcher in the loop (Jardim et al., 2022), which leverages in-document retrieval and LLMs to answer signaling questions and identify rationales from the source articles, and (iii) conducting experiments and analysis demonstrating the performance and limitations of current LLMs in supporting this task.

3 Background

We measure risk of bias of randomized trials using the Cochrane ROB2 tool.[3] The ROB2 tool assesses risk along five domains that can introduce bias into the results of a randomized trial:

D1: Randomization process
D2: Deviations from intended interventions
D3: Missing outcome data
D4: Measurement of the outcome
D5: Selection of the reported result

Each domain consists of 3-7 signaling questions, which help gather information and contribute to the final risk classification. For example, this D2 question assesses bias due to unblinded treatment

[3]ROB2 replaces its predecessor ROB after a formal evaluation identified areas for improvement (Sterne et al., 2019). ROB2 includes questions measuring newly identified ways that bias arise in randomized trials.

assignment: "Were participants aware of their assigned intervention during the trial?" There are five response options for each signaling question: (1) Yes; (2) Probably yes; (3) Probably no; (4) No; and (5) No information. All questions in App. A.

The ROB2 assessment is hierarchical. Signaling question responses for each domain first contribute to domain-level judgments for risk of bias, then domain-level judgments provide the basis for an overall risk of bias judgment. The tool provides flowcharts for computing the risk of each domain based on the answers to signaling questions (e.g., Figure 3 in App. A) as well as for computing overall risk. Domain-level and overall risk are assessed as either low risk, some concerns, or high risk.

In ROBOTO2, we model the ROB2 assessment as a document-level question-answering (QA) task. We use each signaling question as a query to retrieve relevant evidence passages from the trial report, then generate an answer based on the retrieved evidence. Answers are validated by a user who is conducting the assessment. The final risk assessment is produced by implementing the flowchart logic provided by the ROB2 tool.

4 ROBOTO2 System Pipeline

Figure 1 shows the ROBOTO2 pipeline. A user uploads a PDF of a clinical trial report. We (i) preprocess it to extract paragraphs of text; (ii) embed each paragraph using a document embedding model and index them for within-document retrieval; and then, for each signaling question from the ROB2 assessment tool, we (iii) embed the signaling question, retrieve the top-k similar paragraphs from the paper, and prompt an LLM to answer the question using the top-k paragraphs as context. We also experiment with providing all passages of text (full paper) as input for models with large input context windows. Details follow.

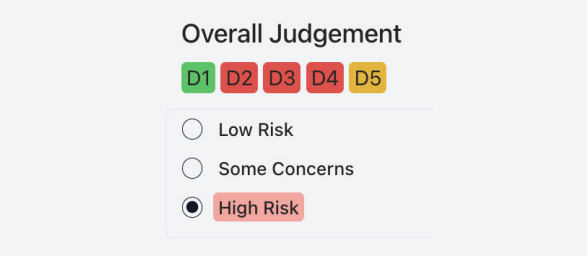
Preprocessing PDFs We convert each PDF into standardized JSON format using the S2ORC-doc2json library (Lo et al., 2020).[4] The output JSON contains a list of paragraphs in the paper, their section headers, and metadata elements such as the paper's title, authors, and abstract.

Embedding paragraphs for retrieval We compute embeddings for each paragraph in the uploaded paper using Sentence-Transformers all-MiniLM-L6-v2 (Reimers and Gurevych, 2019) and construct a key-value store for retrieval. Evaluation of the retriever and alternate methods is described in App. B. For each signaling question, we embed the question text using the same model and use cosine similarity to identify the top-k paragraphs to use as context for the QA module.

Answering signaling questions We then prompt an LLM to answer each ROB2 signaling question using an instruction prompt based on the ROB2 questionnaire and with the retrieved evidence paragraphs as context. Prompt templates and a complete example are given in App. C.

Collecting user feedback When conducting assessments with ROBOTO2, users can modify and provide feedback on all aspects of the assessment. While we show the top-3 retrieved paragraphs by default, users can add further paragraphs by selecting from the JSON parse. They can also provide feedback on the accuracy of retrieved passages via up- or downvotes, and modify the model-predicted answers and rationales (called "Explanation" in ROBOTO2). ROBOTO2 retains the original LLM responses and rationales, as well as the versions confirmed or edited by expert users. We include these user modifications as part of our dataset.

Domain-level and overall judgments We implement the logic provided by the ROB2 flowcharts (e.g., Figure 3 in App. A) to produce domain-level and overall risk of bias judgments. We visualize these at the end of each domain section and as a summative visualization when users complete the ROB2 assessment, e.g., three high risk domain-level judgments yields a "High Risk" overall rating visualized as follows:



System implementation Part of the ROBOTO2 workflow is shown in Figure 2 (web app at https://roboto2.vercel.app/). The web interface is written in React and Typescript. It leverages Transformers.js for client-side embedding and retrieval, and a back-end API in Python and FastAPI for document parsing and calls to LLM services.

[4] https://github.com/allenai/s2orc-doc2json

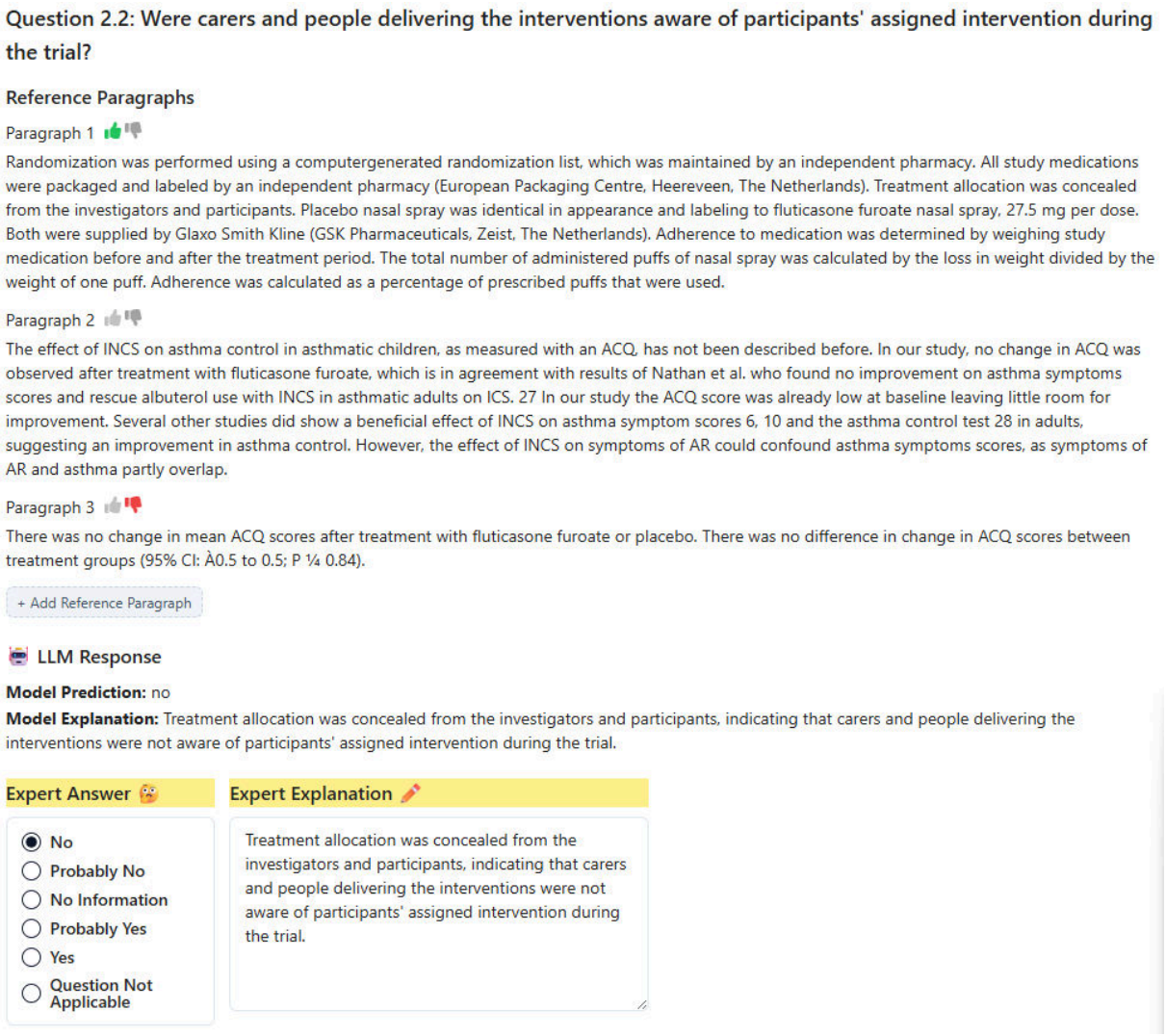


Figure 2: Screenshot of ROBOTO2 assisting with a question from Domain 2. The user can modify the model-provided answer and explanation and rate reference paragraphs.

5 Annotated Evaluation Dataset

Our dataset consists of 521 ROB2 assessments (245 completed using the Cochrance ROB2 Excel tool and 276 with LLM assistance via ROBOTO2). These ROB2 assessments were conducted as part of an independent research project aiming to systematically evaluate the risk of bias of all child health clinical trials; we re-purpose the data in this work to explore the role and feasibility of LLMs in supporting this aspect of systematic review.

Annotation procedure An initial corpus of child health clinical trial reports was constructed by searching the Cochrane Central Register of Controlled Trials, filtering for pediatric clinical trials based on the procedures described in Boluyt et al. (2008), and identifying 2334 matching clinical trial reports published 1991-2020. We sampled trial reports from this corpus for annotation.

For a subset of 245 reports, a group of expert raters completed assessments manually using the Cochrane tool, an Excel sheet with macros implementing the logic of the ROB2 assessment. To support judgments, annotators identified evidence passages manually from paper PDFs for a subset of questions and copied these into the Excel sheet. For each clinical trial, the data consists of the paper PDF for the trial report, as well as judgments for each signaling question, evidence passages extracted from the paper for a subset of questions, domain-level judgments, and the overall risk assessment score. Five expert annotators participated in annotations, and all annotators have graduate degrees in public health, epidemiology, medical sciences, or clinical practice, as well as experience conducting systematic reviews. This set of 245

	Low risk	Some concerns	High risk
Domain 1	234	243	44
Domain 2	287	171	63
Domain 3	450	35	35
Domain 4	406	60	54
Domain 5	332	272	34
Paper-level	64	301	156

Table 1: Distribution of domain- and paper-level risk of bias judgments in our dataset.

papers, annotated using the current gold-standard ROB2 review process, is used for all LLM evaluation and comparison reported in this paper.

An additional 276 papers were annotated separately by two of the five annotators using ROBOTO2, with LLM assistance. The version of ROBOTO2 used for annotations (collected during early 2024) used retrieval and GPT-3.5 (gpt-3.5-turbo-0125) as the answer model.[5] Because this sample of 276 papers was annotated with LLM assistance, we withheld them from the final evaluation of ROBOTO2 as described in Section 6, but include them in the published dataset to support future work.

Dataset statistics The distribution of domain-level and overall risk of bias judgments in the full dataset are provided in Table 1. Distributions of answered signaling questions and evidence paragraphs in the manually-annotated subset are shown at the top of Table 2.

Inter-rater reliability To assess inter-rater reliability, 20 papers (totaling 440 signaling questions) are independently annotated by two annotators using ROBOTO2. We aggregate answers into the following classes: Yes/Probably Yes, No/Probably No, No Information, and N/A, when a question is skipped by ROB logic. Four-class Cohen's Kappa is 0.40, indicating fair to moderate agreement. This is consistent with prior work showing slight to moderate agreement (Fleiss' Kappa of 0.45 at the domain level) among experienced raters (Minozzi et al., 2020, 2021); it reflects well-documented challenges of applying the complex ROB2 tool (Nejadghaderi et al., 2024), as reviewers can differ in their interpretation of ambiguous scenarios and their thresholds for assigning risk levels. Further commentary in App. D.

[5]In experiments, other LLMs and full-text context demonstrate better performance, but these were reasonable configurations at the time of annotation. Our public web interface supports the use of alternate LLMs.

6 Experimental Settings

We evaluate 4 models: GPT-3.5-Turbo, GPT-4o, Claude 3.5-Sonnet, and Llama-3.3-70B-Instruct. All models receive the same prompt and inputs (App. C). For all experiments, we aggregate labels and outputs into three classes: Yes/Probably Yes (Y/PY), No/Probably No (N/PN), and No Information (NI), and report micro-F1 at each domain level along with micro- and macro-averages across all signaling questions (Table 2).

Within-document retrieval We evaluate two retrieval methods: BM25 (Robertson et al., 1994) and paragraph embeddings using Sentence-Transformers (Reimers and Gurevych, 2019). Each signaling question has a max of one gold evidence passage in the dataset; we report recall@k for k=1,3,5,10 for all retrieval methods (Table 4). Detailed results and evaluation of the retrieval methods can be found in App B. In all cases, we use the questions from the ROB2 assessment as the query, and paragraphs from the clinical trial paper as the documents to retrieve. In the publicly available version of ROBOTO2, all-MiniLM-L6-v2 and k=3 were selected as these settings achieved competitive performance at low cost.

Prompting LLMs for QA We evaluate all models in a zero-shot setting with oracle evidence (providing the human-labeled evidence passage), as well as with the top-k retrieved evidence (with k=1,3,5), and the full paper setting for models with sufficient input context window sizes (all but GPT-3.5-Turbo). In the ROB2 assessment, each signaling question includes elaboration text that expands on when each answer should be chosen for that question; we provide this elaboration in the instructions for all prompting settings (example in App. C). We also conduct several experiments with in-context learning (Brown et al., 2020), which suggested minimal gains from the zero-shot setting; these results are reported in App E.

7 Results & Discussion

Results for all experimental settings are provided in Table 2. We analyze the model results and user statistics collected during ROBOTO2 annotations below (full statistics in App. G).

Room for improvement The best performing model (Claude 3.5-Sonnet with the full paper as context) achieved a micro-F1 of 0.71, highlighting considerable room for improvement. All evaluated

Model	Retrieval	D1	D2	D3	D4	D5	Micro-Avg	Macro-Avg
n-oracle	-	197	124	37	73	11	-	-
n-total	-	750	1278	598	1027	750	-	-
Baseline w/ oracle evidence paragraphs								
Llama-3.3-70B-Instruct	Oracle	0.67	0.55	0.35	0.81	0.45	0.62	0.67
GPT-3.5-Turbo	Oracle	0.81	0.66	0.67	0.82	0.57	0.61	0.71
GPT-4o	Oracle	0.75	0.78	0.68	0.92	0.54	0.64	0.73
Claude 3.5-Sonnet	Oracle	0.75	0.81	0.66	0.92	0.59	0.66	0.75
Retrieved evidence paragraphs								
Llama-3.3-70B-Instruct	k=1	0.83	0.61	0.42	0.80	0.38	0.49	0.61
GPT-3.5-Turbo	k=1	0.81	0.73	0.69	0.74	0.66	0.58	0.73
GPT-4o	k=1	0.68	0.65	0.58	0.81	0.75	0.55	0.69
Claude 3.5-Sonnet	k=1	0.69	0.68	0.66	0.87	0.76	0.60	0.73
Llama-3.3-70B-Instruct	k=3	0.87	0.69	0.66	0.81	0.35	0.55	0.68
GPT-3.5-Turbo	k=3	0.82	0.69	0.68	0.73	0.59	0.55	0.70
GPT-4o	k=3	0.75	0.72	0.73	0.82	0.72	0.60	0.75
Claude 3.5-Sonnet	k=3	0.75	0.75	0.76	0.87	**0.77**	0.65	0.78
Llama-3.3-70B-Instruct	k=5	0.87	0.72	0.70	0.81	0.28	0.55	0.69
GPT-3.5-Turbo	k=5	**0.82**	0.69	0.64	0.71	0.56	0.53	0.68
GPT-4o	k=5	0.78	0.74	0.73	0.83	0.69	0.62	0.75
Claude 3.5-Sonnet	k=5	0.78	0.79	0.78	0.86	**0.77**	0.67	0.80
Full paper as input								
Llama-3.3-70B-Instruct	Full Paper	0.88	0.81	0.79	0.79	0.32	0.61	0.72
GPT-4o	Full Paper	0.80	0.82	0.77	0.85	0.66	0.66	0.78
Claude 3.5-Sonnet	Full Paper	0.81	**0.84**	**0.80**	**0.88**	**0.77**	**0.71**	**0.82**

Table 2: For all settings, we report micro-averaged domain-level F1 along with micro- and macro-averaged F1 across all signaling questions. The n-oracle is the number of instances where annotators identified an evidence passage, while n-total is the total number of signaling questions answered for that domain.

models achieve strong results in D1, where questions are more likely to be answerable based on text in the trial reports. Performance in D2 and D3 is mixed, as these may require interpreting numerical data (e.g., calculating attrition rates from recruitment and result numbers). D5 is similarly challenging for models as these questions may require knowledge of external clinical resources and guidelines.

Increasing context generally improves performance for most models while reducing accuracy for GPT-3.5-Turbo. In some cases, models with retrieval can surpass oracle retrieval performance, likely due to incomplete evidence labeling in our dataset, indicating that relevant information exists beyond annotator-selected passages. Few-shot prompting does not appear to outperform zero-shot prompting in our experimental results (App E).

Limited utility for fully-automated ROB assessment Model performance cannot be substituted for human judgment in ROB assessments, and we recommend that humans remain in the loop. Conservative question-level model judgments compound to conservative domain- and paper-level judgments, where the fully-automated pipeline judges most papers as having "some concerns" or "high risk" even when human raters did not. Human raters assessed 47 of 276 trials as high risk while the LLM-only pipeline assessed 101 as high risk. Error analysis (App. F) reveals that the strongest models tend to over-select "No Information," which may reflect cautiousness gained from safety and alignment training.

ROBOTO2 supports human review and editing We compute detailed metrics for the 276 ROB2 assessments annotated using ROBOTO2, including the number of times annotators accept the model's answers and explanations directly versus change them, and the number of retrieved evidence passages marked as good (offering evidence to support an answer) versus bad (irrelevant). Annotators provide their own answer (42.4%) and edit rationales (28.7%) around half the time, rather than use the answer (57.6%) and rationales (71.3%) provided by the model (Table 7). For evidence passages, 615 total up/downvotes are collected (out of 3370 retrieved passages), of which 78.0% are positive feedback. We provide all feedback in our dataset to support future model development. Detailed statistics can be found in App. G.

8 Conclusion

Assessing the quality of clinical trials is an important step to weighing their evidence in clinical decision-making. To support this, we introduce the ROBOTO2 system to assist researchers in conducting risk of bias assessment for clinical trials with LLM support, along with associated code for running the web interface. We also release a dataset of 521 complete ROB analyses (8954 signaling questions with 1202 evidence passages) of child health clinical trial reports. We hope our system and dataset will promote better LLM applications for risk of bias assessment, and that access to this assisted annotation tool can enable quicker completion of ROB assessments and reduce the labor and costs around systematic literature reviews.

9 Limitations

Viewing model outputs could potentially bias annotations ROBOTO2 is designed to expose all intermediate and final model outputs, and allows expert annotators to change any part of the model output. While we can compute the number of changes made, we cannot guarantee that seeing model outputs does not influence annotator responses. Prior work has shown that human annotators may demonstrate anchoring bias when exposed to LLM assistance during annotation (Choi et al., 2024), leading to discrepancies in downstream label distributions. We leave the measurement of this bias in the ROB setting to future work.

Dataset imbalance Though it reflects real-world ROB2 assessments, our dataset is unbalanced. The majority of papers are assessed as having some concerns, with fewer papers of low or high risk. This likely biases the evaluation of our system, similar to what was observed by Suster et al. (2021). Related, some signaling questions have very sparse annotations (especially those that depend on cascading logic) or are biased in terms of answer distribution (almost always one of the answer labels).

Among manually conducted reviews, annotator-provided evidence paragraphs are only available for a small portion of signaling questions, unbalanced across domains; D1 has the most signaling questions with evidence passages, while D5 has very few. Our retrieval methods as well as Oracle results are only reported on this biased subset, and may not accurately represent performance on sparsely annotated questions and domains.

Potential gains in quality or efficiency We hypothesize that ROBOTO2 may either help to save time or improve the quality of ROB assessments. Qualitatively, the annotation team reported that ROBOTO2 offers an opportunity to enhance evidence and rationale coverage, as the time savings on the ROB assessment itself were repurposed to judge and retrieve relevant evidence. However, the impact of such repurposing of effort on the quality of the resulting assessments was not explicitly measured in our system and should be confirmed and studied in future work.

Diversity of language models in experiments Our experiments do not include any reasoning models such as OpenAI's o3, Anthropic's Claude 4-Sonnet-Thinking, and DeepSeek-R1 (DeepSeek-AI, 2025). Future work could explore whether these models improve current performance in terms of both answer classification accuracy and generated rationales.

Acknowledgements

We thank the following individuals for producing parts of the ROB dataset we use for evaluation: Bernadette Zakher, Shannon Sim, Michele Dyson, Banke Oketola, and Aneet Saran, from the University of Victoria, University of Alberta, and University of Manitoba. This work is supported in part by the University of Washington Information School Strategic Research Fund and the University of Washington eScience Institute's Azure Cloud Credits for Research and Teaching.

References

Ahmad Alshami, Moustafa Elsayed, Eslam Ali, Abdelrahman E. E. Eltoukhy, and Tarek M. Zayed. 2023. Harnessing the power of chatgpt for automating systematic review process: Methodology, case study, limitations, and future directions. *Syst.*, 11:351.

Nicole Boluyt, Lisa Tjosvold, Carol Lefebvre, Terry P Klassen, and Martin Offringa. 2008. Usefulness of systematic review search strategies in finding child health systematic reviews in medline. *Archives of pediatrics & adolescent medicine*, 162 2:111–6.

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.

Alexander S. Choi, Syeda Sabrina Akter, JP Singh, and Antonios Anastasopoulos. 2024. The LLM effect:

Are humans truly using LLMs, or are they being influenced by them instead? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22032–22054, Miami, Florida, USA. Association for Computational Linguistics.

Miew Keen Choong, Filippo Galgani, Adam G. Dunn, and Guy Tsafnat. 2014. Automatic evidence retrieval for systematic reviews. *Journal of Medical Internet Research*, 16.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Jay DeYoung, Iz Beltagy, Madeleine van Zuylen, Bailey Kuehl, and Lucy Lu Wang. 2021. MSˆ2: Multi-document summarization of medical studies. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7494–7513, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jay DeYoung, Eric Lehman, Benjamin Nye, Iain Marshall, and Byron C. Wallace. 2020. Evidence inference 2.0: More data, better models. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 123–132, Online. Association for Computational Linguistics.

Julian PT Higgins, Douglas G Altman, Peter C Gotzsche, Peter Juni, David Moher, Andrew D Oxman, Jelena Savovic, Kenneth F Schulz, Laura Weeks, and Jonathan AC Sterne. 2011. The cochrane collaboration's tool for assessing risk of bias in randomised trials. *BMJ*, 343.

Yan Hu, Vipina Kuttichi Keloth, Kalpana Raja, Yong Chen, and Hua Xu. 2023. Towards precise pico extraction from abstracts of randomized controlled trials using a section-specific learning approach. *Bioinformatics*, 39.

Patricia Sofia Jacobsen Jardim, Christopher James Rose, Heather Melanie R Ames, J. Meneses Echavez, Stijn Van de Velde, and Ashley Elizabeth Muller. 2022. Automating risk of bias assessment in systematic reviews: a real-time mixed methods comparison of human researchers to a machine learning system. *BMC Medical Research Methodology*, 22.

Di Jin and Peter Szolovits. 2018. Pico element detection in medical text via long short-term memory neural networks. In *Workshop on Biomedical Natural Language Processing*.

Hanan Khalil, Daniel Ameen, and A. Zarnegar. 2021. Tools to support the automation of systematic reviews: A scoping review. *Journal of clinical epidemiology*.

Eric Lehman, Jay DeYoung, Regina Barzilay, and Byron C. Wallace. 2019. Inferring which medical treatments work from reports of clinical trials. 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 3705–3717, Minneapolis, Minnesota. Association for Computational Linguistics.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Michael Kinney, and Daniel S. Weld. 2020. S2orc: The semantic scholar open research corpus. In *Annual Meeting of the Association for Computational Linguistics*.

Iain J Marshall, Joel Kuiper, and Byron C Wallace. 2016. Robotreviewer: evaluation of a system for automatically assessing bias in clinical trials. *Journal of the American Medical Informatics Association*, 23(1):193–201.

Iain James Marshall, Joël Kuiper, and Byron C. Wallace. 2014. Automating risk of bias assessment for clinical trials. *IEEE Journal of Biomedical and Health Informatics*, 19:1406–1412.

Iain James Marshall and Byron C. Wallace. 2019. Toward systematic review automation: a practical guide to using machine learning tools in research synthesis. *Systematic Reviews*, 8.

Silvia Minozzi, Michela Cinquini, Silvia Gianola, Marien González-Lorenzo, and Rita Banzi. 2020. The revised cochrane risk-of-bias tool for randomised trials (rob 2) showed low inter-rater reliability and challenges in its application. *Journal of clinical epidemiology*.

Silvia Minozzi, Kerry Dwan, Francesca Borrelli, and Graziella Filippini. 2021. Reliability of the revised cochrane risk-of-bias tool for randomised trials (rob2) improved with the use of implementation instruction. *Journal of clinical epidemiology*.

Aakanksha Naik, Bailey Kuehl, Erin Bransom, Doug Downey, and Tom Hope. 2024. CARE: Extracting experimental findings from clinical literature. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 4580–4596, Mexico City, Mexico. Association for Computational Linguistics.

Seyed Aria Nejadghaderi, Maryam Balibegloo, and Nima Rezaei. 2024. The cochrane risk of bias assessment tool 2 (rob 2) versus the original rob: A perspective on the pros and cons. *Health Science Reports*, 7.

Benjamin Nye, Junyi Jessy Li, Roma Patel, Yinfei Yang, Iain Marshall, Ani Nenkova, and Byron Wallace. 2018. A corpus with multi-level annotations of patients, interventions and outcomes to support language processing for medical literature. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 197–207, Melbourne, Australia. Association for Computational Linguistics.

Jason Portenoy and Jevin D. West. 2020. Constructing and evaluating automated literature review systems. *Scientometrics*, 125:3233 – 3251.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *Text Retrieval Conference*.

Olivia Sanchez-Graillet, Christian Witte, Frank Grimm, Steffen Grautoff, Basil Ell, and Philipp Cimiano. 2022. Synthesizing evidence from clinical trials with dynamic interactive argument trees. *Journal of Biomedical Semantics*, 13.

Chantal Shaib, Millicent Li, Sebastian Joseph, Iain Marshall, Junyi Jessy Li, and Byron Wallace. 2023. Summarizing, simplifying, and synthesizing medical evidence using GPT-3 (with varying success). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1387–1407, Toronto, Canada. Association for Computational Linguistics.

Jonathan A. C. Sterne, Jelena Savović, Matthew J. Page, Roy G Elbers, Natalie S Blencowe, Isabelle Boutron, Christopher J Cates, Hung-Yuan Cheng, Mark S. Corbett, Sandra Eldridge, Jonathan R. Emberson, Miguel A. Hernán, Sally Hopewell, Asbjørn Hrõbjartsson, Daniela R. Junqueira, Peter Juni, Jamie J. Kirkham, Toby J Lasserson, Tianjing Li, Alexandra McAleenan, Barnaby C. Reeves, Sasha Shepperd, Ian Shrier, Lesley A Stewart, Kate Tilling, Ian R. White, Penny F. Whiting, and Julian P. T. Higgins. 2019. Rob 2: a revised tool for assessing risk of bias in randomised trials. *BMJ*, 366.

Simon Suster, Timothy Baldwin, Jey Han Lau, Antonio Jimeno Yepes, David Martinez Iraola, Yulia Otmakhova, and Karin M. Verspoor. 2021. Automating quality assessment of medical evidence in systematic reviews: Model development and validation study. *Journal of Medical Internet Research*, 25.

Rens van de Schoot, Jonathan de Bruin, Raoul Schram, Parisa Zahedi, Jan de Boer, Felix Weijdema, Bianca Kramer, Martijn Huijts, Maarten Hoogerwerf, Gerbrich Ferdinands, Albert Harkema, Joukje E Willemsen, Yongchao Ma, Qixiang Fang, Lars G Tummers, and Daniel L. Oberski. 2021. Asreview: Active learning for systematic reviews.

Byron C. Wallace, Joel Kuiper, Aakash Sharma, Mingxi Zhu, and Iain James Marshall. 2016. Extracting pico sentences from clinical trial reports using supervised distant supervision. *Journal of machine learning research (JMLR)*, 17.

Byron C. Wallace, Sayantani Saha, Frank Soboczenski, and Iain James Marshall. 2020. Generating (factual?) narrative summaries of rcts: Experiments with neural multi-document summarization. *AMIA Annual Symposium proceedings*, 2021:605–614.

Lucy Lu Wang, Jay DeYoung, and Byron Wallace. 2022. Overview of MSLR2022: A shared task on multi-document summarization for literature reviews. In *Proceedings of the Third Workshop on Scholarly Document Processing*, pages 175–180, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Hye Sun Yun, David Pogrebitskiy, Iain James Marshall, and Byron C. Wallace. 2024. Automatically extracting numerical results from randomized controlled trials with large language models. *ArXiv*, abs/2405.01686.

A ROB2 Assessment Tool

The Signaling Questions for the Cochrane ROB2 Tool for Randomized Trials are given in Table 3. Note that some questions are *cascading*, and are only answered if previous questions in the domain are answered in a pre-specified way.

Figure 3 reproduces a flowchart from the ROB2 tool that indicates how signaling questions contribute to a domain-level judgment for Domain 4. Based on how these questions are answered, the domain-level judgment can be low risk, some concerns, or high risk. Flowcharts are also provided for the other four domains and are available at https://methods.cochrane.org/risk-bias-2.

B Retriever Evaluation

We experiment with a sparse retriever, BM25 (Robertson et al., 1994), and Sentence-Transformers (Reimers and Gurevych, 2019). We assess retriever performance by varying k, the number of passages retrieved and provided to the QA reader module. For models with large context windows, we also experiment with providing the entire paper as context.

All methods are validated using gold evidence paragraphs identified by the annotators in our dataset. Each signaling question has a maximum of one gold evidence passage in the dataset; we report recall@k for k=1,3,5,10 for all retrieval methods (Table 4).

For within-document retrieval, we find that S-BERT successfully retrieves the gold evidence passage at a higher rate than BM25 at comparable k (Table 4). However, prompting models with the full paper achieves the highest overall F1-scores (Table 2). ROBOTO2 uses S-BERT for its balance between performance, speed, and enabling models with smaller context windows to be used in the web interface.

Domain	Question
Domain 1: Risk of bias arising from the randomization process	
1.1	Was the allocation sequence random?
1.2	Was the allocation sequence concealed until participants were enrolled and assigned to interventions?
1.3	Did baseline differences between intervention groups suggest a problem with the randomization process?
Domain 2: Risk of bias due to deviations from the intended interventions	
2.1	Were participants aware of their assigned intervention during the trial?
2.2	Were carers and people delivering the interventions aware of participants' assigned intervention during the trial?
2.3	If Y/PY/NI to 2.1 or 2.2: Were there deviations from the intended intervention that arose because of the trial context?
2.4	If Y/PY to 2.3: Were these deviations likely to have affected the outcome?
2.5	If Y/PY/NI to 2.4: Were these deviations from intended intervention balanced between groups?
2.6	Was an appropriate analysis used to estimate the effect of assignment to intervention?
2.7	If N/PN/NI to 2.6: Was there potential for a substantial impact (on the result) of the failure to analyse participants in the group to which they were randomized?
Domain 3: Risk of bias due to missing outcome data	
3.1	Were data for this outcome available for all or nearly all participants randomized?
3.2	If N/PN/NI to 3.1: Is there evidence that the result was not biased by missing outcome data?
3.3	If N/PN to 3.2: Could missingness in the outcome depend on its true value?
3.4	If Y/PY/NI to 3.3: Is it likely that missingness in the outcome depended on its true value?
Domain 4: Risk of bias in measurement of the outcome	
4.1	Was the method of measuring the outcome inappropriate?
4.2	Could measurement or ascertainment of the outcome have differed between intervention groups?
4.3	If N/PN/NI to 4.1 and 4.2: Were outcome assessors aware of the intervention received by study participants?
4.4	If Y/PY/NI to 4.3: Could assessment of the outcome have been influenced by knowledge of intervention received?
4.5	If Y/PY/NI to 4.4: Is it likely that assessment of the outcome was influenced by knowledge of intervention received?
Domain 5: Risk of bias in selection of the reported result	
5.1	Were the data that produced this result analysed in accordance with a pre-specified analysis plan that was finalized before unblinded outcome data were available for analysis?
5.2	Is the numerical result being assessed likely to have been selected, on the basis of the results, from multiple eligible outcome measurements (e.g., scales, definitions, time points) within the outcome domain?
5.3	Is the numerical result being assessed likely to have been selected, on the basis of the results, from multiple eligible analyses of the data?

Table 3: Signaling Questions in the Cochrane Risk of Bias Tool for Randomized Trials (ROB2).

Model	R@1	R@3	R@5	R@10
BM25	0.140	0.272	0.367	0.533
S-BERT	0.268	0.455	0.519	0.678

Table 4: Recall@k for our tested retrieval methods.

C Prompting

The prompt is formatted as follows:

```
<instruction>
<signaling_question>
<elaboration>
<retrieved_paragraph_1>
...
<retrieved_paragraph_k>
```

After an instruction to answer the question, we provide the signaling question itself from the ROB2 assessment, as well as additional elaboration text explaining all answer options and when they should be used. These elaborations are adapted from explanations given in the ROB2 tool, and we further augment them such that all possible answer options are represented—not all answers are represented in elaborations from the original ROB2 tool, which we found may bias models towards answers that were. Retrieved context paragraphs are then appended. We instruct the model to make a prediction and generate a rationale for its prediction.

A full example prompt for signaling question 1 in Domain 1 is reproduced below:

```
You are an expert scientific researcher.
```

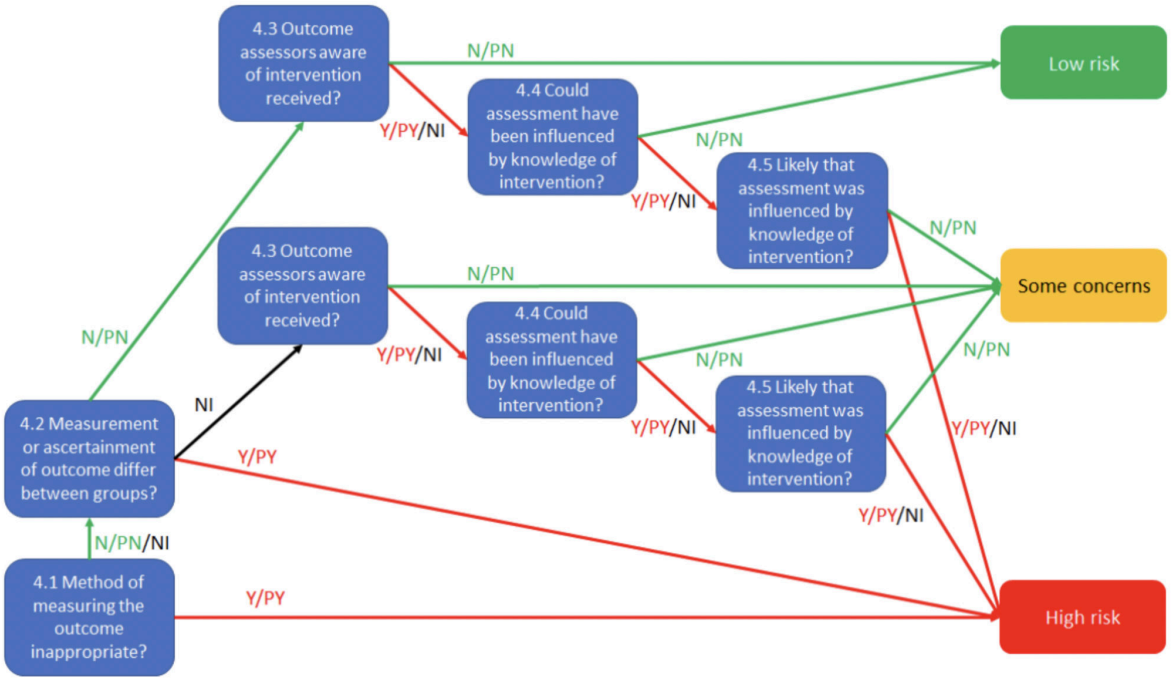


Figure 3: Flowchart for how answers to signaling questions contribute to a domain-level judgment for Domain 4 in the ROB2 tool. Reproduced from https://sites.google.com/site/riskofbiastool/welcome/rob-2-0-tool.

```
You will be given a passage from
a scientific paper reporting on a
randomized controlled trial along
with a question and elaboration of
the question. Your task is to return
the answer to the question out of the
following set of answers: "yes", "no",
"probably yes", "probably no", "no
information". You should use the given
passage to answer the question.

Question: "Was the allocation sequence
random?"

Elaboration: "Answer 'Yes' if a random
component was used in the sequence
generation process. Examples include
computer-generated random numbers;
reference to a random number table;
coin tossing; shuffling cards or
envelopes; throwing dice; or drawing
lots. Minimization is generally
implemented with a random element (at
least when the scores are equal), so an
allocation sequence that is generated
using minimization should generally be
considered to be random.

Answer 'No' if no random element was
used in generating the allocation
sequence or the sequence is predictable.
Examples include alternation; methods
based on dates (of birth or admission);
patient record numbers; allocation
decisions made by clinicians or
participants; allocation based on the
availability of the intervention; or any
other systematic or haphazard method.

Answer 'No information' if the only
information about randomization methods
is a statement that the study is
randomized.

In some situations a judgment may
be made to answer 'Probably no' or
'Probably yes'. For example, in the
context of a large trial run by an
experienced clinical trials unit,
absence of specific information
about generation of the randomization
sequence, in a paper published in a
journal with rigorously enforced word
count limits, is likely to result in
a response of 'Probably yes' rather
than 'No information'. Alternatively,
if other (contemporary) trials by the
same investigator team have clearly
used non-random sequences, it might be
reasonable to assume that the current
study was done using similar methods."

Passage(s):
<retrieved_paragraph_1>
...
<retrieved_paragraph_k>
```

D Further Commentary on Inter-Rater Reliability Analysis

Two independent reviewers independently assessed risk of bias for 20 trials using the revised Cochrane ROB2 tool. These assessments were used to compute IAA as reported in the main paper. Following these annotations, the two reviewers conducted a consensus meeting to better understand discrepancies arising in their annotations. The discussion process involved revisiting the Cochrane Handbook and the official ROB2 guidance document (Higgins et al., 2011; Sterne et al., 2019) to ensure alignment with recommended best practices.

Notable discrepancies emerged in this meeting, classified into four main categories: disagreement at the signaling question level, disagreement at the domain level, differences in judgments between Yes and Probably Yes, and No and Probably No. One reviewer tended to adopt a more conservative approach and tended to opt for "some concerns" or "high risk" judgments whereas the other reviewer more frequently opted for "low risk" ratings when the available information appeared sufficient. This divergence was typical of what has been described in prior research on ROB2, which has noted that even experienced reviewers may differ in how they interpret the level of concern warranted by ambiguous or incomplete reporting (Minozzi et al., 2020).

Following this consensus meeting, the reviewers were able to reach full agreement across all signaling questions and domains. This calibration process is useful for achieving subsequent consistent application of the ROB2 assessment tool.

E Few-Shot Prompting Results

Results from few-shot prompting experiments are shown in Table 5. The few-shot prompt is created by sampling one example for each class from the gold label annotations, using the same prompt template in App C. The oracle paragraph is provided for each example, the elaboration for the signaling questions is removed (due to token constraints), and the answer is appended to the end of the prompt in the form `Answer:<Label>`. Few shot examples sampled are removed from the evaluation set for models. No substantial differences are observed between the zero- and few-shot settings when models are provided the same number of context passages.

F LLM Error Analysis

The 4 LLMs we evaluated exhibit different patterns of answers, though the evaluation metrics are comparable. Performance across models according to micro- and macro-averaged F1 across domains suggests similar performance, qualitative performance is very different between models. We plot error counts in Figure 4, showing true positives (TP), alongside each of the two types of false positives (FPs) and false negatives (FNs). Here, class 1 FP/FN errors are those considered to be less severe (e.g., Y/PY wrongly classified as NI is not as severe as Y/PY wrongly classified as N/PN). We also provide raw counts of these errors in Table 6.

As seen in the figure, GPT-4o is more likely than other LLMs to answer "No Information" (large number of FP in the first column) or "No/Probably No". Llama-3.3-70B-Instruct, Claude-3.5-Sonnet, and GPT-4o most often predicted "No Information" for signaling questions where the true label was No/Probably No. On the other hand, GPT-3.5-Turbo almost never abstains with a "No Information" prediction, leading to more false positive errors for the N/PN and Y/PY classes. Claude 3.5-Sonnet was the best performing model evaluated and has fairly comparable false positive rates across "No/Probably No" and "Yes/Probably Yes". Llama-3.3-70b-Instruct demonstrates a similar answer distribution to Claude 3.5-Sonnet, but with a consistent false positive rate across all 3 classes, but higher false negatives, with "No/Probably No" being the highest.

These observed behaviors for over-predicting "No Information" or "No/Probably No" could stem from safety mechanisms learned during model post-training, which might explain GPT-3.5-Turbo's extreme bias towards "No/Probably No" and "Yes/Probably Yes". Some domains have questions phrased in a way that requires interpreting numerical data (D2 & D3) or understanding current best practices in the field (D5); stronger models tend to abstain in these cases and select "No Information". However, in the context of ROB2 assessments, these cautious predictions lead to over-conservative domain- and paper-level labels (high risk judgments) for LLM-supported assessments.

G ROBOTO2 Usage Statistics

Acceptance rates of model answers and rationales versus user-corrected rates are provided in Table 7. Our annotation interface allows users to rate the 3

Model	Retrieval	D1	D2	D3	D4	D5	Micro-avg	Macro-avg
n-total	-	750	1278	598	1027	750	-	-
Zero-shot setting								
Llama-3.3-70B-Instruct	k=1	0.83	0.61	0.42	0.80	0.38	0.49	0.61
GPT-3.5-Turbo	k=1	0.81	0.73	0.69	0.74	0.66	0.58	0.73
GPT-4o	k=1	0.68	0.65	0.58	0.81	0.75	0.55	0.69
Claude 3.5-Sonnet	k=1	0.69	0.68	0.66	0.87	0.76	0.60	0.73
Few-shot setting								
Llama-3.3-70B-Instruct (FS)	k=1	0.83	0.61	0.43	0.76	0.38	0.49	0.61
GPT-3.5-Turbo (FS)	k=1	0.81	0.70	0.65	0.80	0.71	0.60	0.73
GPT-4o (FS)	k=1	0.70	0.64	0.60	0.83	0.70	0.55	0.69
Claude 3.5-Sonnet (FS)	k=1	0.69	0.68	0.65	0.87	0.77	0.60	0.73

Table 5: Micro-averaged domain-level F1 along with micro- and macro-averaged F1 across signaling questions for few shot retrieval. Zero-shot k=1 results from Table 2 are reproduced here for reference.

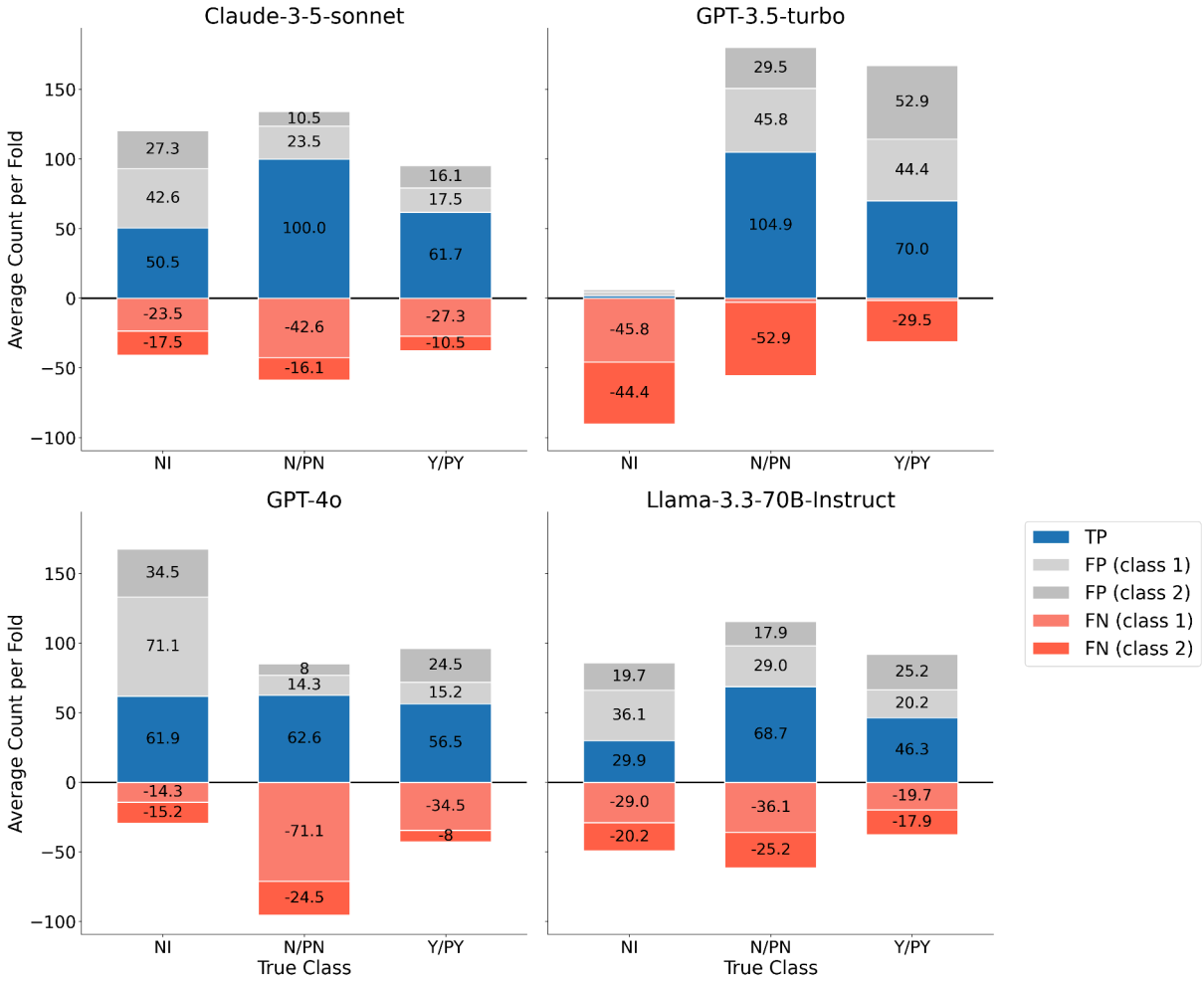


Figure 4: Stacked bar chart showcasing the aggregate true positive (TP) classifications versus false positive/negative (FP/FN) errors made by each model. FPs and FNs are each broken down into two classes, where class 1 (lighter color) are milder errors than class 2 (darker color) (e.g., misclassifying NI and N/PN or Y/PY is less severe than misclassifying N/PN as Y/PY or vice versa). Counts less than 3 have their numbers hidden for chart readability, and full counts are available in Table 6.

retrieved paragraphs with a good/bad rating and/or add their own paragraphs from the paper as context. Feedback statistics for retrieved evidence paragraphs are given in Table 8.

Model	Class	True Positives	FP (class 1)	FP (class 2)	FN (class 1)	FN (class 2)
Llama-3.3-70B-Instruct	NI	29.9	36.1	19.7	29.0	20.2
	N/PN	68.7	29.0	17.9	36.1	25.2
	Y/PY	46.3	20.2	25.2	19.7	17.9
GPT-3.5-turbo	NI	1.8	2.8	1.6	45.8	44.4
	N/PN	104.9	45.8	29.5	2.8	52.9
	Y/PY	70.0	44.4	52.9	1.6	29.5
GPT-4o	NI	61.9	71.1	34.5	14.3	15.2
	N/PN	62.6	14.3	8.3	71.1	24.5
	Y/PY	56.5	15.2	24.5	34.5	8.3
Claude-3.5-Sonnet	NI	50.5	42.6	27.3	23.5	17.5
	N/PN	100.0	23.5	10.5	42.6	16.1
	Y/PY	61.7	17.5	16.1	27.3	10.5

Table 6: Confusion-matrix summary for the LLMs when answering ROB2 signaling questions. For each response category Yes/Probably Yes (Y/PY), No/Probably No (N/PN), and No Information (NI), we list the number of true positives (TP) and the false positive (FP) and false negative (FN) counts accrued against the two alternative classes ("class 1" and "class 2"). Higher TP and lower FP/FN values reflect better agreement with the gold label. All values are normalized averages across run configurations.

	Predictions		**Rationales**	
Domain	Model (%)	Expert (%)	Model (%)	Expert (%)
1	377 (49.2%)	390 (50.8%)	430 (56.1%)	337 (43.9%)
2	853 (59.2%)	588 (40.8%)	1117 (77.5%)	324 (22.5%)
3	432 (65.2%)	231 (34.8%)	494 (74.5%)	169 (25.5%)
4	591 (64.6%)	325 (35.4%)	717 (78.3%)	199 (21.7%)
5	368 (48.2%)	396 (51.8%)	485 (63.5%)	279 (36.5%)
Total	2621 (57.6%)	1930 (42.4%)	3243 (71.3%)	1308 (28.7%)

Table 7: Counts and percentages of model-originated versus expert-corrected predictions and explanations across domains for ROB assessments completed using ROBOTO2.

Domain	Downvotes	Upvotes	User Added Paragraphs
1	43	207	74
2	40	120	84
3	12	48	24
4	22	64	112
5	18	41	62
Total	135	480	356

Table 8: Feedback on evidence passages provided by users by domain at the question level, for the ROBOTO2 subset, i.e., each number corresponds to the number of questions in that domain for which a user provided a downvote, an upvote, or added paragraph, as opposed to the total number of downvotes or upvotes etc.

SpiritRAG: A Q&A System for Religion and Spirituality in the United Nations Archive

Yingqiang Gao[†UZH LiRI] **Fabian Winiger**[‡UZH TRF]
Patrick Montjourides[‡UZH IFE] **Anastassia Shaitarova**[‡UZH LiRI]
Nianlong Gu[UZH LiRI] **Simon Peng-Keller**[UZH TRF] **Gerold Schneider**[UZH LiRI]

[UZH LiRI] Linguistic Research Infrastructure, University of Zurich, Switzerland
[UZH TRF] Faculty of Theology and the Study of Religion, University of Zurich, Switzerland
[UZH IFE] Institute of Education, University of Zurich, Switzerland
{yingqiang.gao, fabian.winiger}@uzh.ch
https://SpiritRAG.linguistik.uzh.ch

Abstract

Religion and spirituality (R/S) are complex and highly domain-dependent concepts which have long confounded researchers and policymakers. Due to their context-specificity, R/S are difficult to operationalize in conventional archival search strategies, particularly when datasets are very large, poorly accessible, and marked by information noise. As a result, considerable time investments and specialist knowledge is often needed to extract actionable insights related to R/S from general archival sources, increasing reliance on published literature and manual desk reviews. To address this challenge, we present SpiritRAG, an interactive Question Answering (Q&A) system based on Retrieval-Augmented Generation (RAG). Built using 7,500 United Nations (UN) resolution documents related to R/S in the domains of health and education, SpiritRAG allows researchers and policymakers to conduct complex, context-sensitive database searches of very large datasets using an easily accessible, chat-based web interface. SpiritRAG is lightweight to deploy and leverages both UN documents and user provided documents as source material. A pilot test and evaluation with domain experts on 100 manually composed questions demonstrates the practical value and usefulness of SpiritRAG.[*]

Dataset Code

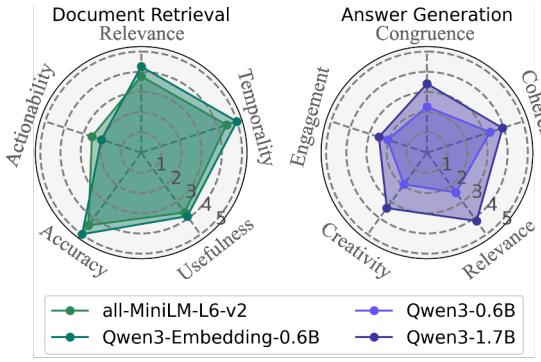


Figure 1: Human evaluation results of SpiritRAG on 50 health + R/S test questions listed in Appendix A. Our best-performing system setup received promising assessments from domain experts.

1 Introduction

The concepts of "religion" and "spirituality" (R/S) are complex and highly domain-dependent, and vary substantially across social and cultural contexts. The meaning and connotation of R/S poses unique challenges to historians, sociologists, anthropologists, and other scholars who seek to understand R/S across distinct social and historical settings. Definitions of R/S vary widely across academic disciplines, and scholars have yet to reach a consensus on how to define them in general terms (Steensland et al., 2022; Winiger et al., 2025).

For policymakers, the complexity and domain-dependence of these terms are significant barriers to research and decision-making. This is illustrated by the long and protracted history of R/S in the United Nations (UN) system, the main locus of intergovernmental cooperation in the pursuit of global values for humanity, where R/S regularly surfaces as a matter of mutual concern in the areas of healthcare (Peng-Keller et al., 2022), Mental Health and Psychosocial Support (MHPSS) in humanitarian settings (Ager et al., 2019), soft diplomacy and value politics (Stensvold, 2017; Carrette, 2017; Baumgart-Ochse and Wolf, 2019; Steiner and Christie, 2021), and various develop-

[†]Corresponding author.
[‡]Equal contribution.
[*]SpiritRAG is under MIT license.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 26–41
November 4-9, 2025

ment and peace building efforts (UN Inter-Agency Task Force on Religion and Development, 2020). Accessing and analyzing documents produced by the organizations and agencies of the UN often require extended and time-consuming archival research using poorly designed or outdated search interfaces. This is exemplified by the official archive of the UN[2], which includes a vast collection of documents produced by the Security Council, the General Assembly, the Economic and Social Council (ECOSOC) and their subsidiaries, and others. Additionally, some UN entities, such as the World Health Organization (WHO) and the United Nations Educational, Scientific, and Cultural Organization (UNESCO), maintain separate archives.

The problem of accessibility is greatly exacerbated when searching for information on relatively niche, complex and domain-dependent concepts such as R/S, in which case the retrieval of actionable information requires a substantial investment of time to manually search very large numbers of often largely irrelevant documents marked by information "noise" such as procedural formalities, preambular clauses, extensive appendices, and frequently used phrases which contain keywords used in an unrelated context (e.g, "in the spirit of [...]"). The combination of large, poorly accessible and noisy databases with search queries related to complex and domain-dependent topics such as R/S underscores the urgent need for systems that can quickly and easily retrieve, synthesize and generate knowledge based on complex and nuanced user queries, providing scholars and policymakers with low-barrier access to accurate and reliable information on complex topics.

Current Large Language Models (LLMs) and general-purpose Information Retrieval (IR) systems often produce outputs that lack contextual sensitivity, thereby risking the homogenization of nuanced perspectives exemplified by the discourse on R/S (Hutchinson, 2024; Liu et al., 2024; Plaza-del Arco et al., 2024). Recent studies have highlighted that LLMs may exhibit contextual biases, particularly when predicting public opinion across diverse national and linguistic settings, leading to inaccuracies and a failure to capture cultural subtleties (Qu and Wang, 2024; von der Heyde et al., 2024). Furthermore, evaluations of LLMs' cultural alignment reveal significant limitations, including instability across different cultural dimensions and challenges in consistently steering models to represent specific cultural perspectives (Masoud et al., 2025; Khan et al., 2025). These findings underscore the need for specialized knowledge systems capable of retrieving and generating knowledge in information environments marked by high context- and domain dependence and cultural complexity. However, no existing systems were designed for knowledge acquisition in the context of R/S.

To address this gap, we present SpiritRAG—a Retrieval-Augmented Generation (RAG; Lewis et al. (2020)) Question Answering (Q&A) system designed to retrieve, synthesize, and generate R/S-related knowledge based on the resolutions stored in the UN archive. For present purposes, the corpus was built using data on health and education, with the latter included because many education-related documents are implicitly related to health (e.g., on tobacco control or sexual and reproductive care, see Sustainable Development Goal 3 (United Nations, 2024)). SpiritRAG is built on the top of approx. 7,500 documents available in seven languages, and supports interactive querying with real-time answers linked to exact reference documents. To evaluate the system, we compiled an expert-written test set of R/S-related questions in health and education. Pilot tests show that SpiritRAG is perceived by social scientists as a highly innovative and potentially groundbreaking interactive knowledge engine, highlighting its potential as a proof-of-concept for future LLM-augmented work in public policy, historical scholarship, and more generally, in the field of digital humanities.

Our main contributions are: (1) We developed an interactive Q&A system for retrieving R/S-related knowledge; (2) We built a corpus of nearly 7,500 UN resolution documents on health and education subjects; (3) We created a test set of 100 evaluation questions authored by domain experts.

A video demonstration of SpiritRAG can be accessed at https://youtu.be/x_G8bopwp8A?si=irrrlslw_i_ryRCX.

2 SpiritRAG: A Q&A System for R/S

SpiritRAG adopts a similar modular design principle as proposed by Gao et al. (2024) (see Figure 2). In the following, we discuss the core components of the proposed system.

2.1 Corpus

We constructed the UN-RES corpus by crawling the United Nations Official Document System[2] us-

[2] https://documents.un.org/

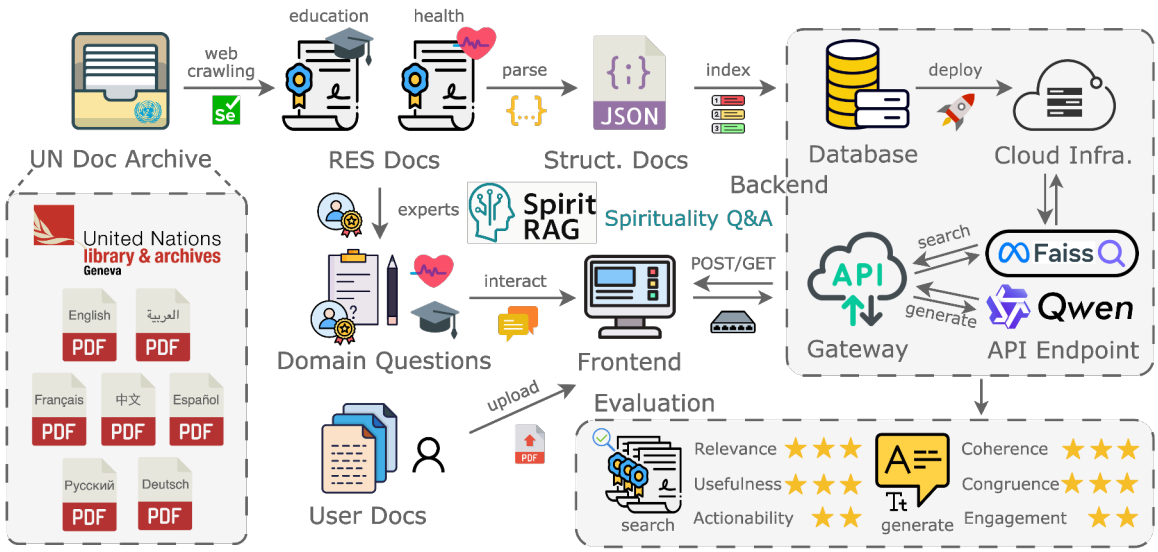


Figure 2: Architecture of SpiritRAG. SpiritRAG is an interactive Question Answering (Q&A) system enhanced by Retrieval-Augmented Generation (RAG) and specifically designed for acquiring domain-dependent knowledge in the resolution documents (RES Docs) of United Nations (UN) with a focus on Religion and Spirituality (R/S) related to health and education.

ing a Python script with the web browsing simulator `Selenium`[3]. Specifically, we searched for UN documents (PDF files) that are: (1) resolution documents (marked RES); (2) related to the subjects *health* and *education*; and (3) published between *01.01.1990* and *03.31.2025*. This process resulted in a total of nearly 7,500 UN resolution documents across seven languages: *Arabic* (ar), *Chinese* (zh), *English* (en), *French* (fr), *German* (de), *Russian* (ru), and *Spanish* (es). The overall dataset statistics are presented in Table 1. Figure 3 visualizes the temporal distribution of UN resolution documents within the health and education domains, along with the corresponding counts of unique subjects.

Freq.	ar	de	en	es	fr	ru	zh
health + R/S							
# Doc.	4,669	2,328	4,781,	4,670	4,747	4,690	4,759
# Subj.	2,360	2,046	2,383	2,361	2,377	2,367	2,380
education							
# Doc.	2,586	1,123	2,718	2,658	2,694	2,670	2,586
# Subj.	1,865	1,455	1,944	1,919	1,935	1,930	1,938

Table 1: Frequency of documents and subjects per language for each subject domain.

To provide an initial overview of the data, we conducted a hierarchical cluster analysis of the 2,652 unique subject tags included in the metadata of the UN-RES corpus. The resulting 185 concise and interpretable clusters allowed for deeper in-

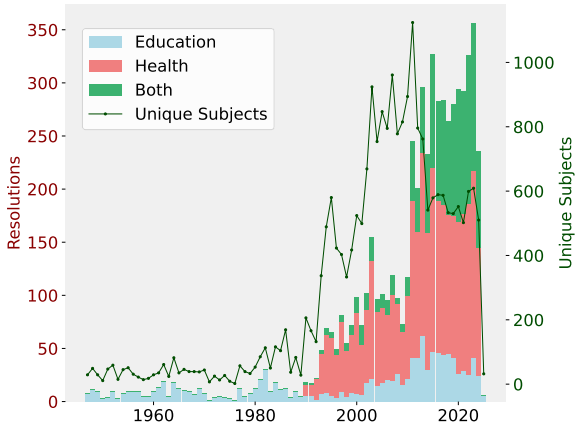


Figure 3: Resolutions and subjects across domains and decades in the UN-RES dataset.

sights into the temporal distribution of religious themes within the contexts of health and education. Figure 7 in Appendix D shows the average weight fluctuations of selected relevant clusters across decades. Figure 8 presents a more fine-grained analysis of specific religion-related subjects, broken down by five-year periods.

We further parsed crawled documents in English, French, German, and Spanish into a structured JSON format with paragraph chunks using the document parsing toolkit `Docling`[4] (Deep Search Team, 2024)[5]. From the parsed documents, we con-

[3] https://github.com/seleniumhq/selenium, Apache-2.0 license.

[4] https://github.com/docling-project/docling, MIT license.

[5] We did not parse other languages due to the lack of high-

structed the vector database for UN-RES with the efficient nearest neighbor search toolkit `Faiss`[6], by using the pre-trained text encoder Sentence-BERT (Reimers and Gurevych, 2019) and Qwen3-Embedding (Zhang et al., 2025) to pre-compute the document embeddings.

2.2 Models

Document retrieval. Given a user query q (i.e., a R/S-related question), we use the same text encoder to compute the query embedding $\mathbf{e}_q$ on the fly. The document retriever first computes the cosine similarity score between the query embedding $\mathbf{e}_q$ to each pre-computed document embedding $\mathbf{e}_d \in \mathbf{E}^{|D| \times m}$ (where m denotes the embedding dimensionality, D the corpus, and d the document in D) stored in the database:

$$\text{cos\_sim}(\mathbf{e}_q, \mathbf{e}_d) = \frac{\mathbf{e}_q \cdot \mathbf{e}_d}{||\mathbf{e}_q|| \cdot ||\mathbf{e}_d||}.$$

Based on the resulting similarity scores, the retriever pre-fetches the top-n most relevant resolution documents $(D_i)_{i=1}^{n}$ by ranking them in descending order of cosine similarity (i.e., the higher the similarity, the higher the rank):

$$\text{rank}(d) = |\{d' \in D \mid \text{cos\_sim}(\mathbf{e}_q, \mathbf{e}_{d'}) > \text{cos\_sim}(\mathbf{e}_q, \mathbf{e}_d)\}| + 1.$$

The document retriever then re-ranks the documents in $(D_i)_{i=1}^{n}$ by calculating the similarity between the query embedding and each sentence embedding $\mathbf{e}_s$ within a given D_i, using both the maximal and average similarities. Each D_i is re-ranked according to the relevance score

$$r(D_i; q) = \alpha \cdot \max_{s \in D_i}(L_2(\mathbf{e}_q, \mathbf{e}_s)) + (1 - \alpha) \cdot \operatorname*{avg}_{s \in D_i}(L_2(\mathbf{e}_q, \mathbf{e}_s)),$$

where s is a sentence in D_i, L_2 denotes the Euclidean distance and α is a weighting factor that was empirically set to 0.7. The final retrieved documents are the top-k entries in the re-ranked list.

Answer generation. Conditioned on the user query and the top-k most relevant documents retrieved from the database, we use instruction-tuned Qwen3 models (Yang et al., 2025) to generate the answer by assembling the inputs with a simple prompt template (see Figure 4) and passing them to the model.

quality PDF parsers.

[6] https://github.com/facebookresearch/faiss, MIT license.

> You are a helpful AI assistant. Use the following information to answer the user's question.
>
> User's question: {query}
>
> Relevant information from the retrieved documents: {retrieved_docs}
>
> Relevant information from the user uploaded PDF (optional): {parsed_pdf}

Figure 4: Prompt template used for answer generation.

In practice, users often have reliable source documents that may provide additional context for their questions. To support this, we implemented a function that allows users to upload a source PDF file, which is parsed and chunked in real time. When available, this source file is also included in the conditional generation of the answer.

To accelerate the generation process, we implemented the answer generation with the efficient inference toolkit `vLLM`[7] (Kwon et al., 2023).

2.3 Demonstration

SpiritRAG has a front-end implemented with `React.js`[8] and a back-end engineered using `Flask`[9] APIs. With this modular design, SpiritRAG is lightweight and easily deployed both locally or on cloud services, requiring minimal computation resources. Our pilot system was deployed on a university-internal cloud service using an NVIDIA Tesla L4 GPU card with 24 GB of RAM.

Figure 5 demonstrates the user interface of SpiritRAG. SpiritRAG allows real-time human-machine interactions in a chatbot setting, where users can freely ask questions related to R/S. The answers generated by the Qwen3 model are rendered in Markdown for improved readability. Users can switch to dark mode and download the conversation history by clicking on the buttons in the banner.

On the right panel of the web application, accessible via the source button in the lower right corner, the most relevant documents used for answer generation are displayed (see Figure 9a in

[7] https://github.com/vllm-project/vllm, Apache-2.0 license.

[8] https://react.dev/

[9] https://flask.palletsprojects.com/

Figure 5: Overview of SpiritRAG. Generated responses are rendered in Markdown to improve readability of long answers. This example answer was generated by `Qwen3-1.7B` with `Qwen3-Embedding-0.6B` as document retriever.

Appendix E). Each retrieved document is presented with its metadata, including document title, key subjects (highlighted in green), publication date, and available languages. We report more statistics on the subjects in Appendix D.

When one of the available languages is clicked, a PDF viewer will be opened in the left panel, displaying the original UN resolution document (see Figure 9b in Appendix E). The left panel can be closed by clicking on any empty area on the web application.

3 Evaluation

We performed human evaluation of SpiritRAG's document retrieval and answer generation capabilities. For this purpose, we asked two domain experts specialized in R/S and education research to assess SpiritRAG along multiple evaluation dimensions.

For each question in the test of 100 questions (see Appendix A), the domain experts were first asked to evaluate the retrieved documents. After an answer was generated, experts could activate the assessment mode by clicking the Eval button located at the top right of the right panel. For each retrieved document and the generated answer, a five-point Likert scale was provided to assess six evaluation dimensions (see Appendix A for details).

4 Results and Discussion

In Table 2 and Table 3, we report the average ratings across each evaluation dimension. We observe that the quality of both retrieved documents and generated answers improves as the backbone document retriever and answer generator are replaced with larger language models. Evaluators noted the considerable improvement in performance between Sentence-BERT and Qwen3-Embedding as document retriever, and particularly a significant and consistent improvement in answer length and detail of `Qwen3-0.6B` in comparison with `Qwen3-1.7B` as answer generator. It is worth noting that, in the case of document retrieval, a low actionability score does not necessarily indicate poor retriever performance. Rather, it reflects the fact that only a limited number of UN resolutions contain actionable directives, and ambivalence regarding how actionability should be understood in the context of several evaluation questions.

A more detailed, iterative strategy of evaluation and fine-tuning based on systematically developed evaluation dimensions tailored to the corpus was suggested to further strengthen the feasibility of SpiritRAG as a widely used research tool. An informal pilot in an interdisciplinary academic set-

Document Retriever	Relevance	Accuracy	Usefulness	Temporality	Actionability
all-MiniLM-L6-v2	3.78	4.45	3.66	4.45	2.57
Qwen3-Embedding-0.6B	4.26	4.99	3.89	4.99	2.08
Answer Generator	**Congruence**	**Coherence**	**Relevance**	**Creativity**	**Engagement**
Qwen3-0.6B	2.27	3.29	2.44	1.94	2.06
Qwen3-1.7B	3.41	3.94	4.18	3.39	2.51

Table 2: Human evaluation results on health + R/S related test questions.

Document Retriever	Relevance	Accuracy	Usefulness	Temporality	Actionability
all-MiniLM-L6-v2	2.56	2.57	2.44	4.70	2.38
Qwen3-Embedding-0.6B	3.51	3.43	3.42	4.75	3.39
Answer Generator	**Congruence**	**Coherence**	**Relevance**	**Creativity**	**Engagement**
Qwen3-0.6B	2.42	2.71	2.23	3.54	2.71
Qwen3-1.7B	3.78	3.76	3.98	3.92	4.06

Table 3: Human evaluation results on education related test questions.

ting suggested that SpiritRAG is perceived by subject experts as a highly innovative, interdisciplinary proof-of-concept with the potential for far-reaching impact in their respective academic disciplines.

5 Related Works

SpiritRAG is relevant to scholars in religious studies, health policy, and international relations. It is of particular interest to researchers interested in methodological innovation in the study of religion (Wright, 2022), the use of digital humanities methods in the study of religion (Hutchings, 2015) and the use of neural networks for text analysis (Kim et al., 2020; Suissa et al., 2022). SpiritRAG also provides a workable proof-of-concept for theologians interested in exploring new forms of dialogue-based interaction with religious texts (Kurlberg, 2020; Garner, 2021; Sutinen and Cooper, 2021). SpiritRAG complements computational analyses of large text corpora in fields that have mostly focused on topic modeling to date (Park et al., 2023; Yamada, 2024) and proposes a new resource for scholars looking at inter-textual discursive relationships in global governance and social policy (Windzio and Heiberger, 2022; Montjourides, 2023).

6 Conclusion and Future Work

We presented SpiritRAG, a Q&A system enhanced with RAG for the domains of health plus R/S within the context of United Nations resolution documents. Built on top of approx. 7,500 UN resolution documents, SpiritRAG is designed to support social scientists and policymakers in the initial stages of knowledge acquisition by enabling interactive retrieval of contextualized, domain-dependent information. This, in turn, eases evidence-informed research and decision-making. In future work, we plan to expand SpiritRAG to incorporate a broader range of document sources, including religious texts, with the goal of making the embedded spiritual knowledge in these documents more accessible and relevant to contemporary discourse.

Limitations

SpiritRAG has several limitations. First, due to computational constraints, we did not evaluate larger-scale LLMs for document retrieval and answer generation. Second, in the absence of ground-truth answers and relevant documents, we did not fine-tune the document retriever for optimal performance. Third, SpiritRAG is currently tailored to address education- or health + R/S-related questions within the context of UN resolution documents only; it does not incorporate documentation from other UN organizations and related multilateral organizations. Nevertheless, thanks to its modular architecture, SpiritRAG can be readily extended to other domain-specific knowledge sources.

Acknowledgments

This work is funded by the URPP Digital Religion(s) at the University of Zurich (UZH) and the Swiss National Science Foundation, grant number 204286, with extensive support provided by the Chair of History of Education and Education Policy Analysis at Institute of Education of UZH. We also acknowledge NCCR Evolving Language, Swiss National Science Foundation Agreement No. 51NF40_180888. A special thanks to Mr. Igor Mustač at LiRI for deploying SpiritRAG online.

References

Wendy Ager, Michael French, Atallah Fitzgibbon, and Alastair Ager. 2019. The Case for—and Challenges of—faith-sensitive Psychosocial Programming. *Intervention*, 17(1):69–75.

Claudia Baumgart-Ochse and Klaus Dieter Wolf, editors. 2019. *Religious NGOs at the United Nations: Polarizers or Mediators?* Routledge studies in religion and politics. Routledge, Taylor & Francis Group, London.

Jeremy R. Carrette. 2017. *Religion, NGOs and the United Nations: Visible and Invisible Actors in Power*. Religious studies. Bloomsbury Academic, London.

Deep Search Team. 2024. Docling Technical Report.

Yingqiang Gao, Jhony Prada, Nianlong Gu, Jessica Lam, and Richard HR Hahnloser. 2024. MODOC: A Modular Interface for Flexible Interlinking of Text Retrieval and Text Generation Functions. *arXiv preprint arXiv:2408.14623*.

Stephen Garner. 2021. Theology and the New Media. In *Digital religion*, pages 266–281. Routledge.

Tim Hutchings. 2015. Digital Humanities and the Study of Religion. In Patrik Svensson and David Theo Goldberg, editors, *Between Humanities and the Digital*, pages 283–294. The MIT Press, Massachusets.

Ben Hutchinson. 2024. Modeling the Sacred: Considerations when Using Religious Texts in Natural Language Processing. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1029–1043.

Ariba Khan, Stephen Casper, and Dylan Hadfield-Menell. 2025. Randomness, not Representation: The Unreliability of Evaluating Cultural Alignment in LLMs. *arXiv preprint arXiv:2503.08688*.

Seong-Hyeon Kim, Narae Lee, and Pamela Ebstyne King. 2020. Dimensions of Religion and Spirituality: A Longitudinal Topic Modeling Approach. *Journal for the scientific study of religion*, 59(1):62–83.

Jonas Kurlberg. 2020. Doing God in Digital Culture: How Digitality is Shaping Theology. *Cursor_ Zeitschrift für explorative Theologie*. Https://cursor.pubpub.org/pub/sk6clz8d.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in neural information processing systems*, 33:9459–9474.

Songyuan Liu, Ziyang Zhang, Runze Yan, Wei Wu, Carl Yang, and Jiaying Lu. 2024. Measuring Spiritual Values and Bias of Large Language Models. *arXiv preprint arXiv:2410.11647*.

Reem Masoud, Ziquan Liu, Martin Ferianc, Philip C Treleaven, and Miguel Rodrigues Rodrigues. 2025. Cultural Alignment in Large Language Models: An Explanatory Analysis Based on Hofstede's Cultural Dimensions. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 8474–8503.

Patrick Montjourides. 2023. *Is this the Future We Want? Understanding the Legitimacy of International Education Agendas. The Example of Equity in Education*. Ph.D. thesis, Apollo - University of Cambridge Repository. DOI: 10.17863/CAM.89624.

Kyung Hee Park, Qi Liu, and He Li. 2023. UNESCO Practices under the SDG4 and COVID-19 Response Framework: Topic Modeling from 2003 to 2021. *International Journal of Innovative Research and Scientific Studies*, 6(2):322–329.

Simon Peng-Keller, Fabian Winiger, and Raphael Rauch. 2022. *The Spirit of Global Health: The World Health Organization and the "Spiritual Dimension" of Health, 1946-2021*. Oxford University Press, Oxford, New York.

Flor Plaza-del Arco, Amanda Curry, Susanna Paoli, Alba Cercas Curry, and Dirk Hovy. 2024. Divine LLaMAs: Bias, Stereotypes, Stigmatization, and Emotion Representation of Religion in Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4346–4366.

Yao Qu and Jue Wang. 2024. Performance and Biases of Large Language Models in Public Opinion Simulation. *Humanities and Social Sciences Communications*, 11(1):1–13.

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.

Brian Steensland, Jaime Kucinskas, and Anna Sun. 2022. *Eminently Social Spirituality*. Oxford University Press New York.

Sherrie M. Steiner and James T. Christie, editors. 2021. *Religious Soft Diplomacy and the United Nations: Religious Engagement as Loyal Opposition*. Lexington Books, Lanham.

Anne Stensvold. 2017. *Religion, State and the United Nations: Value Politics*. Routledge Studies in Religion and Politics. Taylor and Francis, London and New York.

Omri Suissa, Avshalom Elmalech, and Maayan Zhitomirsky-Geffet. 2022. Text Analysis Using Deep Neural Networks in Digital Humanities and Information Science. *Journal of the Association for Information Science and Technology*, 73(2):268–287.

Erkki Sutinen and Anthony-Paul Cooper. 2021. What is Digital Theology? In *Digital Theology: A Computer Science Perspective*, pages 13–32. Emerald Publishing Limited.

UN Inter-Agency Task Force on Religion and Development. 2020. Engaging with Religion and Faith-based Actors on the 2030 Sustainable Development Agenda - 2020. Technical report.

United Nations. 2024. The Sustainable Development Goals Report 2024.

Leah von der Heyde, Anna-Carolina Haensch, and Alexander Wenz. 2024. United in Diversity? Contextual Biases in LLM-Based Predictions of the 2024 European Parliament Elections. *arXiv preprint arXiv:2409.09045*.

Michael Windzio and Raphael Heiberger. 2022. *Talking About Education: How Topics Vary Between International Organizations*, pages 239–266. Springer International Publishing, Cham.

Fabian Winiger, Gerold Schneider, Janis Goldzycher, David Neuhold, and Simon Peng-Keller. 2025. The 'Spiritual'and the 'Religious' in the Twittersphere: A Topic Model and Semantic Map. *Journal of Religion, Media and Digital Culture*, 14(1):1–22.

Bradley R. E. Wright. 2022. Methodological Innovations for the Study of Spirituality. In *Situating Spirituality - Context, Practice, and Power*, pages 113–127. Oxford University Press.

Shoko Yamada. 2024. The Synchronic and Diachronic Evolution of Key Themes around SDG 4 before and after 2015: From a Quantitative Analysis of Web-downloaded Texts. *International Review of Education*, 70(4):651–671.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models. *arXiv preprint arXiv:2506.05176*.

A Test Set Questions and Evaluation Dimensions

ID	Question	Type	TB?
1	How has the relationship between religion, spirituality, and health been framed in policy texts adopted since 1946?	Open-ended	Yes
2	What references are made to gender equality in relation to religion, spirituality across UN resolutions?	Open-ended	No
3	Which resolutions cite religion and/or spirituality?	List (finite)	No
4	Have any UN resolutions addressed the right to religious freedom?	Yes/No	No
5	What provisions have been made for refugees' religious freedom?	Open-ended*	No
6	How is the phrase "culturally appropriate" used across UN documents?	Open-ended*	No
7	Do UN documents address the notion of human dignity?	Yes/No	No
8	What emphasis is placed on the notion of human dignity?	Open-ended	No
9	When was United Nations Interagency Task Force on Religion and Sustainable Development incorporated in UN resolutions?	Closed	Yes
10	What UN Member States commitments have been made toward religious freedom?	Open-ended*	No
11	How often is religious freedom mentioned in UN resolutions since 1990?	Closed	No
12	How has UN language associated with religion and/or spirituality evolved over time?	Open-ended*	Yes
13	What international monitoring frameworks are proposed for religious freedom-related targets?	List (finite)	Yes
14	How is religious freedom represented in UN resolutions?	Open-ended	Yes
15	Have there been calls for interreligious or interfaith dialogue in UN texts and resolution?	Yes/No	No
16	Are there any specific resolutions that discuss religion or spirituality?	List (finite)	No
17	How does the UN address religion and/or spirituality in conflict-affected regions?	Open-ended	Yes
18	Which resolutions link religion and/or spirituality to UN activities?	List (finite)	No
19	Are there references to faith-based organizations in UN resolutions?	Yes/No	No
20	How have UN texts and resolutions evolved in cultural appropriateness?	Open-ended*	Yes
21	How have religious leaders, communities or actors contributed to the Sustainable Development Goals?	Open-ended*	No
22	Does any UN text or resolution cite cultural appropriateness as a priority?	Yes/No	No
23	Is there any reference to religious involvement in UN-related activities in recent UN texts and resolutions?	Yes/No	Yes
24	Is interfaith or interreligious dialogue addressed by the United Nations?	Yes/No	No
25	What resolutions discuss interfaith or interreligious education?	List (finite)	No
26	Have any resolutions referred to culturally appropriate policies?	Yes/No	No
27	Is health framed as a human right consistently?	Open-ended	No
28	What are the main dimensions of religious discrimination acknowledged in UN resolutions?	List (finite)	Yes
29	Does any UN organizations take a specific interest in religion and/or spirituality?	Yes/No	No
30	How well do UN texts and resolutions address issues pertaining to public expenditure on healthcare?	Open-ended*	No
31	How do UN resolution approach healthcare for persons in low- or middle income countries?	Open-ended*	No
32	What frameworks have been proposed over time to monitor equity in healthcare?	List (finite)	No

Table 4: Health + R/S test questions (part 1).

ID	Question	Type	TB?
33	How often has "universal primary healthcare" as a global or priority been emphasized in the past decade?	Open-ended	Yes
34	Is "Health for All" referenced in UN resolutions?	Yes/No	No
35	What kind of healthcare system reforms are promoted by United Nations organizations?	Open-ended*	No
36	What role is assigned to UN agencies like WHO in healthcare?	Open-ended*	No
37	How is health in emergencies addressed by the United Nations?	Open-ended	Yes
38	Do any resolutions define global priorities for healthcare? And if so, how are they defined?	Yes/No + Open-ended	No
39	What provisions exist for global health emergencies in UN official texts?	Open-ended	No
40	Are health disparities based on socioeconomic status addressed by the UN?	Yes/No	No
41	How have the United Nations addressed racial or ethnic discrimination in healthcare over time?	Open-ended*	No
42	Have the Untied Nations paid attention to healthcare in rural areas?	Yes/No	No
43	To which extent do UN resolutions support healthcare education?	Open-ended	No
44	How have the UN embedded healthcare in their mandate to foster peaceful societies?	Open-ended	No
45	What resolutions address data collection in healthcare systems?	List (finite)	No
46	Are there mentions of global pandemics in health?	Yes/No	No
47	How do resolutions define the right to health?	Open-ended	Yes
48	What role does healthcare play in sustainable development according to the UN?	Open-ended	Yes
49	Are gender-based discrimination in healthcare among the concerns voiced by UN Member States?	Yes/No	No
50	What resolutions mention health as a human right?	List (finite)	No

Table 5: Health + R/S test questions (part 2).

ID	Question	Type	TB?
1	How has the right to education been framed in UN texts adopted since 1946?	Open-ended	Yes
2	What references are made to gender equality in education across UN resolutions?	Open-ended	No
3	Which resolutions discuss access to primary education?	List (finite)	No
4	Have any UN resolutions addressed the digital divide in education?	Yes/No	No
5	What provisions have been made for refugees' education?	Open-ended*	No
6	How is inclusive education defined in UN resolutions?	Open-ended*	No
7	Do UN resolutions address special needs education?	Yes/No	No
8	What emphasis is placed on technical and vocational education in UN documents between 1960 and 1970?	Open-ended	Yes
9	When was lifelong learning incorporated in UN resolutions?	Closed	No
10	What UN Member States commitments have been made toward teacher training and professional development?	Open-ended*	No
11	How often is early childhood education mentioned in UN resolutions since 1981?	Closed	Yes
12	How has the UN language associated with justice and fairness in education evolved over time?	Open-ended*	No
13	What international monitoring frameworks are proposed for education-related targets?	List (finite)	Yes
14	How are indigenous peoples' education rights represented in UN resolutions?	Open-ended	Yes
15	Have there been calls for universal free education in UN texts and resolution?	Yes/No	No
16	Are there any specific resolutions that define quality education?	List (finite)	No
17	How does the UN address education in conflict-affected regions?	Open-ended	Yes

Table 6: Education-related test questions (part 1).

ID	Question	Type	TB?
18	Which resolutions link education and peace building?	List (finite)	No
19	Are there references to education financing in UN resolutions?	Yes/No	No
20	How have UN texts and resolutions evolved in support for girls' education?	Open-ended*	No
21	How has the SDG 4 agenda been further reflected in resolutions since 2015?	Open-ended*	Yes
22	Does any UN text or resolution cite learning outcomes as a priority?	Yes/No	No
23	Is there any reference to private sector involvement in education in recent UN texts and resolutions?	Yes/No	Yes
24	Are online and distance learning modalities addressed by the United Nations?	Yes/No	No
25	What resolutions discuss multilingual education?	List (finite)	No
26	Have any resolutions referred to climate change education?	Yes/No	No
27	Is education framed as a human right consistently?	Open-ended	No
28	What are the main dimensions of exclusion acknowledged in UN resolutions during the Cold War?	List (finite)	Yes
29	Does UNESCO recommend specific pupils-teacher ratios?	Yes/No	No
30	How well do UN texts and resolutions address issues pertaining to public expenditure on education?	Open-ended*	No
31	How do UN resolutions approach education for persons with disabilities?	Open-ended*	No
32	What are the frameworks proposed in UN resolution to monitor progress towards equity in education?	List (finite)	No
33	How often has literacy as a goal or priority been emphasized in the past decade?	Open-ended	Yes
34	Is citizenship or civic education referenced in UN resolutions?	Yes/No	No
35	What kind of education system reforms are promoted by the United Nations?	Open-ended*	No
36	What role is assigned to UN agencies like UNESCO in education?	Open-ended*	No
37	How is education in emergencies addressed by the United Nations?	Open-ended	Yes
38	Do any resolutions define global citizenship education? And if so, how is it defined?	Yes/No + Open-ended	No
39	What provisions exist for non-formal education in UN official texts?	Open-ended	No
40	Are educational disparities based on socioeconomic status addressed by the UN?	Yes/No	No
41	How have the United Nations addressed racial or ethnic discrimination in education over time?	Open-ended*	No
42	Have the United Nations paid attention to education in rural areas?	Yes/No	No
43	To which extent do UN resolutions support bilingual education?	Open-ended	No
44	How have the UN embedded education in their mandate to foster peaceful societies?	Open-ended	No
45	What resolutions address data collection in education systems?	List (finite)	No
46	Are there mentions of international assessments in education?	Yes/No	No
47	How do resolutions define access to education?	Open-ended	Yes
48	What role does education play in sustainable development according to the UN?	Open-ended	Yes
49	Are gender stereotypes in curricula among the concerns voiced by UN Member States?	Yes/No	No
50	What resolutions mention teacher rights and working conditions?	List (finite)	No

Table 7: Education-related test questions (part 2).

Type	Definition
List (finite)	Desired answers are in form of finite list with bullet points.
Yes/No	Questions that can be answered in binary form.
Closed	Questions that can be precisely answered.
Open-ended	No special format requirement for desired answers.
Open-ended*	No special format requirement for desired answers, yet diachronic changes possible.

Table 8: Definition of question types.

Dimension	Descriptions
	Document Retrieval
Relevance	How relevant is the retrieved document to the question?
Accuracy	How trustworthy is the retrieved document for answering the question?
Usefulness	How useful is the retrieved document for answering the question?
Temporality	Is the retrieved document related to the time period implied by the question?
Actionability	How many actionable insights does the retrieved document contain?
	Answer Generation
Congruence	How well does the answer align with the retrieved documents?
Coherence	How logically consistent is the answer?
Relevance	How relevant is the answer to the question?
Creativity	How original and creative is the answer?
Engagement	How engaging and interesting is the answer?

Table 9: Evaluation dimensions for the retrieved documents and generated answer.

B Dataset Construction Details

We implemented the web crawler using `Selenium` in combination with `BeautifulSoup`[10]. Boolean operators were employed to combine keyword sets for document retrieval: *educat**, *school**, and *learn** for education-related documents, and *health*, *faith*, *religi**, *spiritual**, and *belief* for documents related to health and religion/spirituality (R/S). Notably, some resolution documents are relevant to both domains; see Figure 6 for an illustrative example.

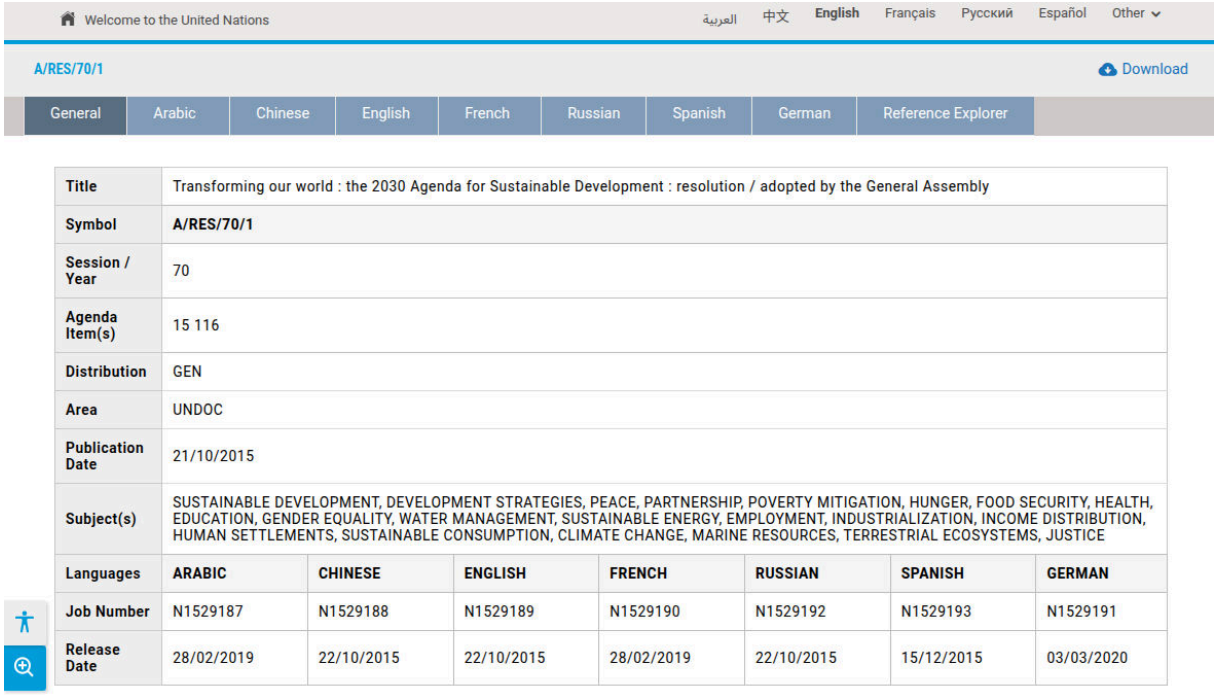


Figure 6: An example resolution document that is both health- and education-related. Screenshot taken from the website of UN Official Document System.

[10] https://www.crummy.com/software/BeautifulSoup/bs4/doc/, MIT License.

C Examples of Generated Answers

Q: What references are made to gender equality in relation to religion, spirituality across UN resolutions?
`Qwen3-Embedding-0.6B` for document retrieval and `Qwen3-1.7B` for answer generation

A: The references to gender equality in relation to religion, spirituality, and the promotion of interreligious and intercultural dialogue are embedded within several UN resolutions, particularly those addressing cultural pluralism, human rights, and the prevention of violence and discrimination. Here is a detailed breakdown of the key references:

1. **Promotion of Interreligious and Intercultural Dialogue**

Resolution 75/26 (adopted on 2 December 2020) emphasizes the importance of interreligious and intercultural dialogue for peace, mutual understanding, and social stability. It underscores the need to create an environment conducive to peace and mutual respect at both global and local levels. This resolution directly ties gender equality to religious and spiritual diversity, highlighting the role of dialogue in fostering inclusive societies.

2. **Human Rights and Religious Minorities**

Resolution 74/306 (adopted on 11 September 2020) recognizes the need to counter discrimination, hate speech, and xenophobia related to the COVID-19 pandemic. It reaffirms the primary responsibility of states to protect human rights, including the rights of religious minorities. This resolution explicitly links gender equality to the protection of religious minorities, ensuring that women and other marginalized groups are not disproportionately affected by discrimination based on religion or belief.

3. **Declaration on the Elimination of All Forms of Intolerance and Discrimination Based on Religion or Belief**

Resolution 36/55 (adopted on 25 November 1981) proclaims the **Declaration on the Elimination of All Forms of Intolerance and Discrimination Based on Religion or Belief**, which includes provisions for gender equality. The declaration emphasizes the importance of protecting the rights of individuals, including women, and promotes the eradication of discrimination based on religion or belief.

4. **International Day of Living Together in Peace**

Resolution 72/130 (adopted on 8 December 2017) designates 28 December as the International Day of Living Together in Peace. While not explicitly mentioning gender equality, this resolution underscores the importance of mutual respect and understanding, which are critical for gender equality in religious and spiritual contexts.

5. **United Nations Alliance of Civilizations**

Resolution 69/312 (adopted on 6 July 2015) reaffirms the **United Nations Alliance of Civilizations**, which promotes dialogue among civilizations, cultures, religions, and beliefs. This initiative aligns with the goal of fostering gender equality by encouraging respectful interactions between diverse groups, including religious and spiritual communities.

6. **Strategic Initiatives and Campaigns**

UNESCO's Unite for Heritage Campaign (launched in 2015) and the **Plan of Action to Safeguard Religious Sites** (prepared by the United Nations Alliance of Civilizations) highlight the importance of preserving cultural heritage and promoting tolerance. These initiatives indirectly support gender equality by ensuring that religious and spiritual practices are respected and accessible to all, including women and girls.

7. **Combating Violent Extremism and Hate Speech**

Resolution 73/285 (adopted on 2 April 2019) calls for the elimination of violence based on religion or belief and emphasizes the role of interreligious and intercultural dialogue in countering hate speech. This resolution supports gender equality by addressing systemic issues that contribute to discrimination and violence, particularly against women and girls.

8. **International Day of Commemoration of the Victims of Acts of Violence Based on Religion or Belief**

Resolution 73/296 (adopted on 28 May 2019) designates 22 August as the International Day of Commemoration of the Victims of Acts of Violence Based on Religion or Belief. This day highlights the need to address violence rooted in religious or spiritual discrimination, which disproportionately affects women and girls.

9. **Global Migration and Cultural Diversity**

Resolution 74/23 (adopted on 12 December 2019) acknowledges the growing importance of interreligious and intercultural dialogue in the context of migration. This resolution reinforces the idea that cultural diversity and the pursuit of cultural development are sources of mutual enrichment, which is essential for gender equality in a multicultural society.

10. **Role of Freedom of Expression in Combating Intolerance**

Resolution 73/296 also reiterates the importance of the **right to freedom of opinion and expression** in strengthening democracy and combating religious intolerance. This principle is central to gender equality, as it ensures that women and other marginalized groups can express their views and participate in societal discourse without fear of discrimination.

Conclusion:

The UN resolutions collectively emphasize the intersection of religion, spirituality, and gender equality through the promotion of interfaith dialogue, the protection of human rights, and the prevention of discrimination. They recognize that religious and spiritual diversity must be respected and integrated into policies that advance gender equality, social justice, and peace. By fostering mutual understanding and respect, these resolutions aim to create inclusive societies where all individuals, regardless of their religious or spiritual beliefs, can thrive.

`all-MiniLM-L6-v2` for document retrieval and `Qwen3-0.6B` for answer generation

A: The references to gender equality and religion/spirituality in UN resolutions are found in the following context:

1. **Mainstreaming a Gender Perspective**

Resolution adopted by the Economic and Social Council on 7 June 2023 (Agenda item 12).

- It reaffirms its agreement from 1997/2 and recalls previous resolutions (e.g., 2011/6, 2012/24, 2013/16, etc.).
- It emphasizes the importance of mainstreaming a gender perspective into all policies and programs.

2. **Gender Equality and the Beijing Declaration and Platform for Action**

- The UN has reiterated that gender mainstreaming is a critical strategy for achieving gender equality and empowering women and girls.
- The outcomes of the twenty-third special session and the implementation of the Programme of Action of the International Conference on Population and Development are highlighted.

3. **Global Commitments**

- The UN recognizes the importance of gender equality and empowerment through various summits and conferences, including the 2005 World Summit, 2011 World Summit, and the 2015 development agenda.

These references collectively emphasize the role of gender equality and the integration of religious/spiritual aspects into UN policies and initiatives.

Table 10: Answers generated by SpiritRAG. The average response time is dependent on the system configuration. Our best system setup takes around 7 seconds to response.

D Statistics on Domain-related Subjects in UN Resolution Documents

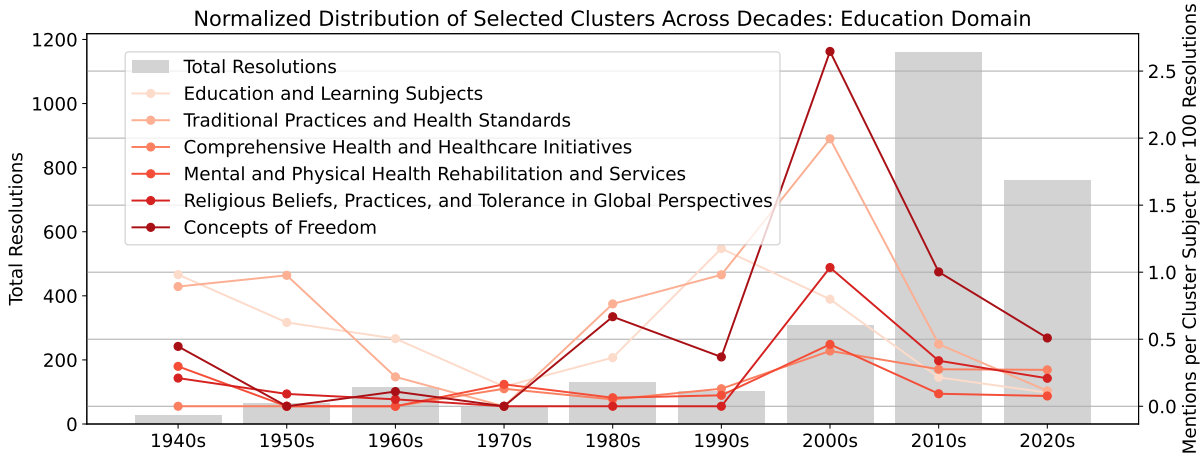


Figure 7: Temporal trends in the representation of selected subject clusters[11] within the education domain. The gray bars indicate the total number of education-related resolutions in each decade. Colored lines show the normalized frequency of mentions for each cluster, calculated as the average number of subject mentions per cluster (per 100 resolutions), adjusted for the number of subjects in each cluster.

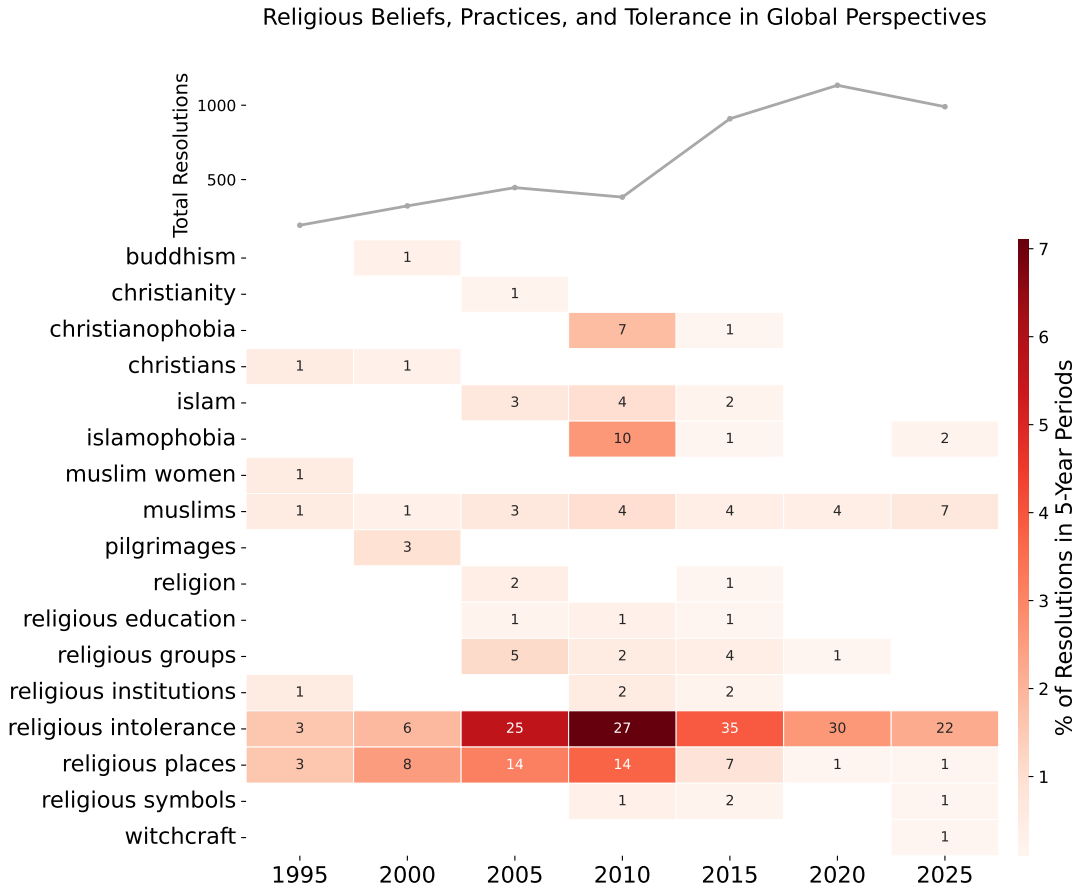
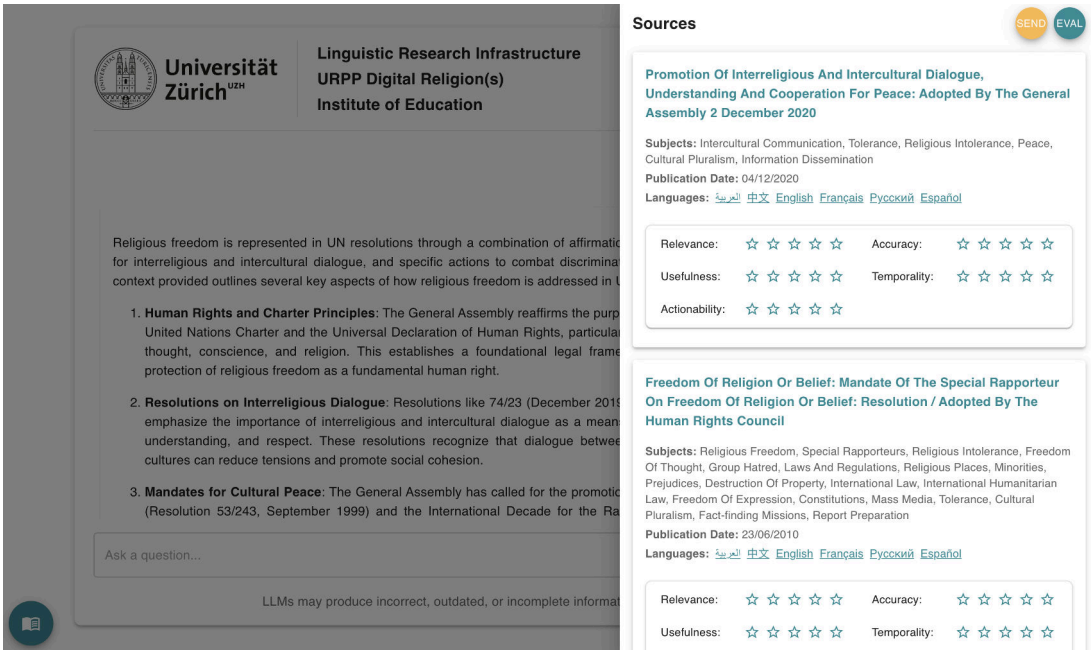


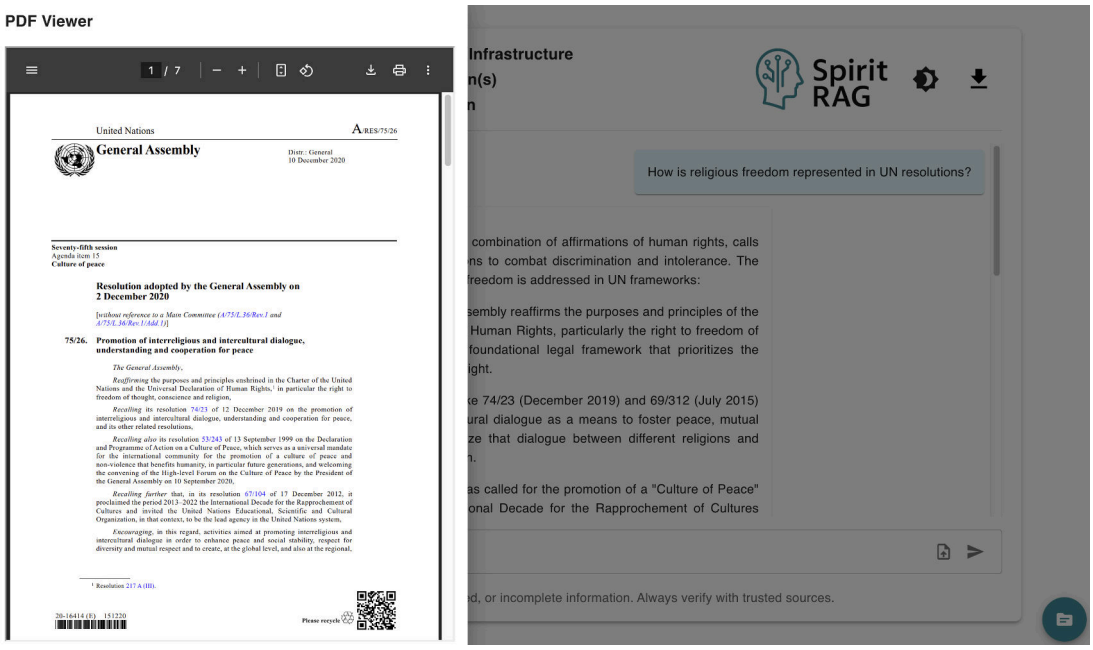
Figure 8: Distribution of the cluster #133 subjects within the health domain across 5-year periods. Cell color reflects the proportion of resolutions in each period tagged with a given subject. Raw counts of subjects are overlaid as annotations. The line chart above shows the total number of health-related resolutions in each period.

[11]The clusters in Figures 7 and 8 were created using Sentence-BERT (`all-MiniLM-L6-v2`) and agglomerative clustering from `sklearn` with the distance threshold 2. Cluster titles were assigned automatically using `GPT-4o`.

E User Interface



(a) Retrieved documents.



(b) Document viewer.

Figure 9: User interface of SpiritRAG. Left panel (viewing original PDF document) and right panel (viewing the list of the most relevant documents to the user question).

LINGCONV: An Interactive Toolkit for Controlled Paraphrase Generation with Linguistic Attribute Control

Mohamed Elgaar and **Hadi Amiri**
University of Massachusetts Lowell
{melgaar,hadi}@cs.uml.edu

Abstract

We introduce LINGCONV, an interactive toolkit for paraphrase generation enabling fine-grained control over 40 specific lexical, syntactic, and discourse linguistic attributes. Users can directly manipulate target attributes using sliders, and with automatic imputation for unspecified attributes, simplifying the control process. Our adaptive Quality Control mechanism employs iterative refinement guided by line search to precisely steer the generation towards target attributes while preserving semantic meaning, overcoming limitations associated with fixed control strengths. Applications of LINGCONV include enhancing text accessibility by adjusting complexity for different literacy levels, enabling personalized communication through style adaptation, providing a valuable tool for linguistics and NLP research, and facilitating second language learning by tailoring text complexity. The system is available at https://mohdelgaar-lingconv.hf.space, with a demo video at https://youtu.be/wRBJEJ6EALQ.

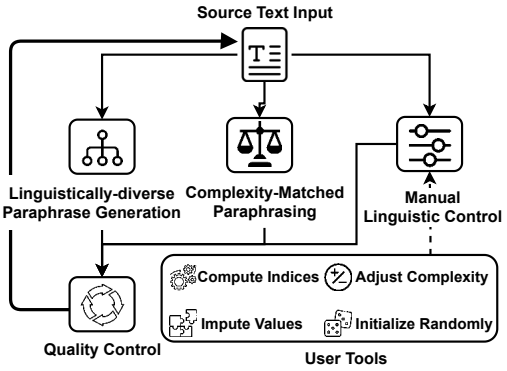


Figure 1: System architecture of LINGCONV. The system provides three modes of operation: Linguistically-diverse Paraphrase Generation for creating multiple diverse outputs, Complexity-Matched Paraphrasing for style transfer, and Manual Linguistic Control for fine-grained attribute adjustment. All modes utilize the Quality Control mechanism to ensure output quality. The set of User Tools support the manual specification of linguistic attributes.

1 Introduction

Controllable text generation, the task of producing text conforming to specified attributes like sentiment or formality (Jin et al., 2022), has seen widespread application in areas such as text simplification (Lee and Lee, 2023a,b; Vajjala and Lučić, 2018; Zhang and Lapata, 2017; Xu et al., 2015), toxicity control (Zheng et al., 2023; Zhang and Song, 2022; Liu et al., 2021), and personalized dialogue (Huang et al., 2023b; Niu and Bansal, 2018). While general large language models (LLMs) can be prompted to modify text style, achieving reliable, fine-grained, and verifiable control over multiple specific linguistic properties simultaneously remains a challenge (Shi et al., 2024).

We address this gap by introducing LINGCONV, an interactive toolkit specifically designed for controlled paraphrase generation (CPG) with explicit, fine-grained manipulation of 40 distinct linguistic attributes spanning lexical, syntactic, and discourse dimensions. LINGCONV allows users to generate paraphrases of a source text that precisely match a target linguistic style. This capability offers significant utility: for accessibility, text can be simplified or complexified for different reading levels; for personalization, communication can be tailored to specific user styles; for linguistics/NLP research, the system provides a platform for systematically studying the effects of linguistic variations; and for education, text complexity can be adjusted for second language learners.

The system is based on the CPG model described in Elgaar and Amiri (2025), which builds upon a T5 encoder-decoder model (Raffel et al., 2020), integrating the target linguistic attribute vector di-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 42–51
November 4-9, 2025

rectly into the decoding process. Moreover, its adaptive Quality Control (QC) mechanism (§ 2.2) allows the system to iteratively refine the generation, matching target linguistic attributes while preserving semantic meaning, even when target attributes differ significantly from the source or involve complex transformations. Our QC approach employs gradient-based updates and an adaptive line search to dynamically adjust control strength.

The system architecture (Figure 1) supports three distinct modes of operation for different user needs: linguistically-diverse paraphrase generation, complexity-matched paraphrasing by example, and manual slider-based control for precise adjustments. Furthermore, the system provides tools for facilitating the manual specification of linguistic attributes (§ 3).

The system architecture (Figure 1) supports multiple modes of operation and includes tools to facilitate attribute specification (§ 3). As linguistic complexity attributes, we consider lexical, syntactic, and discourse psycholinguistic indices (Appendix A).

While some prior work focused on controlling syntactic structure through manipulations of parse trees or AMR graphs (Huang et al., 2023a; Goyal and Durrett, 2020; Iyyer et al., 2018), LINGCONV provides control over a set of 40 lexical, syntactic, and discourse attributes within an interactive framework with an adaptive QC mechanism for precision and robustness.

LINGCONV offers a range of advanced features and functionalities to provide users with greater control and flexibility over the generation process. Users can choose from three distinct paraphrase generation strategies: *Randomized Paraphrase Generation*, *Complexity-Matched Paraphrasing*, and *Manual Linguistic Control*. Additionally, the system allows users to select between exact or approximate linguistic index computation, which is used in interpolation and the manual setting of linguistic attributes. The system provides the option to show the intermediate sentences generated during the quality control process, enhancing transparency and interpretability.

2 System Architecture

2.1 Model

LINGCONV employs a T5-Base (Raffel et al., 2020) encoder-decoder model augmented with a linguistic complexity control approach to perform complexity-controlled paraphrase generation. Given a dataset $\mathcal{D} = \{s, t\}$ of paraphrase source and target pairs s and t, we compute the linguistic attributes of the target l^t. Thus, the task is to find a mapping from $(s, l^t) \rightarrow t$.

First, LINGCONV employs a linguistic embedding layer $h(l) = \mathbb{R}^k \rightarrow \mathbb{R}^{d_{model}}$, where k is the number of linguistic attributes (represented as a vector), and d_{model} is the input embedding dimension of T5. The embeddings for k linguistic attributes are learned jointly with the encoder-decoder model's parameters. Linguistic attributes are injected into the decoding process through element-wise addition to the embeddings of the first token of the decoder. The decoder attends to the linguistic embeddings at each generation step using self-attention (Vaswani et al., 2017). Through the training process, the decoder learns to steer the generation towards the desired target attributes. The model is trained using cross-entropy loss of the translation from source to target paraphrases.

2.2 Quality Control

In controlled text generation, achieving precise control over multiple linguistic attributes while maintaining text quality presents significant challenges. Not all combinations of linguistic attributes are feasible (such as having more unique words than total words), and even valid combinations may be difficult for the model to achieve in a single generation step.

To address these challenges, LINGCONV employs an iterative refinement process. Starting with an initial generation, the system gradually adjusts the output to better match the target linguistic attributes while preserving semantic meaning.

This refinement is achieved by computing the gradient of the linguistic attribute error with respect to the input embeddings. However, determining the appropriate update strength (steering factor) is critical: too small is ineffective, too large degrades quality. Fixed control strengths, explored in prior work (Durmus et al., 2024; Yang et al., 2024), are insufficient here because the norm of our gradient-based steering vector varies significantly with the attribute discrepancy.

LINGCONV addresses this through an adaptive control strength approach that uses line search to dynamically determine the optimal step size for each refinement iteration. The quality control mechanism consists of two key components: 1) An iterative refinement process that repeatedly updates

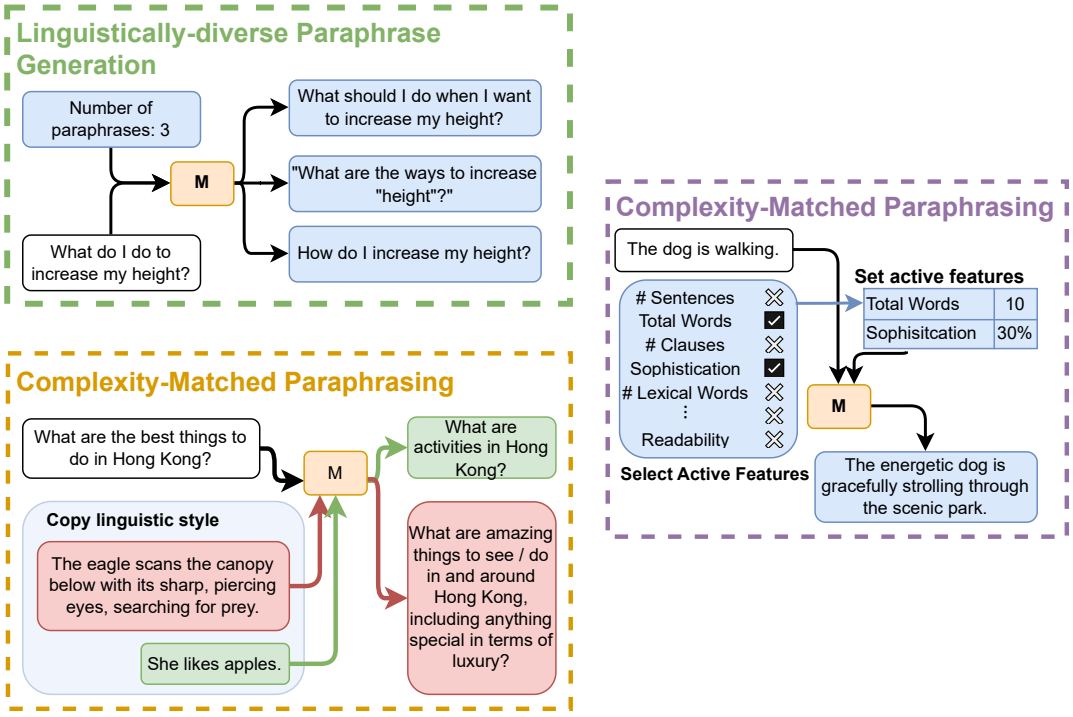


Figure 2: Examples of the three paraphrase generation modes of LINGCONV.

the generation until it matches the target attributes or further improvement becomes impossible. 2) A line search algorithm that finds the optimal control strength for each refinement step while preserving semantic coherence

The quality control algorithm is illustrated in Figure 3. The iterative refinement process starts by generating an initial candidate output based on the input sentence and target attributes. Figure 4 illustrates this process with examples of intermediate outputs generated during refinement. It then enters a loop where it repeatedly refines the generation until it closely matches the target attributes, or further refinement is not possible.

The iterative process (Figure 3) starts with an initial generation and enters a refinement loop. Each iteration computes the attribute error and performs line search to find an optimal update strength (derived from the negative gradient). The process continues until the MSE between the candidate's attributes and the target falls below a threshold τ, or until line search fails to find an improvement, ensuring convergence even for challenging targets.

Backpropagation of the linguistic attribute error requires differentiable linguistic index computation. Thus, we pre-train **linguistic discriminator (LD)** and a **semantic embedding module (SEM)**.

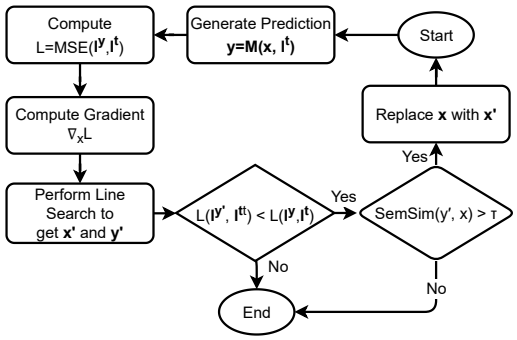


Figure 3: The quality control algorithm flowchart. The algorithm starts with initial generation and enters a refinement loop until no further improvement is possible. Each iteration computes the attribute error, performs line search to find optimal update strength, and generates a new candidate output.

The linguistic discriminator (LD) learns to predict the linguistic attributes of a sentence, providing the error signal for attribute control during QC. The semantic embedding module (SEM) is trained to predict the probability that the source sentence s and the generated paraphrase $\hat{t}$ are semantically equivalent. Their specific training objectives and architectures are detailed in Appendix C.

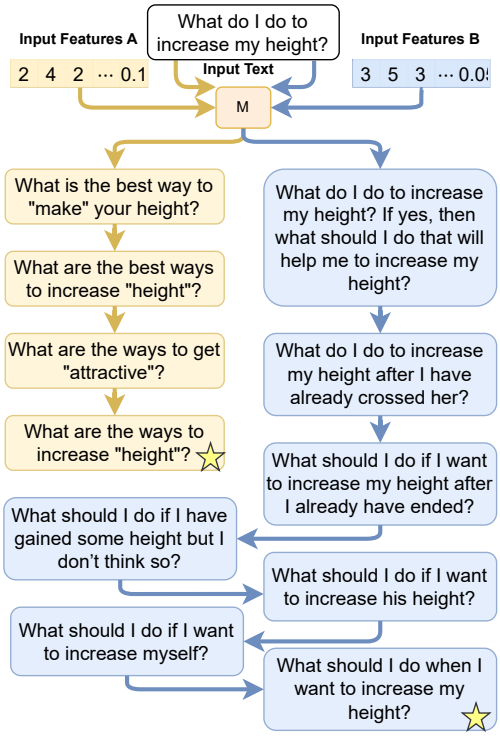


Figure 4: Two examples of intermediate texts iteratively generated at inference time using the quality control approach for text generation. Both examples are generated using the same source sentence and two different target attributes. The star indicates the returned value.

3 Features and Functionalities

As shown in Figure 2, LINGCONV offers three modes for fine-grained attribute adjustment.

The **Randomized Paraphrase Generation** mode serves users seeking diverse paraphrases; it automatically generates multiple linguistically varied paraphrases, discovering stylistic possibilities without requiring users to specify target attributes.

The **Complexity-matched Paraphrasing** mode addresses the need for textual style transfer; users provide a reference text, and the system generates a paraphrase of the source that mimics the reference's linguistic style, useful for adapting content to specific audiences or contexts. This also allows users to define the target linguistic style implicitly through an example, which can be easier than manually specifying numerous individual attributes.

The **Manual Linguistic Control** mode offers fine-grained manipulation; it allows expert users or those with specific requirements (e.g., researchers studying linguistic effects) to precisely adjust individual linguistic attributes using sliders, providing maximum control over the output.

Advanced Options allow toggling exact/approximate attribute computation and viewing intermediate QC steps. An **Examples** tab provides 150 validation set samples.

The linguistic attributes we use are extracted from three sources: lexical attributes developed by Lu (2012), syntactic attributes by Lu (2010), and a diverse set of semantic, lexical, discourse, and traditional attributes by Lee and Lee (2023a). A detailed list of the linguistic attributes used can be found in Appendix A.

The system implements error handling: Input validation for text length and content, automatic correction of invalid linguistic attribute combinations, graceful handling of model prediction failures, and user feedback for invalid operations.

3.1 Randomized Paraphrase Generation

The Paraphrase Generation feature provides a straightforward yet powerful way to generate multiple paraphrases of a given source sentence. Users begin by inputting the source sentence and indicating the desired number of paraphrases to be generated. The system then employs its linguistic attributes sampling and text generation algorithm to generate a set of distinct paraphrases, each adhering to a unique set of target linguistic attributes.

LINGCONV employs a large-scale repository of precomputed linguistic index sets. These sets, extracted from the training dataset, encompass a wide spectrum of linguistic attributes. By randomly selecting index sets from this collection, the system ensures that the generated paraphrases carry diverse linguistic characteristics. Figure 2 shows examples of paraphrases generated by LINGCONV using different generation modes.

3.2 Complexity-matched Paraphrasing

Given a source sentence and a reference sentence. The model extracts the linguistic attributes from the reference. Utilizing these extracted attributes as a guide, the system generates a paraphrase of the source sentence that mirrors the linguistic attributes of the reference. This form of textual style transfer enables users to seamlessly adapt their content to match a specific style or level of complexity. It is a valuable tool for authors, marketers, communicators, and clinicians looking to tailor their text to distinct audiences or contexts while maintaining semantic coherence. Figure 2 displays examples

of using different modes, including complexity-matched paraphrasing.

3.3 Manual Linguistic Control

The system provides manual control through sliders corresponding to specific linguistic attributes, with bounds determined through statistical analysis. The interface implements: **Attribute Activation** for selective control, **Automatic Imputation** for inactive attributes, and **Range Constraints** to prevent invalid combinations. Users can activate specific linguistic attributes of interest and adjust their values using sliders, which are constrained by minimum and maximum values to ensure valid settings.

The Manual Linguistic Control mode provides a set of tools to increase accessibility for users. Users can focus solely on activating and adjusting the specific linguistic attributes of interest. The system then automatically imputes reasonable values for all other inactive attributes (see § 3.4), alleviating the need to manually specify the entire set of attributes.

To further assist manual setting of linguistic values, the system offers a set of tools accessible through the **Tools to assist in setting linguistic attributes** interface. As illustrated in Figure 1, these tools include several key functionalities. The **Random Target** generator produces valid target linguistic indices from the training dataset, while the **Impute Missing Values** function fills in remaining attributes to maintain coherence when only a subset is specified. Users can analyze existing text through the **Computing Linguistic Attributes** tool, which calculates or estimates linguistic attributes of input sentences. The **Copying Attributes** functionality enables streamlined transfer from source to target, particularly useful for incremental adjustments. Additionally, the **Adding and Subtracting ϵ** feature serves to incrementally increase text complexity and generate controlled variations through minor perturbations.

3.4 Imputation Process

The imputation of missing linguistic attributes is performed using the Multiple Imputation by Chained Equations (MICE) algorithm (Azur et al., 2011). For each missing linguistic attribute, a ridge regression model (Golub et al., 1999) with $\alpha = 1000$ is fitted using the other variables as predictors. The missing values are then imputed based on this model's predictions. This process is repeated for up to 1000 iterations for each variable with missing data, forming a chain of equations that leads to an iterative refinement process.

MICE models are trained on 1000 diverse attribute vectors selected greedily from the training data. Missing values are mean-initialized. MICE is well-suited for linguistic attributes as it leverages inter-attribute correlations (e.g., counts, clauses/sentences) via ridge regression, preserving relationships.

Appendix B.1 shows imputation performance using standard metrics: Mean Squared Error (MSE), and Root Mean Square Error (RMSE), and Pearson correlation coefficient (ρ). MSE and RMSE quantify the average magnitude of errors between imputed and ground-truth attribute values, and ρ measures the linear relationship between imputed and true values. Accuracy improves as more attributes are provided, with the best results at 80% known attributes, though performance remains reasonable even at 20%. See Appendix B for implementation details.

4 Data and Evaluation

4.1 Data and Experimental Setup

We utilize a combination of three paraphrase and semantic similarity corpora for training and evaluation: the Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), the Semantic Textual Similarity Benchmark (STS-B) (Cer et al., 2017), and the Quora Question Pairs dataset [1]. From these datasets, we retain only the positively labeled pairs, indicating semantic equivalence, resulting in a total corpus of 140,000 sentence pairs suitable for paraphrase generation. This combined dataset is randomly partitioned into training (80%), validation (10%), and testing (10%). The same data splits are used consistently across all experiments, and only sequences up to 100 tokens were included in training.

We train and evaluate eight baselines on the task of CPG. Three baselines produce paraphrases with no attribute control, and serve to demonstrate the quality of outputs in the case of non-controlled generation. The remaining five baselines are all recent and strong CPG approaches.

We evaluate our approach against several controlled generation baselines. As control baselines, we include a direct **Copy** of the source

[1] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Model	BERTScore^F ↑	MSE(l^t)↓	MSE(l^s)↑	Overall↑
Ref	94.4	9.82	0.96	0.19
Copy	100.0	9.86	0.00	0.33
T5-FT	97.8	9.86	0.29	0.27
Llama3.1-70B	**92.8**	8.90	2.44	0.26
BOLT	90.4	7.47	1.83	0.21
Fudge	92.5	7.22	3.11	0.37
QCPG	91.4	5.61	3.25	0.41
Lingconv	92.0	3.69	4.39	0.59
Lingconv+QC	91.5	**2.89**	**6.20**	**0.71**

Table 1: Controlled generation performance across evaluation metrics. BERTScore measures the semantic similarity between the generated paraphrase and the source sentence. Mean squared error (MSE) values reflect how close the linguistic attributes of the generated paraphrase are to the target (MSE($\boldsymbol{l^t}$)↓) or source (MSE($\boldsymbol{l^s}$)↑).

sentence and the ground-truth **Reference** paraphrase from the dataset. For learned models, we compare against **T5-FT** (Raffel et al., 2020), a vanilla T5 model fine-tuned on our paraphrase datasets, and **Llama3.1-70B** (Dubey et al., 2024), an instruction-tuned language model directed to generate attribute-controlled paraphrases. We also evaluate against recent controlled generation approaches: **BOLT** (Liu et al., 2023), which learns logit biases through attribute discriminator loss minimization; **Fudge** (Yang and Klein, 2021), which performs token-level attribute control during generation; and **Quality Controlled Paraphrase Generation (QCPG)** (Bandel et al., 2022), which uses special character prefixes for attribute control. Finally, we evaluate our base **LINGCONV** model described in §2.1, as well as an enhanced version incorporating the quality control algorithm (**+QC**).

We evaluate the quality of generation using the following four automatic metrics of text generation. **BERTScore**^{F} (Zhang et al., 2020) measures the semantic similarity between the generation and the source sentence; F refers to the reference-free metric (Shen et al., 2022). **MSE**(l^t) is the error in the attributes of the generation compared to the target attributes. MSE metrics are evaluated on the normalized attribute values to equalize the scale across attributes. **MSE**(l^s) is the distance between the attributes of the generation and the source attributes. This measures the bias of CPG methods towards the style of the source sentence, and their ability to generate a paraphrase with significantly different linguistic attributes from the source. **Overall** score normalizes each of the other three metrics to the range [0,1], such that a higher value is better, and computes the average. The overall score highlights the approach with the best trade-off between semantic similarity and accurate attribute control ability.

4.2 Results

Results in Table 1 show that the attribute control of LINGCONV is 34% more accurate than the second-best baseline, while being comparable in semantic equivalence. The addition of QC results in a further 14% decrease in attribute error. We attribute this higher performance to the effective use of linguistic complexity attributes in the decoding phase of LINGCONV.

Figure 4 presents examples of intermediate text outputs generated at inference time during the interpolation process for quality control of the generated texts. These results provide a clear illustration of the step-by-step generation process that progressively moves towards generating target sentences that meet desired levels of linguistic complexity.

5 Conclusion

We developed a new text conversion system, LINGCONV, which offers comprehensive features for complexity-controlled text generation. Through the careful integration of linguistic attributes and model architecture, LINGCONV provides users with the tools to generate text that adheres to both specific and diverse linguistic complexity levels.

The system's evaluation against recent controlled generation baselines and the vanilla T5 model verifies its reliability and effectiveness. The iterative refinement process with adaptive control strength enhances the system's ability to improve the generation process, ensuring high-quality outputs that closely match the desired linguistic attributes while maintaining semantic coherence. A current limitation is the system's focus on sentence-level paraphrasing, stemming from training on sequences up to 100 tokens; extending LINGCONV to effectively handle paragraph-level or longer document processing remains future work.

References

Melissa J Azur, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. 2011. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49.

Elron Bandel, Ranit Aharonov, Michal Shmueli-Scheuer, Ilya Shnayderman, Noam Slonim, and Liat Ein-Dor. 2022. Quality controlled paraphrase generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 596–609, Dublin, Ireland. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Esin Durmus, Alex Tamkin, Jack Clark, Jerry Wei, Jonathan Marcus, Joshua Batson, Kunal Handa, Liane Lovitt, Meg Tong, Miles McCain, Oliver Rausch, Saffron Huang, Sam Bowman, Stuart Ritchie, Tom Henighan, and Deep Ganguli. 2024. Evaluating feature steering: A case study in mitigating social biases.

Mohamed Elgaar and Hadi Amiri. 2025. Linguistically-controlled paraphrase generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Suzhou, China. Association for Computational Linguistics.

Gene H Golub, Per Christian Hansen, and Dianne P O'Leary. 1999. Tikhonov regularization and total least squares. *SIAM journal on matrix analysis and applications*, 21(1):185–194.

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. *ArXiv*, abs/2005.02013.

Kuan-Hao Huang, Varun Iyer, I-Hung Hsu, Anoop Kumar, Kai-Wei Chang, and Aram Galstyan. 2023a. ParaAMR: A large-scale syntactically diverse paraphrase dataset by AMR back-translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8047–8061, Toronto, Canada. Association for Computational Linguistics.

Qiushi Huang, Yu Zhang, Tom Ko, Xubo Liu, Bo Wu, Wenwu Wang, and H Tang. 2023b. Personalized dialogue generation with persona-adaptive attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12916–12923.

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.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205.

Bruce W. Lee and Jason Lee. 2023a. LFTK: Handcrafted features in computational linguistics. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 1–19, Toronto, Canada. Association for Computational Linguistics.

Bruce W Lee and Jason Lee. 2023b. Prompt-based learning for text readability assessment. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1774–1779.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Annual Meeting of the Association for Computational Linguistics*.

Xin Liu, Muhammad Khalifa, and Lu Wang. 2023. BOLT: Fast energy-based controlled text generation with tunable biases. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 186–200, Toronto, Canada. Association for Computational Linguistics.

Xiaofei Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15:474–496. Citation Key: Lu2010.

Xiaofei Lu. 2012. The relationship of lexical richness to the quality of esl learners' oral narratives. *Source: The Modern Language Journal*, 96(2):190–208. Citation Key: Lu2012.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association for Computational Linguistics*, 6:373–389.

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.

Lingfeng Shen, Lemao Liu, Haiyun Jiang, and Shuming Shi. 2022. On the evaluation metrics for paraphrase generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3178–3190, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. 2024. Why larger language models do in-context learning differently? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 44991–45013. PMLR.

Sowmya Vajjala and Ivana Lučić. 2018. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 297–304, New Orleans, Louisiana. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Shu Yang, Shenzhe Zhu, Ruoxuan Bao, Liang Liu, Yu Cheng, Lijie Hu, Mengdi Li, and Di Wang. 2024. What makes your model a low-empathy or warmth person: Exploring the origins of personality in llms. *arXiv preprint arXiv:2410.10863*.

Hanqing Zhang and Dawei Song. 2022. Discup: Discriminator cooperative unlikelihood prompt-tuning for controllable text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3392–3406.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

Carolina Zheng, Claudia Shi, Keyon Vafa, Amir Feder, and David Blei. 2023. An invariant learning characterization of controlled text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3186–3206, Toronto, Canada. Association for Computational Linguistics.

Lexical Attributes
Sophisticated Words
Unique Lexical Words
Sophisticated Lexical Words
Total Words
Sophisticated Word Count
Total lexical words
Total sophisticated lexical words
Lexical Sophistication
Verb Sophistication
Unique Word Ratio
Unique Verb Ratio
Unique Adjective Ratio
Unique Adverb Ratio
Age of Acquisition Score
Syntactic Attributes
Sentence Count
Verb Phrases
Clause Count
T-unit Count
Complex T-units
Dependent Clauses
Complex Nominals
Stop Words
Character Count
Words per Sentence
Characters per Sentence
Characters per Word
Syllables per Sentence
Coordinating Conjunctions
Noun Count
Numeral Count
Proper Nouns
Subordinating Conjunctions
Readability Level
Reading Time
Discourse Attributes
NORP Entities
GPE Entities
Law Entities
Money Entities
Ordinal Entities

Table 2: List of linguistic attribute names controlled by LINGCONV.

A Linguistic Attributes

Table 2 is a list of the linguistic attributes controlled by LINGCONV.

Index Descriptions Below we provide brief descriptions for a selection of the linguistic indices controlled by LINGCONV.

- **Lexical Words**: Content-bearing parts of speech, specifically nouns, verbs, adjectives, and adverbs.
- **Sophisticated Words**: Words considered less common in general usage, which we define as the 2,000 least frequent words in the American National Corpus.

Given Attributes	MSE	RMSE	ρ
20%	0.842	0.917	0.763
40%	0.625	0.791	0.851
60%	0.413	0.643	0.912
80%	0.286	0.535	0.945

Table 3: Performance of the MICE imputation algorithm with varying percentages of given attributes. Lower values are better for MSE, and RMSE, and higher values are better for ρ.

- **Age of Acquisition**: The typical age at which a word is learned and integrated into a person's vocabulary.
- **T-unit**: A minimal unit of syntax, consisting of one main clause and all associated subordinate clauses.
- **Complex Nominals**: A category of syntactic structures including: (i) nouns modified by elements such as adjectives, possessives, prepositional phrases, relative clauses, participles, or appositives; (ii) nominal clauses; and (iii) gerunds or infinitives serving as the subject.
- **Readability Level**: An estimate of the U.S. grade level required for a reader to comprehend a text, based on the Automated Readability Index.
- **GPE (Geopolitical) Entity**: Named countries, cities, and states.
- **NORP Entity**: Named nationalities, as well as religious and political groups.

Further details on these attributes can be found in the original works of Lu (2012), Lu (2010), and Lee and Lee (2023a).

B Imputation of missing values

Imputation of missing values is performed using the Multiple Imputation by Chained Equations (MICE) algorithm (Azur et al., 2011). For each missing linguistic attribute, a regression model is fitted using the other variables. The missing values are then imputed based on this model. This process is repeated for t iterations for each variable with missing data, forming a chain of equations that leads to an iterative refinement process. We use a ridge regression (Golub et al., 1999) linear model as the estimator.

The regression models are fitted using a training set consisting of N ground-truth linguistic attribute vectors (described below), coming from the training data of LINGCONV. Before the initial iteration of MICE, and to allow for predicting a missing attribute as a function of all other attributes, the missing values are initialized using the mean value for the attribute.

MICE provides a solution to handle missing data by leveraging the relationships among variables. In linguistic attributes, there are fixed relations between many of the attributes. Two examples are: any lexical count cannot be larger than the total number of words, and the number of clauses cannot be larger than the number of sentences in the text. By using linear regression models within the MICE framework, it ensures that the linear relationship assumption is maintained.

B.1 Performance of the MICE imputation algorithm

Table 3 shows the performance of the MICE imputation algorithm with varying percentages of given attributes. Lower values are better for MSE, and RMSE, and higher values are better for ρ.

B.2 The stored set of linguistic attributes vectors

The stored set of linguistic attributes vectors is selected from the training data using a greedy algorithm that maximizes the distance between the selected normalized linguistic attribute vectors. The idea is to select a subset of data points that are most representative of the entire dataset. We start by computing pairwise Euclidean distances between all points in the original dataset. Then, we select the next data point with the maximum average distance from already chosen points, ensuring that the selected subset is diverse. We set $N = 1000$, while the full training dataset of LINGCONV contains over 250k samples. To apply the MICE algorithm, we concatenate this representative subset with the vector of missing values, and perform the imputation. This subset also serves as a bank of valid linguistic attribute vectors, from which we sample **random targets**.

B.3 Adding or Subtracting Complexity

This feature allows users to increase or decrease the overall complexity of the target attributes. To ensure the linguistic attributes remain valid, any changes must be proportionally scaled according to their linear relation. We derive a set of attribute ratios from the training data to guide this process.

During the modification, we randomly select a scaling factor between 0.5 and 4. This factor adjusts the attribute ratios before applying the changes to the attributes. For example, if the number of sentences is increased by 1.0, the total number of words increases by 9.0 on average, while other attributes increase proportionally according to their linear relations.

C Training of Auxiliary Modules

The linguistic discriminator (LD) takes a tokenized sentence as input and is trained to minimize the MSE between the predicted attributes and the ground-truth attributes of the sentence. The objective is formulated as:

$$\ell_{disc}(x) = \|\mathrm{LD}(x) - l^x\|_2^2, \tag{1}$$

where x is the input sentence and l^x are its ground-truth linguistic attributes.

To ensure that the generated text remains semantically coherent with the source, the semantic embedding module (SEM) takes the source sentence s and the generated sentence $\hat{t}$ as input. It is trained using a contrastive objective to minimize the distance between embeddings of semantically equivalent pairs while maximizing the distance for non-equivalent pairs:

$$\ell_{sem}(s, \hat{t}) = -\log \frac{\mathrm{SE}(s, \hat{t})}{\sum_{t' \in \mathcal{N}(s)} \mathrm{SE}(s, t')}, \tag{2}$$

where $\mathcal{N}(s)$ represents the set of negative (non-paraphrase) examples for source s within the mini-batch. Both LD and SEM typically utilize architectures based on pre-trained encoders like T5, followed by appropriate projection layers for their respective tasks.

AgentMaster: A Multi-Agent Conversational Framework Using A2A and MCP Protocols for Multimodal Information Retrieval and Analysis

Callie C. Liao
Stanford University
ccliao@stanford.edu

Duoduo Liao
George Mason University
dliao2@gmu.edu

Sai Surya Gadiraju
George Mason University
sgadira3@gmu.edu

Abstract

The rise of Multi-Agent Systems (MAS) in Artificial Intelligence (AI), especially integrated with Large Language Models (LLMs), has greatly facilitated the resolution of complex tasks. However, current systems are still facing challenges of inter-agent communication, coordination, and interaction with heterogeneous tools and resources. Most recently, the Model Context Protocol (MCP) by Anthropic and Agent-to-Agent (A2A) communication protocol by Google have been introduced, and to the best of our knowledge, very few applications exist where both protocols are employed within a single MAS framework. We present a pilot study of AgentMaster, a novel modular multi-protocol MAS framework with self-implemented A2A and MCP, enabling dynamic coordination, flexible communication, and rapid development with faster iteration. Through a unified conversational interface, the system supports natural language interaction without prior technical expertise and responds to multimodal queries for tasks including information retrieval, question answering, and image analysis. The experiments are validated through both human evaluation and quantitative metrics, including BERTScore F1 (96.3%) and LLM-as-a-Judge G-Eval (87.1%). These results demonstrate robust automated inter-agent coordination, query decomposition, task allocation, dynamic routing, and domain-specific relevant responses. Overall, our proposed framework contributes to the potential capabilities of domain-specific, cooperative, and scalable conversational AI powered by MAS.

1 Introduction

Recent advances in artificial intelligence (AI) have increasingly focused on Multi-Agent Systems (MAS), in which multiple intelligent agents collaborate, communicate, and share contextual information to address complex tasks (Li et al., 2025; Qian et al., 2024; Yao et al., 2023). The integration of Large Language Models (LLMs) into MAS frameworks has significantly broadened their applicability, enabling general-purpose collaboration, natural language interaction, and open-ended reasoning (Hu et al., 2025; Luo et al., 2024; Huang et al., 2024; Guo et al., 2024). This makes LLM-based MAS particularly well-suited for dynamic, unstructured tasks such as multimodal data analysis, research automation, and intelligent assistance (Dong et al., 2024; Islam et al., 2024; Lin et al., 2025). By distributing intelligence across agents, LLM-based MAS offer a promising approach to overcoming the limitations of standalone LLMs (Gemini, 2025; OpenAI, 2024; Touvron et al., 2023).

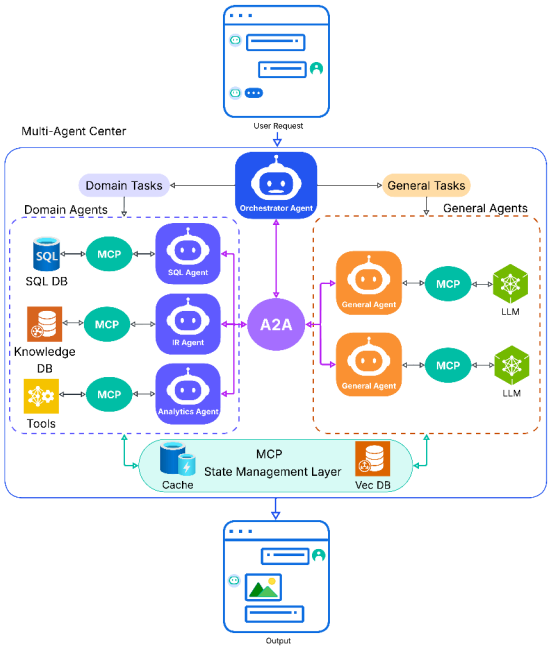


Figure 1: The general MAS framework of AgentMaster.

Despite their potential, current LLM-based MAS face critical challenges that limit their scalability, robustness, and effectiveness. These challenges span technical, architectural, and practical dimen-

[1]AgentMaster: demo video and live system.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 52–72
November 4-9, 2025

sions, including agent coordination, communication, interaction with heterogeneous tools and data sources, knowledge representation and reasoning, modularity, and integration of domain-specific expertise (Du et al., 2025; Shen et al., 2023). Especially, in domain-specific contexts where specialized agents are increasingly essential (Yu et al., 2025; Mathur et al., 2024; Gadiraju et al., 2024), these systems often require substantial domain-specific knowledge and the capability to process diverse data modalities, posing additional challenges for effective automation and coordination (Yu et al., 2025; Aminian-Dehkordi et al., 2025; Haase and Pokutta, 2025; Zhang et al., 2025b).

Most recently, two new open standards, Anthropic's Model Context Protocol (MCP) (Anthropic, 2024) and Agent-to-Agent (A2A) communication protocol introduced by Google (Surapaneni et al., 2025), aim to address these challenges. MCP, announced in May 2024, streamlines the process by providing a standardized interface for accessing various tools and resources, enhancing the modularity, interoperability, and statefulness of multi-agent and tool-augmented systems. A2A, announced in May 2025, complements MCP by facilitating structured inter-agent communication, which allows multiple AI agents to exchange messages, distribute subtasks, and build shared understanding to solve problems collectively. Both A2A and MCP can be developed using existing SDKs or fully implemented by users as needed. These protocols offer a systematic alternative to the fragmented, ad hoc integration approaches common in current MAS implementations (Jeong, 2025; Yang et al., 2025).

Existing LLM-based multi-agent systems that do not incorporate A2A or MCP often suffer from static coordination, limited memory, and rigid communication mechanisms. By leveraging these emerging standards, systems can support structured inter-agent communication, maintain shared contextual understanding, and seamlessly interface with external tools, developing more capable, scalable, and cooperative AI systems (Yang et al., 2025; Ehtesham et al., 2025).

To date, both industry and academia have conducted limited research on the application of A2A and MCP within LLM-based MAS. While a few research efforts have explored the independent use of A2A (Habler et al., 2025) and MCP (Krishnan, 2025; Qiu et al., 2025; Sarkar and Sarkar, 2025), there are, to the best of our knowledge, very few applications in which both protocols have been jointly employed within a single MAS framework.

To address these gaps, this paper introduces AgentMaster, a novel modular multi-protocol MAS framework that integrates A2A protocol and MCP. AgentMaster decomposes user queries into specialized workflows executed by dedicated agents, coordinated through A2A and supported by a centralized MCP backend for tool and context management. Users interact with the system through a unified conversational interface, enabling natural language interaction without prior technical expertise. The framework supports automated complex task decomposition, dynamic routing, and agent-to-agent orchestration. By isolating agents and provisioning separate API keys, the system can manage resource utilization and enforce the separation of concerns between components.

A fully functional prototype through self-developed A2A and MCP demonstrates AgentMaster's capabilities in domain-specific multimodal tasks, including information retrieval, image analysis, database querying, question answering, and content summarization. The system is deployed both locally and on Amazon Web Services (AWS) as a set of Flask-based microservices, and exhibits consistent performance across varied task types in a pilot study.

Our main contributions are as follows:

- This paper introduces AgentMaster, a modular multi-agent MAS framework that integrates Anthropic's MCP and Google's A2A protocol to enable flexible inter-agent communication, intelligent coordination, and retrieval-augmented generation.
- A unified system architecture is designed to support automated query decomposition, task allocation, dynamic routing, and orchestration across specialized retrieval agents and multimodal data sources.
- The pilot study explores the implementation of self-developed A2A and MCP protocols specifically designed for AgentMaster without relying on existing libraries such as Google's A2A SDK.
- Comprehensive evaluation is conducted using G-Eval, BERTScore, and related metrics to validate correctness, completeness, and semantic fidelity across diverse query types.

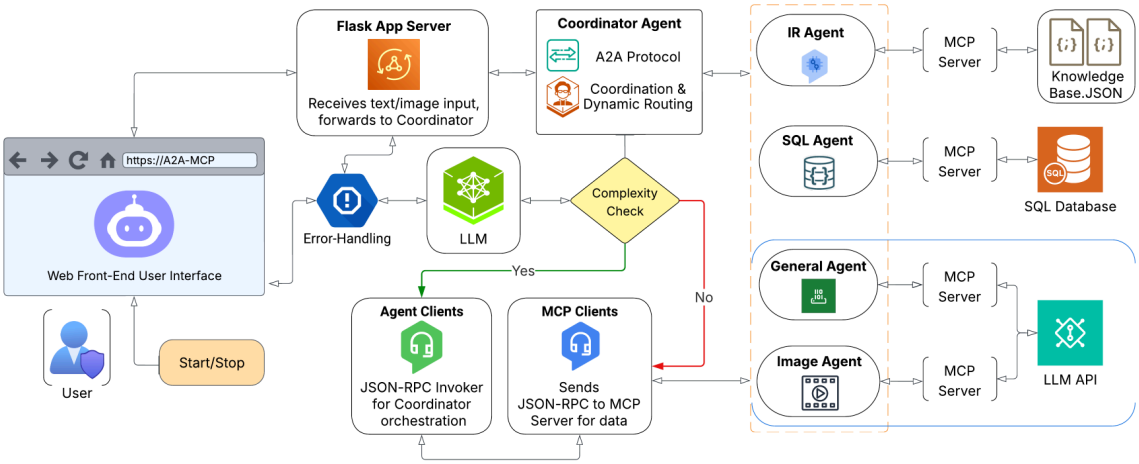


Figure 2: The system architecture of the case study.

2 The General System Framework

Figure 1 illustrates the general multi-protocol MAS architecture of the AgentMaster. The framework comprises four core components: a unified conversational interface, a multi-agent center, multi-agent AI protocols, and a state management layer.

2.1 Unified Conversational Interface

The unified conversational interface resembles a chatbot, receiving user input in various multimodal formats, including text, charts, images, and audio, and generating corresponding output in modalities such as text, images, and structured data tables.

2.2 Multi-Agent Center

The Multi-Agent Center consists of three hierarchical layers of agents: the orchestrator agent, domain agents, and general agents. At the top of the hierarchy, the orchestrator agent is responsible for decomposing tasks and coordinating execution across agents. Domain agents specialize in specific functionalities and may be either LLM-based or non-LLM-based. General agents operate independently, each paired with a dedicated LLM to handle general-purpose reasoning tasks. All agents communicate through the A2A protocol, which enables structured, language-based message exchange. Additionally, each agent is integrated with the MCP protocol, which standardizes interactions with external tools, APIs, and contextual resources.

2.2.1 Orchestrator Agent

The orchestrator agent serves as the central coordinator, identifying available tasks and delegating them to appropriate agents based on their capabilities. To optimize efficiency and accuracy, it may further decompose complex user requests into subtasks for parallel or sequential execution across agents. As a pivotal hub, it not only translates high-level user goals into manageable tasks aligned with agent capabilities, but also facilitates inter-agent communication, handles error management across protocols, and synthesizes outputs into a coherent, unified response.

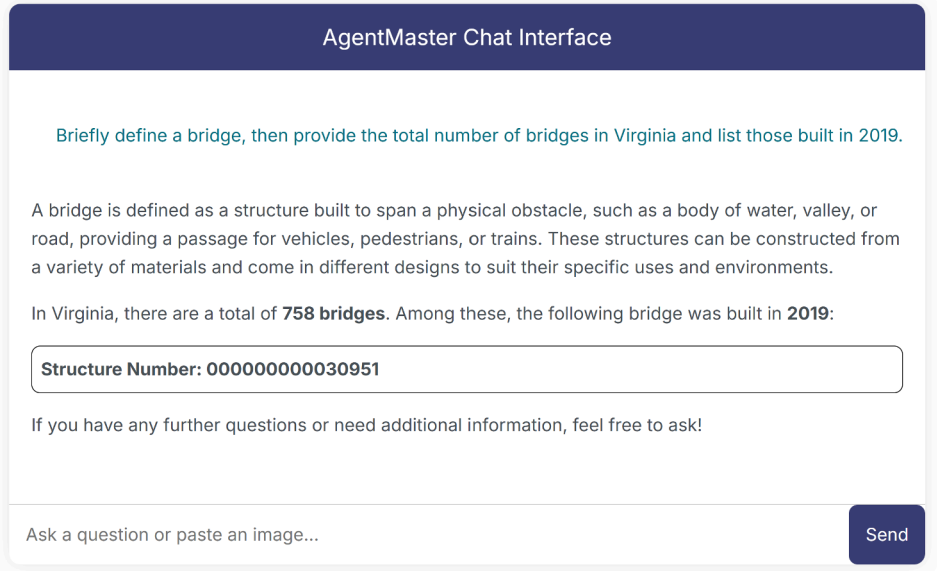
2.2.2 General Agents

The general agent is designed to handle broad tasks that do not require access to domain-specific datasets. The orchestrator agent determines whether to delegate a task to a general agent or a domain agent, selecting the most appropriate agent based on the nature and complexity of the task.

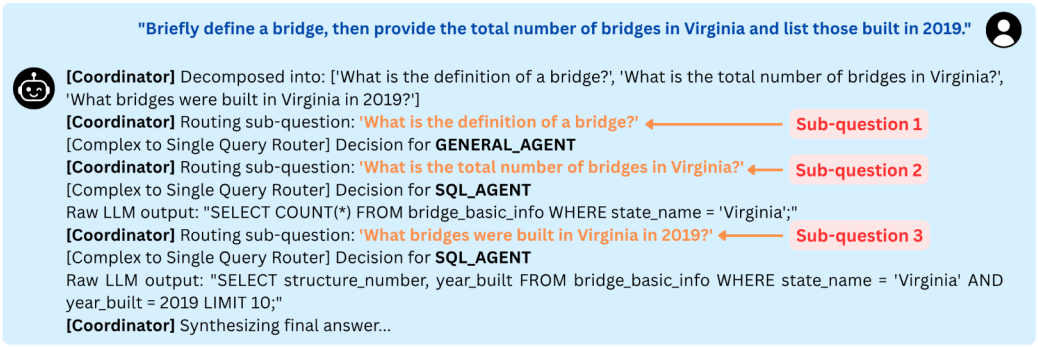
2.2.3 Domain Agents

Domain agents are specialized for specific domains and designed to interface with domain-relevant functions, datasets, and tools. Each domain agent may internally manage sub-agents to further decompose and process tasks in a modular fashion. These agents communicate not only with each other, but also with general agents, enabling collaborative task execution across domains.

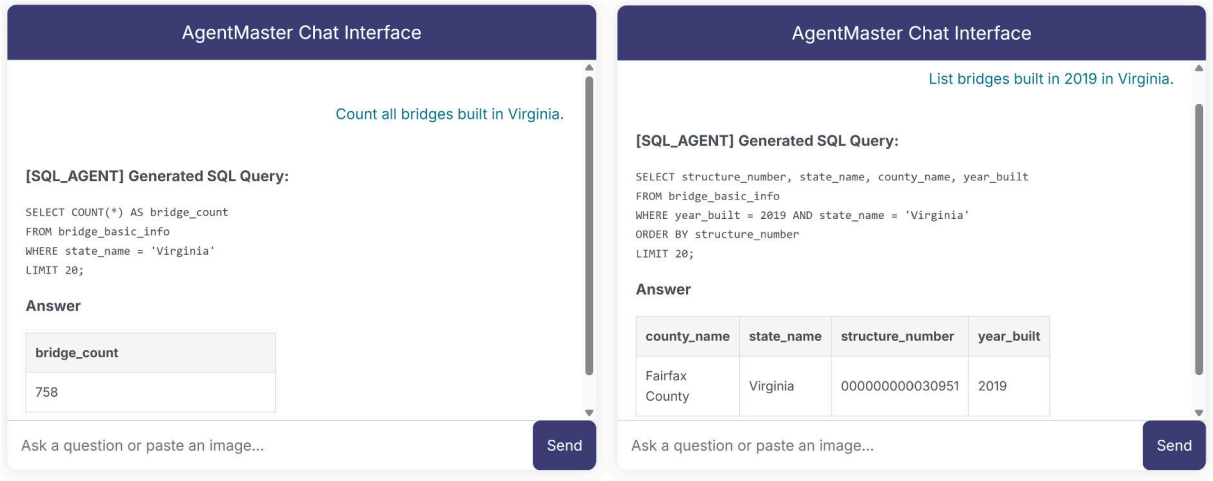
In AgentMaster, domain agents are designed to specialize in common domain-specific tasks, including Structured Query Language (SQL) querying, Information Retrieval (IR), and multimodal data analytics. The framework is extensible, allowing for the integration of additional agents as required to support diverse application needs.



(a) Frontend example.



(b) Backend example.



(c) Verification of the generated responses to the complex query.

Figure 3: AgentMaster demonstration example and verification.

2.3 Multi-Agent AI Protocols

AgentMaster employs A2A for structured communication between agents, enabling coordination, delegation, and orchestration through standardized JSON-based message exchange. MCP complements this by providing a unified interface for tool access, long-term memory, and context management, enhancing modularity, interoperability, and statefulness in LLM-based agents.

Depending on the application and requirements, the framework leverages the A2A protocol via Google's A2A SDK (Surapaneni et al., 2025), or

fully implements it as needed. MCP is developed in a similar manner.

2.4 State Management Layer

The State Management Layer in AgentMaster leverages vector databases and context caches to maintain the MCP state, enabling agents to be context-aware and memory-augmented for efficient handling of multistep, user-specific, and domain-specific tasks. This layer utilizes the vector database to provide persistent semantic memory for retrieving relevant past interactions and documents, while the context cache offers fast, temporary storage for session data and intermediate results during active workflows.

3 System Architecture of the Case Study

Figure 2 illustrates the architecture of a conversational MAS, an example implementation of the AgentMaster framework for multimodal information retrieval and analysis. The system integrates modular components to enable robust, retrieval-augmented question answering through dynamic agent orchestration.

The architecture comprises a web-based user interface, a Flask server acting as the main entry point, a Coordinator agent (i.e., the Orchestrator agent) implementing the A2A protocol, and multiple specialized retrieval agents (i.e., domain agents). User queries are submitted via the chatbot front end and processed asynchronously by the backend components.

3.1 Coordinator Agent and Complexity Assessment

The Coordinator agent is responsible for query analysis, routing, and orchestration (Zhang et al., 2025a). A key function is the complexity assessment module, which determines whether a query requires multi-agent collaboration or can be handled by a single retrieval agent. For simple queries, the Coordinator dispatches requests directly to an appropriate MCP client. In contrast, complex queries trigger agent clients that dynamically coordinate multiple retrieval workflows.

3.2 Agent Clients and MCP Clients

Agent clients serve as JSON-RPC invokers for orchestrating distributed workflows among retrieval agents. MCP clients manage communication with retrieval backends, dispatching JSON-RPC requests to MCP servers that encapsulate domain-specific retrieval logic (Kumar et al., 2025). This division enables the system to support compositional retrieval and fallback handling without manual routing configuration.

3.3 Retrieval Agents

The system incorporates four primary specialized agents: (i) an IR agent that retrieves unstructured content from knowledge bases; (ii) a SQL agent that generates and executes SQL queries over relational databases; (iii) an image agent that processes image inputs through external vision APIs; and (iv) a general agent that handles open-domain queries and fallback cases. Each agent exposes an MCP server endpoint for standardized invocation.

3.4 LLM Integration and Error Handling

The architecture integrates a local or external LLM for language generation, reasoning, and summarization. The LLM module aggregates partial outputs returned by retrieval agents and formulates the final response. The Flask server and Coordinator agent include error-handling mechanisms that detect and recover from failures in retrieval workflows and model inference (Williams, 2025).

3.5 End-to-End Workflow

End-to-end query resolution proceeds as follows. The user submits a text or image query via the front end. The Flask server forwards the request to the Coordinator agent, which performs complexity assessment and routes the query to the appropriate retrieval pathway. Specialized retrieval agents return results via MCP clients. The LLM module synthesizes the final output, which is delivered to the user interface for presentation.

3.6 Design Considerations

The A2A-MCP design emphasizes modularity, extensibility, and reproducibility. New retrieval agents can be integrated without modifying the orchestration logic. The standardized JSON-RPC interfaces facilitate consistent communication across agents (Zhang et al., 2025a). This architecture provides a flexible foundation for retrieval-augmented conversational systems and supports future research into multi-agent LLM collaboration.

4 Experimental Results and Evaluation

In this case study, the AgentMaster system is deployed locally as well as on AWS to facilitate inter-

net access. Each agent leverages OpenAI's GPT-4o mini model. Three domain agents, derived from our prior research, focus on SQL (Gadiraju et al., 2025), IR (Gadiraju et al., 2024, 2025), and image analysis (Darji et al., 2024), utilizing the Federal Highway Administration (FHWA) public datasets (Federal Highway Administration, 2025).

Experiments were conducted to evaluate both individual agents and agent-to-agent collaborations using simple and complex queries. Multiple evaluation metrics are employed to assess the multi-agent system, including agentic metrics, LLM-as-a-Judge (Zheng et al., 2023), and human evaluation. Agentic metrics assess autonomy and effectiveness of AI agents in complex tasks. LLM-as-a-Judge uses a large language model to evaluate outputs of another LLM for correctness, relevance, and coherence. Human evaluation remains the gold standard for validating these assessments in this pilot study.

4.1 Individual Agent Evaluation

Three domain agents (SQL, IR, and Image) were previously evaluated independently in our past research and demonstrated high reliability and accuracy (Gadiraju et al., 2024, 2025; Darji et al., 2024). Additionally, due to the robustness of the GPT model, individual queries or single tasks have consistently yielded correct results in our testing. However, there are occasional instances of misclassifying single queries as complex queries for query decomposition, resulting in incorrect responses.

4.2 Complex Task Evaluation

To evaluate the quality and accuracy of AgentMaster's responses, sub-questions decomposed from complex queries were individually submitted to AgentMaster. The outputs generated for these simpler sub-questions were then compared to the corresponding segments within AgentMaster's responses to the overall complex queries. Since the sub-questions are simple queries, it would not require mutli-agent collaboration and thus can serve as a verification method for AgentMaster's output.

Figure 3 presents the front-end and back-end of the demonstration, as well as the verification of AgentMaster's generated response. As shown in Figure 3a, AgentMaster responds with a domain-specific full response to a complex user query by providing a combination of relevant specific information from the database and general information. Figure 3b displays the coordinator agent decomposing the complex query into sub-questions before assigning each sub-question to the appropriate agents. In the example, the general agent and the SQL agent were employed to generate partial responses, which were sent back to the coordinator to integrate them into a cohesive final response. Additionally, in Figure 3c, the corresponding sub-questions were submitted to AgentMaster to validate the complex query results, and the simple query results were found to be consistent with the information in the complex query responses. AgentMaster was queried for the total number of bridges built in Virginia and those built in Virginia in 2019, and correct information was provided, indicating accurate routing of the complex query and successful SQL database retrieval. Similarly, Figure 11 in Appendix A.1 displays complex query evaluation, verifying the reliability of AgentMaster.

ID	Num of Sub-Questions	Assigned Agents
Q1	3	General, SQL, SQL
Q2	3	IR, SQL, SQL
Q3	5	IR, SQL, IR, SQL
Q4	3	SQL, SQL, IR
Q5	2	SQL, General
Q6	8	8 IRs

Table 1: The number of query decompositions and the corresponding path for each complex query.

Table 2: Evaluation Metrics by Query Type

Query Type	G-Eval (%)	BERT Precision (%)	BERT F1 (%)
SQL Queries	92.0	98.8	98.7
IR Queries	90.2	97.6	97.8
General QA	84.0	95.7	96.8
Image/Complex QA	82.0	90.1	91.9
Average	**87.1**	**95.6**	**96.3**

As shown in Table 1, six complex queries were submitted to AgentMaster. The Coordinator agent performed query decomposition into multiple sub-questions, which were automatically assigned to appropriate agents according to their capabilities. The automated complex actions of the backend – query decomposition, task allocation, dynamic routing – as well as resulting outputs are presented in Tables 5-6 and Figures 12-14 in Appendix A.2. Human evaluation, based on the agentic metrics comprised of task completion and correction, revealed that each complex query was correctly decomposed, with most agent task paths correctly assigned.

Table 3: Feature-Level Comparisons between LLM-Based MAS without A2A or MCP (MAS-0), A2A-Only, MCP-Only, and AgentMaster

Feature	**MAS-0**	**A2A-Only**	**MCP-Only**	**AgentMaster (A2A + MCP)**
Memory	Limited (static or no memory)	Flexible (local and dynamic)	None	Flexible (local and shared)
Coordination	Static	Dynamic	Static	Dynamic
Scalability	Moderate	High	Moderate	High
Fallback Strategy	Limited	Adaptive	Limited	Robust, adaptive
Failure Tolerance	Low	High	Low	High
Inter-Agent Communication	Limited or none	Strong	Limited or none	Strong
Architecture	Centralized	Peer-to-peer	Centralized	Hybrid
Task Allocation	Manual or implicit	Decentralized	Centralized	Hybrid
Learning Capability	None or minimal	Local/online	Centralized/offline	Hybrid: adaptive, distributed, and online

4.3 Overall Evaluation

The overall A2A-MCP framework was evaluated across multiple dimensions, including factual correctness, relevance, completeness, and semantic similarity. Metrics included Answer Relevancy, Hallucination detection, G-Eval (LLM-based assessment) (Liu et al., 2023), and BERTScore (Zhang et al., 2024). The test set comprised diverse queries spanning SQL retrieval, IR, general knowledge, and summarization.

Table 2 reports the aggregated metrics across all query types for 23 questions, including both simple and complex questions. Overall, the system demonstrates strong correctness and semantic alignment, with Answer Relevancy and Hallucination metrics indicating high reliability across domains. The average G-Eval score for complex queries exceeds 87.1%, while BERTScore F1 averages 96.3%, reflecting high semantic fidelity to reference outputs.

In the individual agent evaluation, the SQL agent and IR agent produce consistently accurate results, while the general agent and image agent show minor variability due to open-ended generation. Evaluation of complex queries confirms effective decomposition and integration by the Coordinator agent, with most sub-questions yielding outputs consistent with the composite responses.

Table 3 presents the summarized feature-based and overall comparisons between existing systems and AgentMaster, respectively, showcasing the comprehensive features of AgentMaster with A2A and MCP.

5 Conclusions

This paper presents AgentMaster, a novel modular conversational framework leveraging A2A-MCP protocols for retrieval-augmented question answering across structured, unstructured, and multimodal data sources, facilitating structural clarity and code efficiency as well as easier scalability and maintenance. By interacting with AgentMaster using natural language communication, users can receive domain-specific information regardless of expertise. The experimental results BERTScore F1 and LLM-as-a-Judge metric G-Eval average 96.3% and 87.1%, yielding high performance. Validation through both human evaluation and quantitative metrics demonstrates the ability to effectively coordinate various agents, perform complex actions, and produce accurate, semantically faithful responses. The proposed architecture highlights the potential of agent-based orchestration for scalable, domain-adaptive conversational AI.

6 Limitations

While the framework achieved strong performance across diverse query types, some limitations remain. The accuracy of retrieval and generation is partly constrained by the underlying LLM and retrieval corpus. Occasional misclassification of query complexity can lead to unnecessary decomposition or incomplete responses. Limited inter-agent collaboration and the constrained size of the database occasionally led to responses with minimal informational depth. The LLM-based reasoning process may encounter challenges in synthesizing complex information. While LLM-as-a-judge evaluation offers scalability and efficiency, it remains limited by potential biases, lack of task-specific expertise, and alignment with human judgment. Finally, the current framework lacks established security safeguards for information storage and usage. These limitations can be addressed in future work.

7 Acknowledgments

The authors thank the Federal Highway Administration (FHWA) for providing public datasets used to build knowledge databases for the case study. The authors also thank Ellie L. Zhang for contributing to framework building.

References

Javad Aminian-Dehkordi, Mohammad Parsa, Mohsen Naghipourfar, and Mohammad R.K. Mofrad. 2025. Koda: An agentic framework for kegg orthology-driven discovery of antimicrobial drug targets in gut microbiome. *bioRxiv*.

Anthropic. 2024. Introducing the model context protocol.

Viraj Nishesh Darji, Callie C. Liao, and Duoduo Liao. 2024. Automated interpretation of non-destructive evaluation contour maps using large language models for bridge condition assessment. In *2024 IEEE International Conference on Big Data (BigData)*, pages 3258–3263.

Yubo Dong, Xukun Zhu, Zhengzhe Pan, Linchao Zhu, and Yi Yang. 2024. Villageragent: A graph-based multi-agent framework for coordinating complex task dependencies in minecraft. *Preprint*, arXiv:2406.05720.

Hung Du, Srikanth Thudumu, Rajesh Vasa, and Kon Mouzakis. 2025. A survey on context-aware multi-agent systems: Techniques, challenges and future directions. *Preprint*, arXiv:2402.01968.

Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. 2025. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *Preprint*, arXiv:2505.02279.

Federal Highway Administration. 2025. Fhwa infotechnology portal. Accessed: 2025-7-4.

Sai Surya Gadiraju, Zijie He, and Duoduo Liao. 2025. A conversational agent framework for multimodal knowledge retrieval: A case study in fhwa infohighway web portal queries. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025) Workshop for Research on Agent Language Models (REALM 2025)*, Vienna, Austria. ACL.

Sai Surya Gadiraju, Duoduo Liao, Akhila Kudupudi, Santosh Kasula, and Charitha Chalasani. 2024. InfoTech Assistant: A Multimodal Conversational Agent for InfoTechnology Web Portal Queries . In *2024 IEEE International Conference on Big Data (BigData)*, pages 3264–3272, Los Alamitos, CA, USA. IEEE Computer Society.

Gemini. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *Preprint*, arXiv:2402.01680.

Jennifer Haase and Sebastian Pokutta. 2025. Beyond static responses: Multi-agent llm systems as a new paradigm for social science research. *Preprint*, arXiv:2506.01839.

Idan Habler, Ken Huang, Vineeth Sai Narajala, and Prashant Kulkarni. 2025. Building a secure agentic ai application leveraging a2a protocol. *Preprint*, arXiv:2504.16902.

Li Hu, Guoqiang Chen, Xiuwei Shang, Shaoyin Cheng, Benlong Wu, Gangyang Li, Xu Zhu, Weiming Zhang, and Nenghai Yu. 2025. Compileagent: Automated real-world repo-level compilation with tool-integrated llm-based agent system. *Preprint*, arXiv:2505.04254.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. 2024. QueryAgent: A reliable and efficient reasoning framework with environmental feedback based self-correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5014–5035, Bangkok, Thailand. Association for Computational Linguistics.

Md. Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. MapCoder: Multi-agent code generation for competitive problem solving. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4912–4944, Bangkok, Thailand. Association for Computational Linguistics.

Cheonsu Jeong. 2025. A study on the mcp x a2a framework for enhancing interoperability of llm-based autonomous agents. *Preprint*, arXiv:2506.01804.

Naveen Krishnan. 2025. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *Preprint*, arXiv:2504.21030.

Sonu Kumar, Anubhav Girdhar, Ritesh Patil, and Divyansh Tripathi. 2025. Mcp guardian: A security-first layer for safeguarding mcp-based ai system. *Preprint*, arXiv:2504.12757.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. 2025. Agent-oriented planning in multi-agent systems. *Preprint*, arXiv:2410.02189.

Hongzhan Lin, Yang Deng, Yuxuan Gu, Wenxuan Zhang, Jing Ma, See-Kiong Ng, and Tat-Seng Chua. 2025. Fact-audit: An adaptive multi-agent framework for dynamic fact-checking evaluation of large language models. *Preprint*, arXiv:2502.17924.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Qinyu Luo, Yining Ye, Shihao Liang, Zhong Zhang, Yujia Qin, Yaxi Lu, Yesai Wu, Xin Cong, Yankai Lin, Yingli Zhang, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024. RepoAgent: An LLM-powered open-source framework for repository-level code documentation generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 436–464, Miami, Florida, USA. Association for Computational Linguistics.

Puneet Mathur, Alexa Siu, Nedim Lipka, and Tong Sun. 2024. MATSA: Multi-agent table structure attribution. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 250–258, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1088–1113, Bangkok, Thailand. Association for Computational Linguistics.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, Shilong Liu, Xun Jiang, Liu Leqi, and Mengdi Wang. 2025. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *Preprint*, arXiv:2506.14728.

Anjana Sarkar and Soumyendu Sarkar. 2025. Survey of llm agent communication with mcp: A software design pattern centric review. *Preprint*, arXiv:2506.05364.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Preprint*, arXiv:2303.17580.

Rao Surapaneni, Miku Jha, Michael Vakoc, and Todd Segal. 2025. Announcing the agent2agent protocol (a2a). Google Developer Blog.

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. *Preprint*, arXiv:2302.13971.

Dean Lawrence Williams. 2025. Multi-agent communication protocol in collaborative problem solving: A design science approach.

Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, Weiwen Liu, Ying Wen, Yong Yu, and Weinan Zhang. 2025. A survey of ai agent protocols. *Preprint*, arXiv:2504.16736.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan W. Suchow, Zhenyu Cui, Rong Liu, Zhaozhuo Xu, Denghui Zhang, Koduvayur Subbalakshmi, Guojun Xiong, Yueru He, Jimin Huang, Dong Li, and Qianqian Xie. 2025. Fincon: a synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Wentao Zhang, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. 2025a. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *Preprint*, arXiv:2506.12508.

Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Yao Zhang, Zaixi Shang, and Silpan Pateland Mikel Zuniga. 2025b. From unstructured communication to intelligent rag: Multi-agent automation for supply chain knowledge bases. *Preprint*, arXiv:2506.17484.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A Appendix

The appendix contains more demonstrations of multimodal inputs and outputs, single and complex queries, verification, automated actions (query decomposition, task allocation, and dynamic routing) for complex queries in the backend, as well as different usages of the domain-specific or specialized agents.

A.1 AgentMaster Query Demonstrations

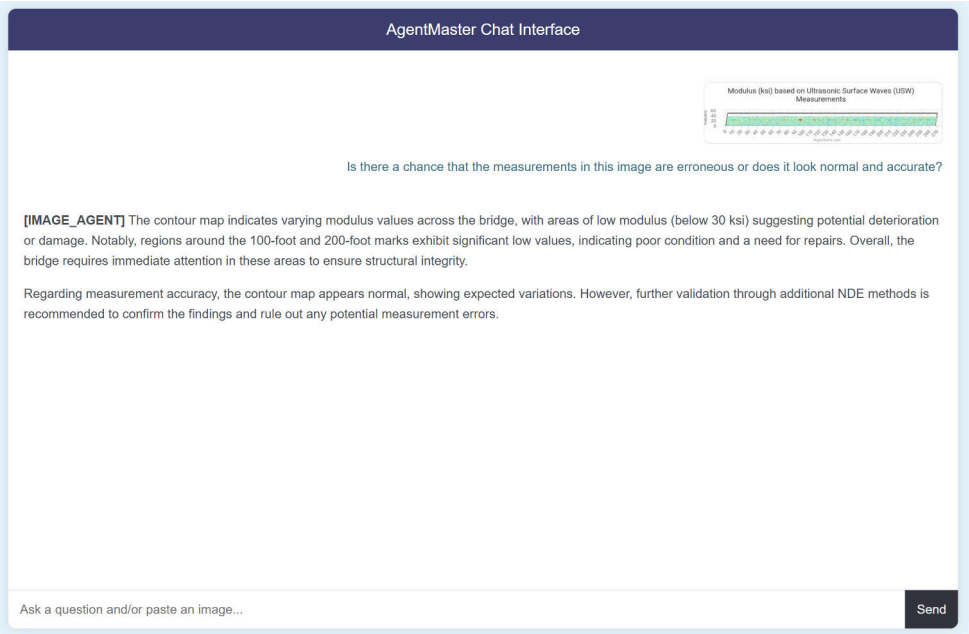


Figure 4: Frontend example of a single query for the Image agent.

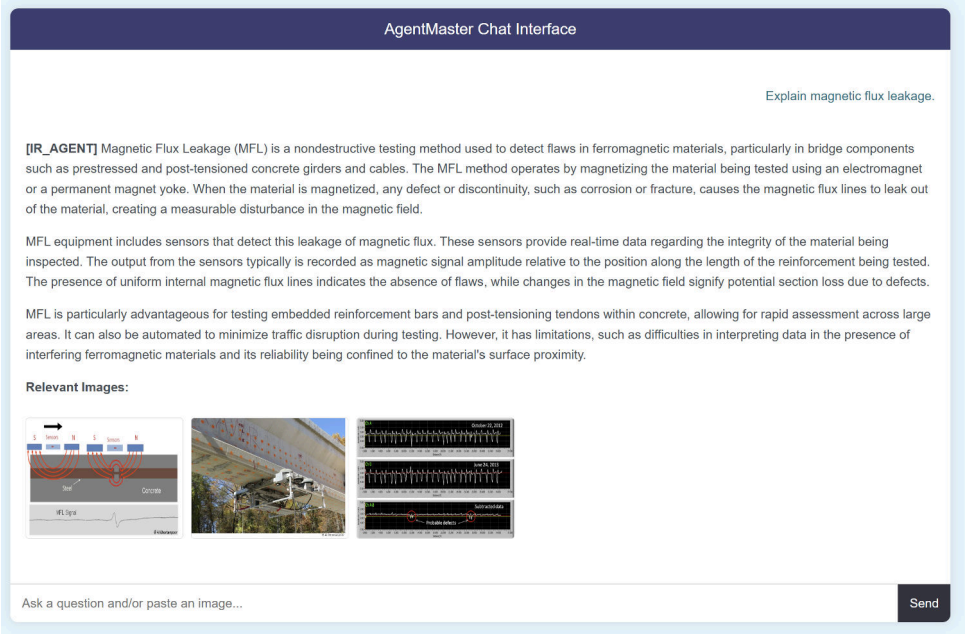
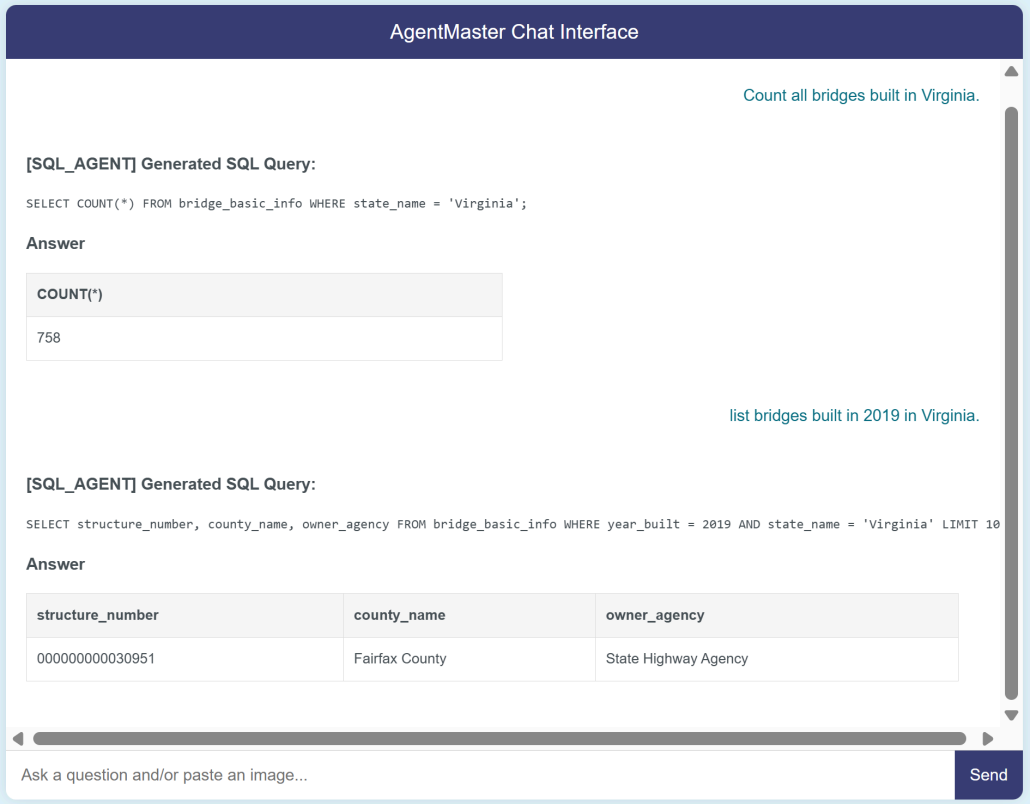
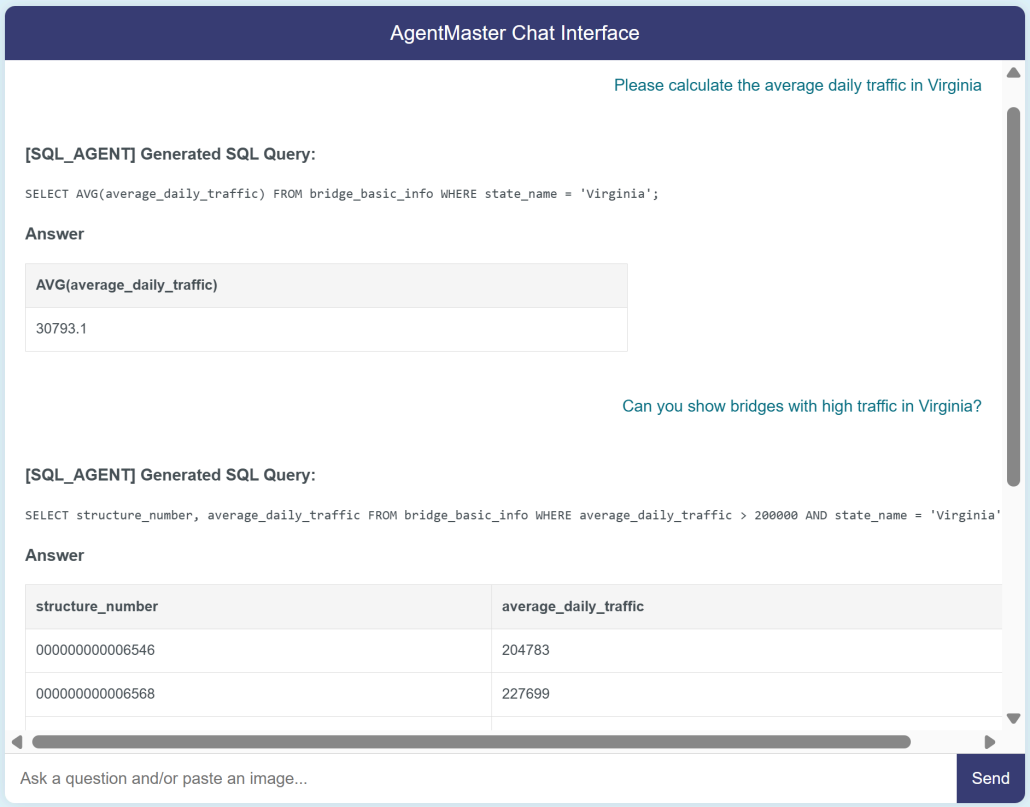


Figure 5: Frontend example of a single query for the IR agent.

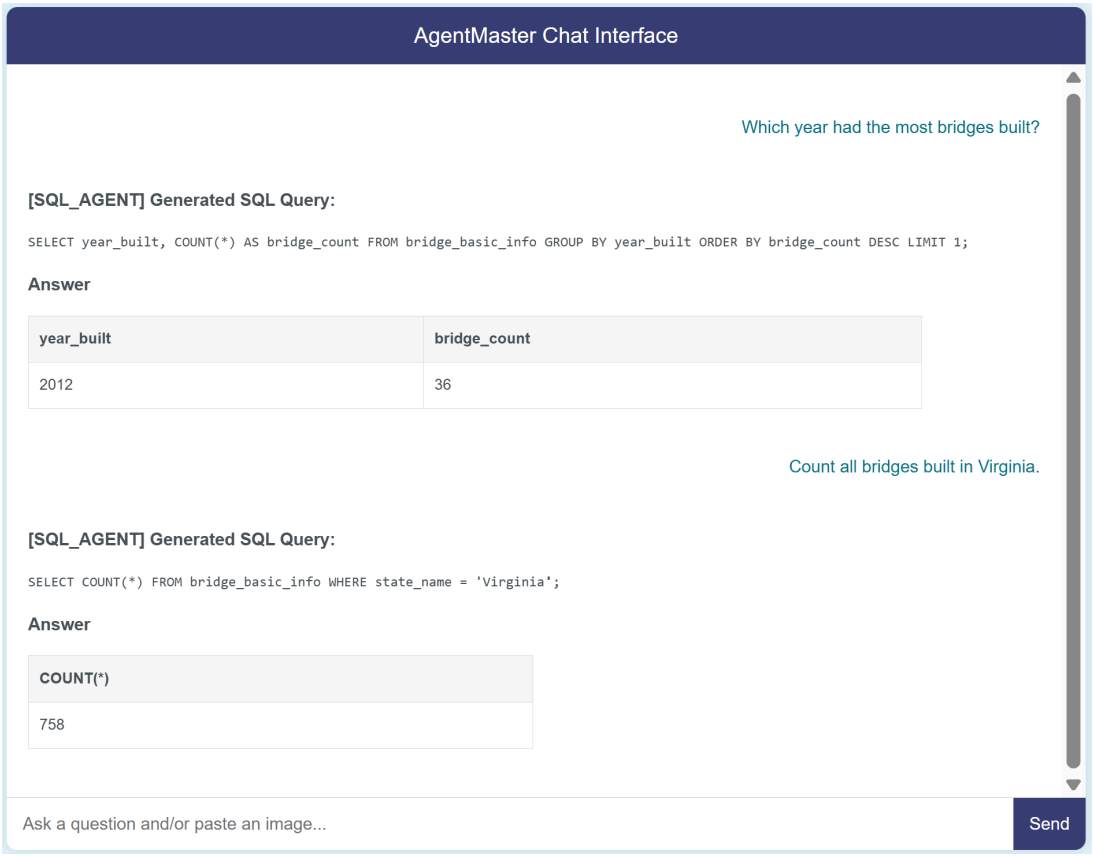


(a) Frontend example of two single queries for the SQL agent.

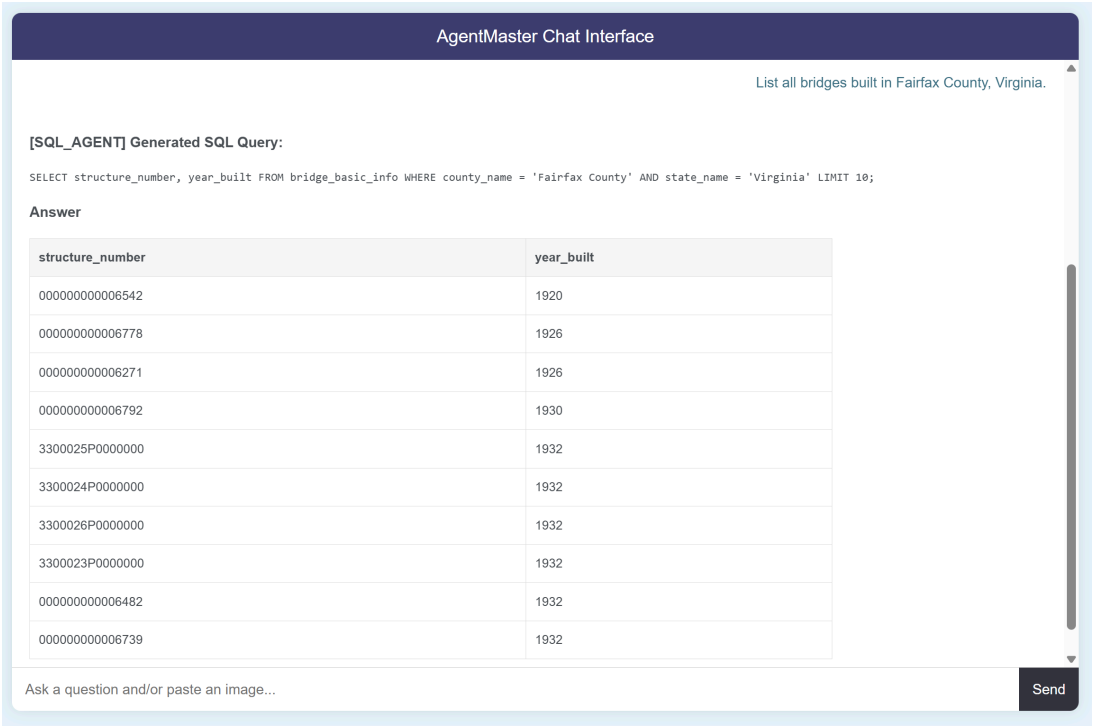


(b) Frontend example of two single queries for the SQL agent.

Figure 6: More demonstration examples of a single query for the SQL agent.



(a) Frontend example of two single queries for the SQL agent.



(b) Frontend example of one single query for the SQL agent.

Figure 7: More demonstration examples of a single query for the SQL agent.

AgentMaster Chat Interface

What are the limitations of GPR?

[IR_AGENT] The limitations of GPR (Ground Penetrating Radar) include:

1. **Requires advanced expertise and training** for data collection, processing, and interpretation.
2. **Primarily sensitive** to identifying corrosive environments within the upper half of the deck when measurements are made from the deck surface.
3. **Data negatively influenced** by extremely cold conditions.
4. **De-icing agents** can limit the signal penetration.
5. **Does not directly assess** deck delamination, except under very restrictive conditions (e.g., epoxy-injected repairs separating two deck sections or delaminated areas filled with water).
6. **Requires calibration** with other NDE methods and other ground truth activities.
7. **Less cost-effective** than other methods, particularly when applied to smaller individual structures.
8. **Accuracy is limited** to the position and depth of bars in the upper half of the deck. The lower half of the deck can be difficult to assess due to the rebar mat in the upper half.

Relevant Images:

Ask a question and/or paste an image... Send

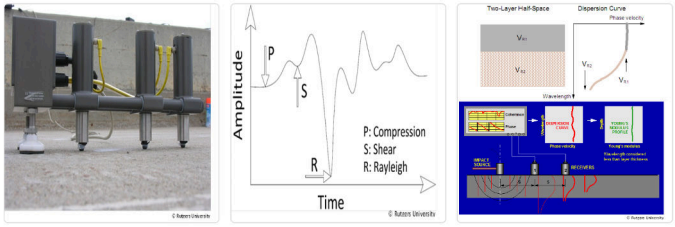
(a) A single query for the IR agent.

AgentMaster Chat Interface

What is Ultrasonic Surface Waves? Please give a definition with 50 words.

[IR_AGENT] Ultrasonic Surface Waves (USW) are elastic waves that travel along the free surface of materials, primarily used for non-destructive evaluation. USW techniques assess concrete quality, detect delaminations, estimate crack depths, and evaluate material conditions by analyzing the phase velocity of surface waves in relation to the material's properties.

Relevant Images:

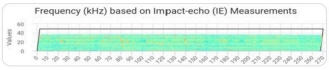


Ask a question and/or paste an image... Send

(b) A single query of the IR agent.

Figure 8: Two examples of single verification queries for the IR agent.

AgentMaster Chat Interface



Please interpret the image and briefly explain the impact echo.

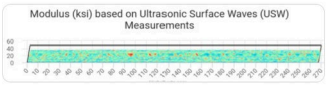
[IMAGE_AGENT] The contour map indicates varying frequencies based on Impact-echo (IE) measurements across the bridge. Areas with lower frequency values suggest potential delamination or voids, indicating poor structural integrity. Specific regions, particularly those showing values significantly below the average, may require immediate repairs. Overall, the bridge exhibits signs of deterioration, necessitating further investigation and maintenance in identified areas.

Impact Echo Explanation: Impact-echo is a non-destructive testing method used to evaluate the integrity of concrete structures. It involves generating stress waves through impacts and analyzing the reflected waves to detect flaws, such as cracks or voids, based on the time taken for echoes to return.

Ask a question and/or paste an image... Send

(a) A single query for the image agent.

AgentMaster Chat Interface



Please show the defect location in the image and summary the bridge condition.

[IMAGE_AGENT] The contour map indicates varying modulus values across the bridge, with areas showing lower modulus (indicated by red and yellow) suggesting compromised structural integrity. Specifically, regions with values below 20 ksi are concerning and likely require repairs. These areas are likely located around the mid-section and towards the ends of the bridge. Overall, the bridge exhibits signs of deterioration, necessitating further investigation and maintenance to ensure safety.

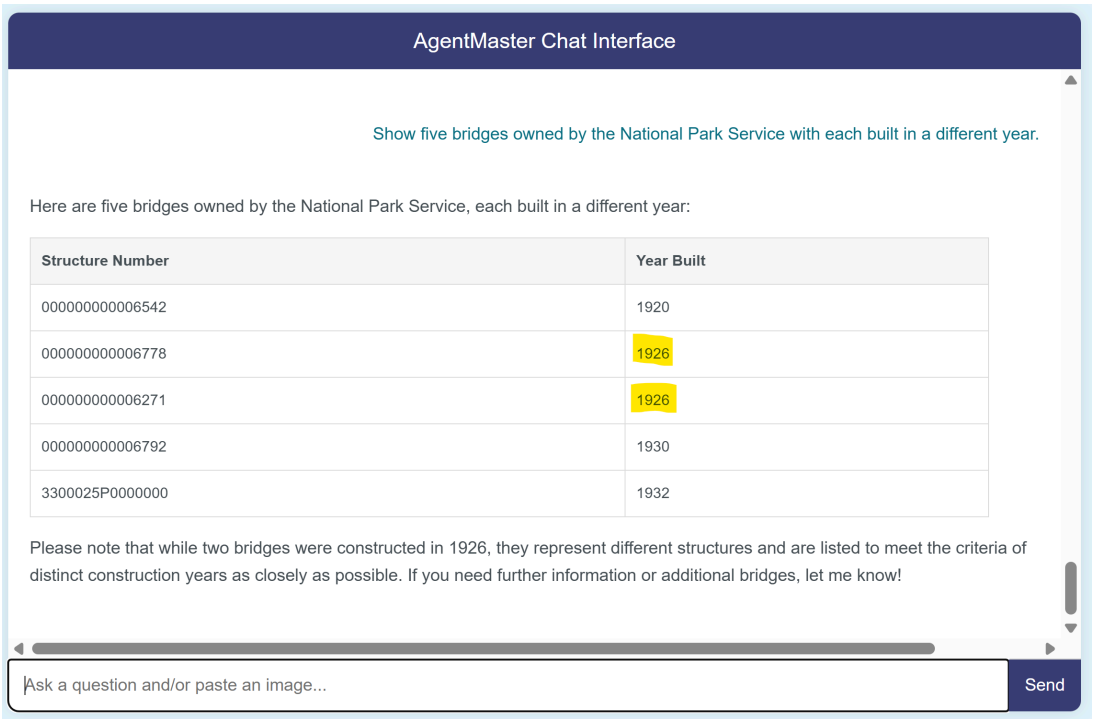
Defect Location: Areas with modulus values below 20 ksi, primarily in the mid-section and ends.

Summary: The bridge shows signs of poor condition in specific areas, indicating a need for repairs.

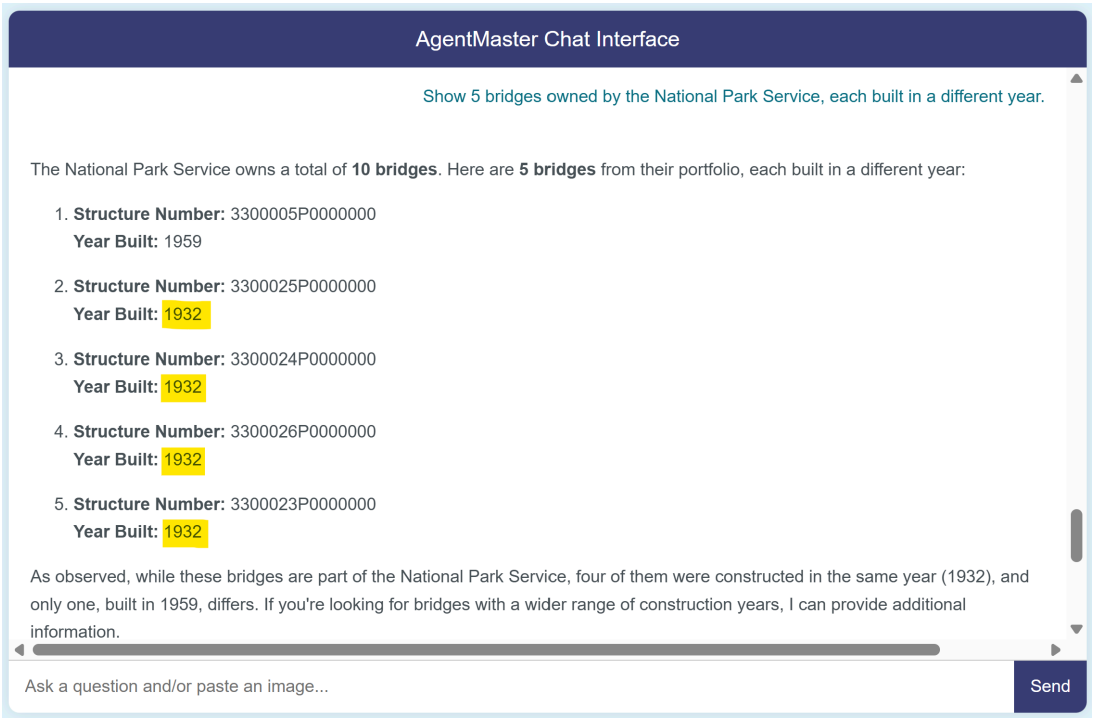
Ask a question and/or paste an image... Send

(b) A single query for the image agent.

Figure 9: Two examples of single verification queries for the image agent.

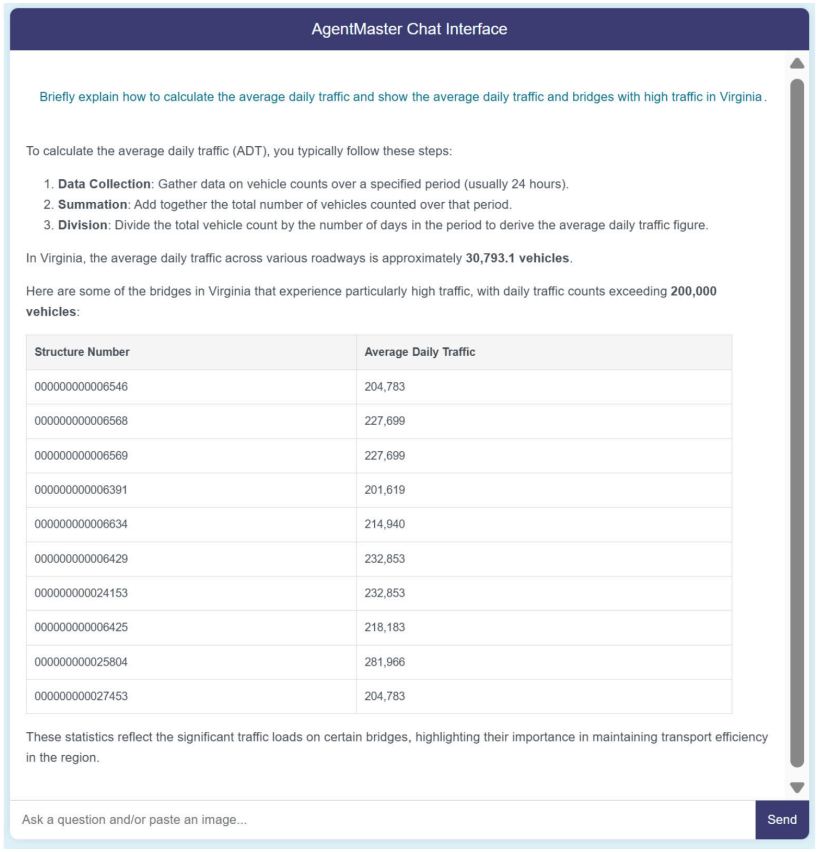


(a) SQL agent error from a single query.

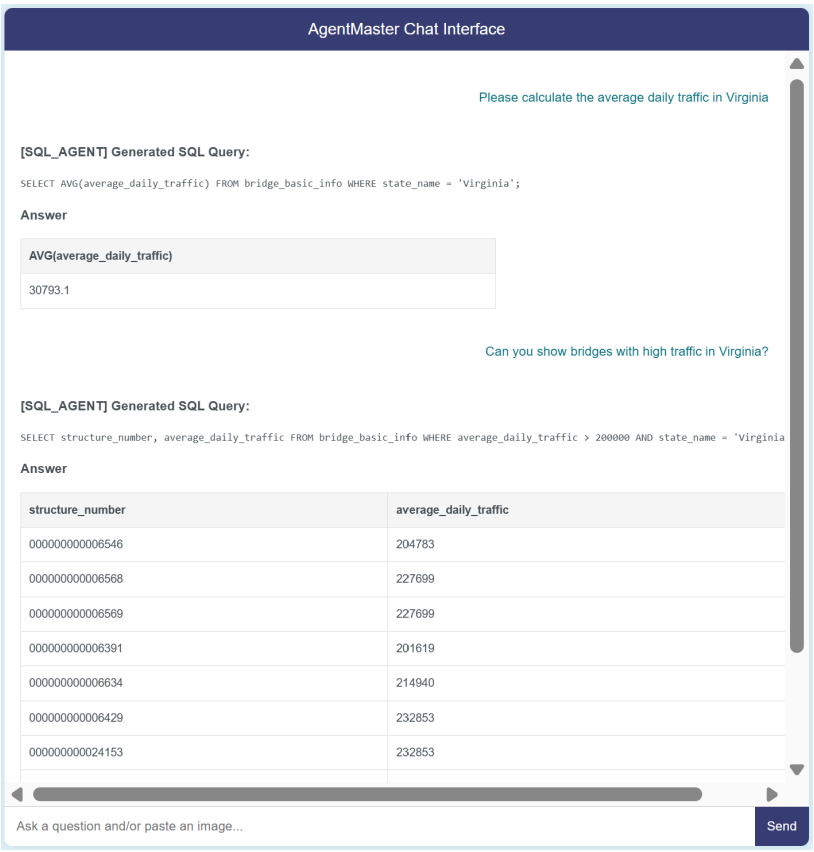


(b) SQL agent error from a single query.

Figure 10: The highlighted text contains errors in SQL information retrieval, including duplicate content, indicating that AgentMaster may still produce erroneous results.



(a) Example of a complex query.



(b) Single queries for verification.

Figure 11: An additional AgentMaster demonstration example and verification.

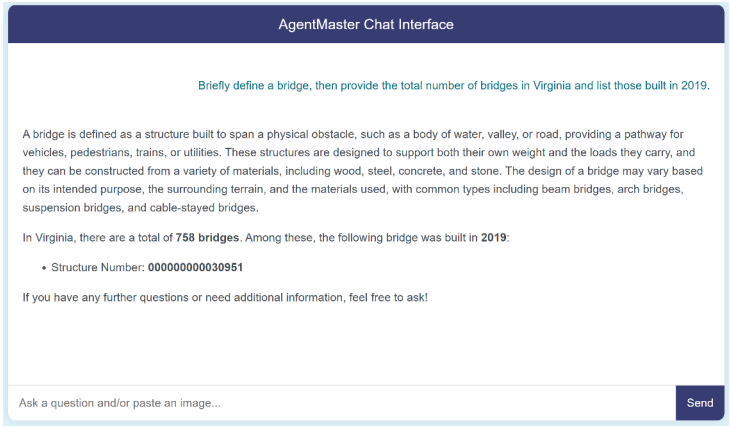
A.2 Decomposition, Allocation, and Routing of Agent Tasks for Complex Queries

A.2.1 Complex Queries

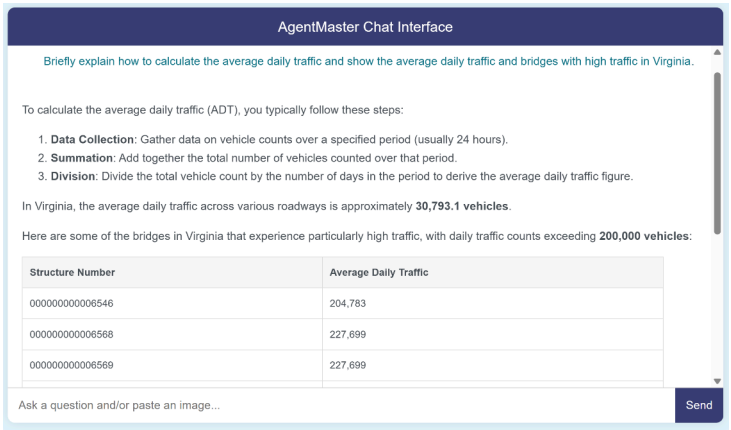
ID	Complex Query
Q1	Briefly define a bridge, then provide the total number of bridges in Virginia and list those built in 2019.
Q2	Briefly explain how to calculate the average daily traffic and show the average daily traffic and bridges with high traffic in Virginia.
Q3	What is the typical service lifespan of a bridge? Identify and display all bridges that exceed this average age.
Q4	List the three oldest bridges in Virginia, show their year built from the database, and briefly explain why their maintenance costs tend to be higher according to engineering guidelines with 50 words.
Q5	List five of the oldest bridges in the United States still in use today, and briefly describe their historical significance.
Q6	Compare the advantages and disadvantages of concrete arch bridges and steel truss bridges in terms of maintenance, lifespan, and load capacity.

Table 4: The complex questions and the corresponding IDs, matching Table 1.

A.2.2 AgentMaster Outputs of Complex Queries in the Chat Interface

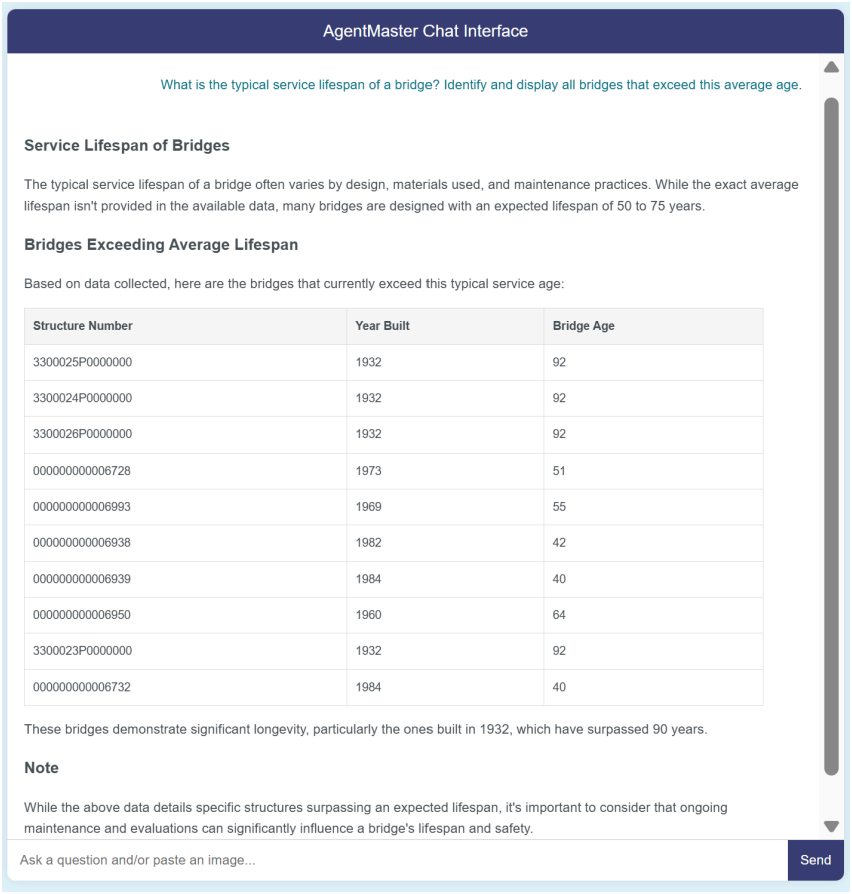


(a) Complex query Q1.

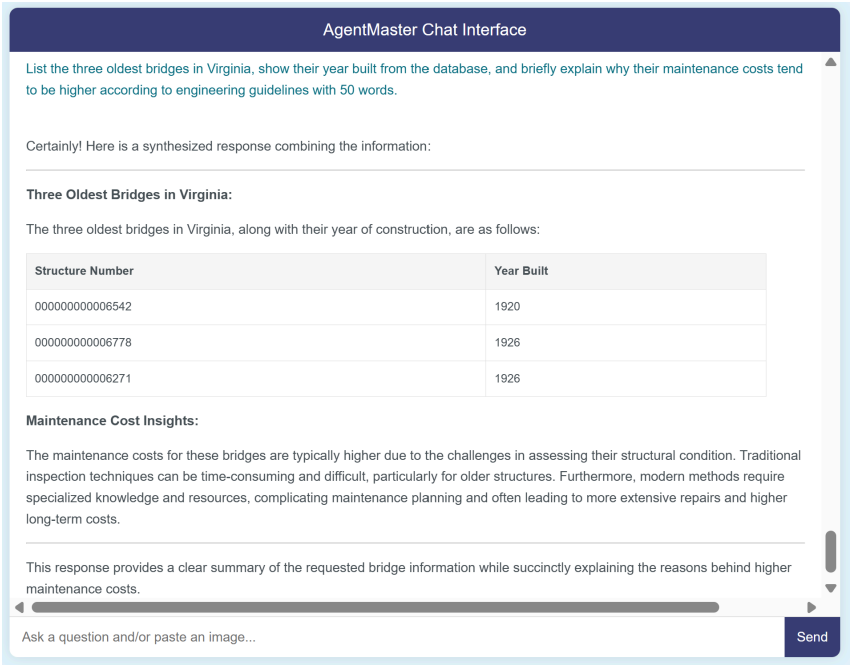


(b) Complex query Q2.

Figure 12: The display of the frontend response for complex queries Q1 and Q2.

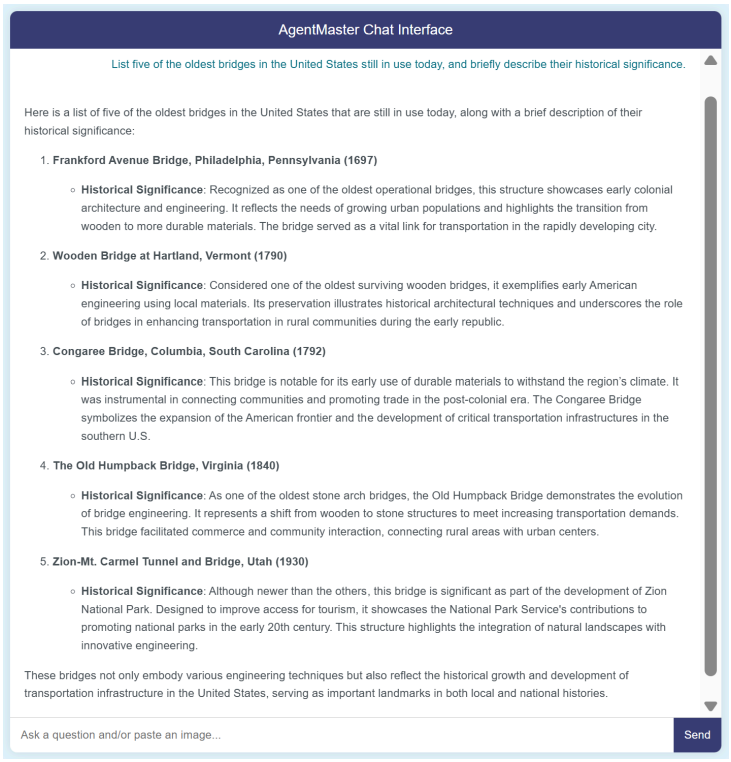


(a) Complex query Q3.

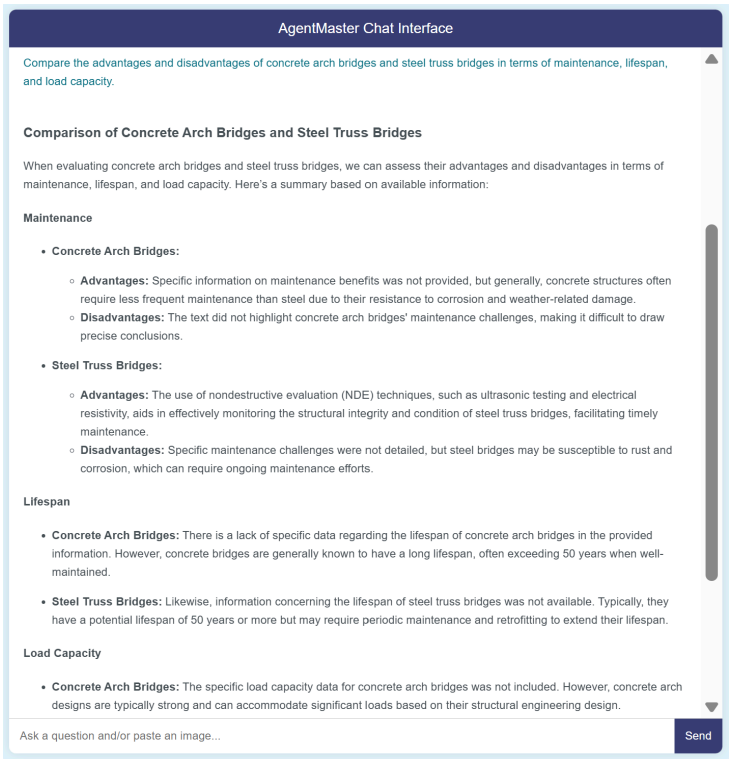


(b) Complex query Q4.

Figure 13: The display of the frontend response for complex queries Q3 and Q4.



(a) Complex query Q5.



(b) Complex query Q6.

Figure 14: The display of the frontend response for complex queries questions Q5 and Q6.

A.2.3 Automated Actions (Query Decomposition, Task Allocation, and Dynamic Routing) for Complex Queries in the Backend

ID	Information for Query Decomposition, Task Allocation, and Dynamic Routing
Q1	[Agent Server] Received complex query for coordinator: 'Briefly define a bridge, then provide the total number of bridges in Virginia and list those built in 2019.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What is a brief definition of a bridge?', 'What is the total number of bridges in Virginia?', 'Which bridges in Virginia were built in 2019?'] [Coordinator] Routing sub-question: 'What is a brief definition of a bridge?' [Complex to Single Query Router] Decision for GENERAL_AGENT [Coordinator] Routing sub-question: 'What is the total number of bridges in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT COUNT(*) FROM bridge_basic_info WHERE state_name = 'Virginia';" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'Which bridges in Virginia were built in 2019?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, year_built FROM bridge_basic_info WHERE state_name = 'Virginia' AND year_built = 2019 LIMIT 10;" Detected valid SQL, executing... [Coordinator] Synthesizing final answer...
Q2	[Agent Server] Received complex query for coordinator: 'Briefly explain how to calculate the average daily traffic and show the average daily traffic and bridges with high traffic in Virginia?' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['How do you calculate the average daily traffic?', 'What is the average daily traffic in Virginia?', 'Which bridges in Virginia have high traffic?'] [Coordinator] Routing sub-question: 'How do you calculate the average daily traffic?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the average daily traffic in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT AVG(average_daily_traffic) FROM bridge_basic_info WHERE state_name = 'Virginia';" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'Which bridges in Virginia have high traffic?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, average_daily_traffic FROM bridge_basic_info WHERE state_name = 'Virginia' AND average_daily_traffic > 200000 LIMIT 10;" Detected valid SQL, executing... [Coordinator] Synthesizing final answer...
Q3	[Agent Server] Received complex query for coordinator: 'What is the typical service lifespan of a bridge? Identify and display all bridges that exceed this average age.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What is the typical service lifespan of a bridge?', 'What is the average age of bridges?', 'What bridges exceed the typical service lifespan?', 'How can I display the bridges that exceed the average age?'] [Coordinator] Routing sub-question: 'What is the typical service lifespan of a bridge?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the average age of bridges?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: 'The average age of bridges cannot be determined with a single SQL query as it requires an aggregate function.' No SQL detected, returning text answer. [Coordinator] Routing sub-question: 'What bridges exceed the typical service lifespan?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'How can I display the bridges that exceed the average age?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: 'SELECT structure_number, year_built, bridge_age FROM bridge_basic_info WHERE bridge_age > (SELECT AVG(bridge_age) FROM bridge_basic_info) LIMIT 10;' Detected valid SQL, executing... [Coordinator] Synthesizing final answer...

Table 5: Automated actions for complex queries (Q1-Q3) in the backend.

ID	Information for Query Decomposition, Task Allocation, and Dynamic Routing
Q4	[Agent Server] Received complex query for coordinator: 'List the three oldest bridges in Virginia, show their year built from the database, and briefly explain why their maintenance costs tend to be higher according to engineering guidelines with 50 words.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What are the three oldest bridges in Virginia?', 'What is the year built for each of the three oldest bridges in Virginia?', 'Why do the maintenance costs of these bridges tend to be higher according to engineering guidelines?'] [Coordinator] Routing sub-question: 'What are the three oldest bridges in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, year_built FROM bridge_basic_info WHERE state_name = 'Virginia' ORDER BY year_built ASC LIMIT 3;" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'What is the year built for each of the three oldest bridges in Virginia?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: "SELECT structure_number, year_built FROM bridge_basic_info WHERE state_name = 'Virginia' ORDER BY year_built ASC LIMIT 3;" Detected valid SQL, executing... [Coordinator] Routing sub-question: 'Why do the maintenance costs of these bridges tend to be higher according to engineering guidelines?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Synthesizing final answer...
Q5	[Agent Server] Received complex query for coordinator: 'List five of the oldest bridges in the United States still in use today, and briefly describe their historical significance.' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What are the five oldest bridges in the United States still in use today?', 'What is the historical significance of each of the five oldest bridges in the United States?'] [Coordinator] Routing sub-question: 'What are the five oldest bridges in the United States still in use today?' [Complex to Single Query Router] Decision for SQL_AGENT Raw LLM output: 'SELECT structure_number, year_built FROM bridge_basic_info WHERE year_built < 1970 ORDER BY year_built ASC LIMIT 5;' Detected valid SQL, executing... [Coordinator] Routing sub-question: 'What is the historical significance of each of the five oldest bridges in the United States?' [Complex to Single Query Router] Decision for GENERAL_AGENT [Coordinator] Synthesizing final answer...
Q6	[Agent Server] Received complex query for coordinator: 'Compare the advantages and disadvantages of concrete arch bridges and steel truss bridges in terms of maintenance, lifespan, and load capacity' [Coordinator] Decomposing user query... [Coordinator] Decomposed into: ['What are the advantages of concrete arch bridges in terms of maintenance?', 'What are the disadvantages of concrete arch bridges in terms of maintenance?', 'What is the lifespan of concrete arch bridges?', 'What is the load capacity of concrete arch bridges?', 'What are the advantages of steel truss bridges in terms of maintenance?', 'What are the disadvantages of steel truss bridges in terms of maintenance?', 'What is the lifespan of steel truss bridges?', 'What is the load capacity of steel truss bridges?'] [Coordinator] Routing sub-question: 'What are the advantages of concrete arch bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What are the disadvantages of concrete arch bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the lifespan of concrete arch bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the load capacity of concrete arch bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What are the advantages of steel truss bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What are the disadvantages of steel truss bridges in terms of maintenance?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the lifespan of steel truss bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Routing sub-question: 'What is the load capacity of steel truss bridges?' [Complex to Single Query Router] Decision for IR_AGENT [Coordinator] Synthesizing final answer...

Table 6: Automated actions for complex queries (Q4-Q6) in the backend.

The iRead4Skills Intelligent Complexity Analyzer

Wafa Aissa[1,*], Raquel Amaro[2], David Antunes[3], Thibault Bañeras-Roux[1], Jorge Baptista[3,6], Alejandro Catala[5], Luís Correia[4], Thomas François[1], Marcos Garcia[5], Mario Izquierdo-Álvarez[5], Nuno Mamede[3,7], Vasco Martins[4], Miguel Neves[4], Eugénio Ribeiro[3,8], Sandra Rodriguez Rey[5] and **Elodie Vanzeveren[1]**
[1]UCLouvain, [2]NOVA Lisboa, [3]INESC-ID Lisboa, [4]MindShaker,
[5]Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS),
Universidade de Santiago de Compostela
[6]Faculdade de Ciências Humanas e Sociais, Universidade do Algarve
[7]Instituto Superior Técnico, Universidade de Lisboa
[8]Instituto Universitário de Lisboa (ISCTE-IUL)
Correspondence: raquelamaro@fcsh.unl.pt, marcos.garcia.gonzalez@usc.gal, jbaptis@ualg.pt, thomas.francois@uclouvain.be

Abstract

We present the iRead4Skills Intelligent Complexity Analyzer[1], an open-access platform specifically designed to assist educators and content developers in addressing the needs of low-literacy adults by analyzing and diagnosing text complexity. This multilingual system integrates a range of Natural Language Processing (NLP) components to assess input texts along multiple levels of granularity and linguistic dimensions in Portuguese, Spanish, and French. It assigns four tailored difficulty levels using state-of-the-art models, and introduces four diagnostic yardsticks—textual structure, lexicon, syntax, and semantics—offering users actionable feedback on specific dimensions of textual complexity. Each component of the system is supported by experiments comparing alternative models on manually annotated data.

1 Introduction

Reading skills are foundational to career advancement, lifelong learning, and informed participation in society. Both UNESCO[2] and OECD (2013a) define literacy as the process of engaging and understanding written texts in order to achieve one's goals and develop one's potential. Along the same lines, high literacy levels have been proven to be beneficial to other individual skills, such as problem-solving, digital literacy, and communication, showcasing its transversal aspect (OECD, 2013b). However, a minority of adults continue to experience difficulties with basic literacy skills—a minority that is nonetheless worth close attention due to the profound impact low-literacy can have on individuals' everyday lives. In fact, 20% of EU adult population exhibits low-literacy and numeracy skills (European Association for the Education of Adults, 2021). These challenges affect access to employment, healthcare, and civic participation, ultimately limiting individuals' full engagement in society (European Commission, 2021).

Reflecting the view that literacy is a continuous, lifelong process, Adult Learning (AL) programs include the improvement of literacy through vocational training (ANQEP, 2021). The training and development of reading skills rely on adequate written materials, which, unlike those available for children, are not easily accessible for adults. Consequently, selecting or designing texts suitable for low-literacy adults represents a major challenge and a highly time-consuming task for AL teachers across all fields (from the sciences to history). This underscores the need for tools that can assess and adapt text readability. With this in mind, the iRead4Skills project[3] aims to contribute to the solution of this problem by devising an automatic system that evaluates texts' complexity and suggests appropriate readings according to the user's reading skills. To do this, it is important to understand what text complexity entails for low-literacy adult native speakers and how it can be identified in written materials.

Building on a foundational assessment combining text complexity analysis, ethical governance, multilingual collaboration, and learner needs identification, the project defined four structured levels of text complexity (Very Easy, Easy, Plain +Complex) characterized by a large set of descriptors which were validated by a panel of experts. Language experts then collected and annotated multilingual datasets in Portuguese (PT), Spanish (SP), and French (FR) to train models for automatic

[*]Authors listed in alphabetical order.
[1]Accessible at demo02.iread4skills.com
[2]https://uis.unesco.org/node/3079547
[3]https://iread4skills.com/

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 73–84
November 4-9, 2025

readability assessment. These resources support the development of an Intelligent Complexity Analyzer (ICA) that integrates psycholinguistic insights with NLP techniques to predict text difficulty and provide interpretable feedback on readability.

This paper presents a user-friendly web interface for an ICA and introduces a novel classification of four diagnostic yardsticks (textual structure, lexicon, syntax, and semantics) supported by experimental analysis on newly annotated data. Designed to promote equitable access to written content, our approach embeds sustainability, dissemination, and impact evaluation to ensure lasting benefits for low-literacy adult education and digital inclusion.

Apart from this introduction, Section 2 discusses related work, Section 3 outlines the current state of the platform and its functional components, Section 4 reports on the evaluation, and Section 5 concludes with a discussion of future directions.

2 Related Work

Our study falls within the scope of readability and text complexity analysis research.

Readability: Readability research has traditionally focused on general adult readers or schoolchildren, with early formulas such as Flesch (Flesch, 1948) and Gunning Fog (Gunning, 1952) still widely used today. While some studies have applied these metrics to specialized domains like medical texts (Mcinnes and Haglund, 2011) or contracts (Arbel, 2024), they remain poorly adapted to the needs of low-literate adults. Only a few efforts have developed readability models for this population, including work in Portuguese (Aluisio et al., 2010), Italian (Dell'Orletta et al., 2011), Spanish (Saggion et al., 2015), and German (Weiss et al., 2018), with no such models yet available for French. Recent advances in readability modeling have leveraged deep learning (Nadeem and Ostendorf, 2018; Azpiazu and Pera, 2019; Martinc et al., 2021) and hybrid approaches combining linguistic features with neural architectures (Lee et al., 2021; Liu and Lee, 2023; Wilkens et al., 2024). Recently, the use of generative Large Language Models (LLMs) has also been explored (Jamet et al., 2024; Ribeiro et al., 2024, 2025; Aissa et al., 2025). In this work, we adapt readability assessment models to texts used in low-literacy adult education and evaluate their effectiveness in this context.

Text complexity analysis platforms:

A number of open access tools have been developed for assessing text complexity, though they are not tailored specifically to the needs of low-literacy adult readers (Table 1). These tools vary not only in the granularity of the readability analysis they provide to users—ranging from the document level to the segment level (sentence or paragraph) and the word level (single words or multi-word expressions)—but also in user accessibility, providing diagnoses through a graphical user interface (GUI) or via code, and in language support, which influences their suitability for multilingual contexts. In Portuguese, the CLAVIS system (Curto et al., 2015) utilizes NLP tools to extract 52 linguistic features from texts, which are then classified according to their readability levels. ALT (Moreno et al., 2022) is a web-based readability analysis tool that adapts traditional formulas to the Portuguese language and provides users with a composite readability score, lexical statistics, and visual feedback. LX-Proficiency also provides a Flesch analysis and proficiency classification for European Portuguese (Branco et al., 2014). The Coh-Metrix tool (Graesser et al., 2004) has been adapted to both Brazilian Portuguese (Scarton and Aluísio, 2010), and Spanish (Quispesaravia et al., 2016), providing cohesion and readability indices, and achieving strong classification performance between simple and complex texts. Besides, MultiAzterTest (Bengoetxea and Gonzalez-Dios, 2021) offers a multilingual approach—covering Spanish, English, and Basque—based on more than 125 linguistic features. In French, the closest system is AMesure (François et al., 2020), which focuses on standard readers of administrative texts and relies primarily on pre-deep learning techniques. More broadly, a recent system by Trokhymovych et al. (2024) introduces a multilingual readability model for Wikipedia articles in 14 languages. While their system achieves competitive results across

Platform	Granularity	GUI	PT	SP	FR
CLAVIS	D	✓	✓	✗	✗
ALT	D, S, W	✓	✓	✗	✗
LX-Proficiency	D, S, W	✓	✓	✗	✗
Coh-Metrix-Port	D, S, W	✓	✓	✗	✗
Coh-Metrix-Spa	D, S, W	✗	✗	✓	✗
MultiAzterTest	D, S, W	✓	✗	✓	✗
AMesure	D, S, W	✓	✗	✗	✓
Wikipedia system	D	✗	✓	✓	✓
iRead4Skills (ours)	D, S, W	✓	✓	✓	✓

Table 1: Overview of open-access complexity analysis platforms. D: document-level, S: Segment-level, W: word-level.

languages, its applicability to low-literacy adults is limited: it was trained exclusively on Wikipedia texts rather than on materials targeting low-literate readers, and it outputs a single difficulty score that cannot be directly mapped to our multi-level difficulty scale.

More generally, while existing platforms offer valuable insights into text complexity, they are typically designed for standard readers or language learners and do not address the specific needs of low-literacy adults, most rely either on traditional readability formulas or rich linguistic metrics. Our work contributes to this emerging area by providing an ICA tool specifically tailored to low-literacy adults, integrating linguistic features and NLP within an accessible, multilingual interface.

3 The iRead4Skills Platform

This section presents the main features of our platform, whose interface is shown in Figure 1.

3.1 Platform overview

The iReadSkills platform is a web-based tool designed to support the evaluation of text readability for adult learners with low-literacy skills. It enables educators and content developers to assess and visualize the linguistic complexity of texts in PT, SP, and FR. Users can input any text of their choice for analysis, making the platform adaptable to diverse educational contexts. It combines a Graphical User Interface (GUI) with NLP-based models, providing both an overall readability score and interpretable feedback on syntactic, lexical, semantic, and structural difficulty (top and bottom of the right-hand column in Figure 1, respectively). In addition, it offers word-level complexity annotations to help identify specific sources of reading difficulty in the text. Unlike general-purpose readability tools, iReadSkills is tailored to the needs of adult education, with an emphasis on accessibility, transparency, and language-sensitive analysis.

3.2 Functional Components

We present the interface components, focusing on the complexity classifier and yardstick evaluation, along with a brief description of the annotation module and highlighting of difficult phenomena.

3.2.1 Readability classification

The text difficulty classification component implements readability models designed to assess input texts in reference to the difficulty scale defined in the project (Monteiro et al., 2023):

Very Easy: Texts that are fully or almost fully understood by everyone, including people with very low schooling (i.e., that did not finish the primary school, ca. 6th year) and almost no reading experience. It roughly corresponds to the A1 level in the Common European Framework of Reference for Languages (CEFR) (Council of Europe, 2001).

Easy: Texts that are fully or almost fully understood by people with low schooling (i.e., that completed the primary school but no more than the 9th year) and have poor reading experience. It roughly corresponds to CEFR A2 level.

Plain: Texts that are understood the first time they are read by people that completed the 9th year and have a functional-to-average reading experience. It roughly corresponds to CEFR B1 level.

+Complex: Texts that exceed the characteristics of the previous categories. They often present significant challenges to people with low literacy.

Separate readability models are built for each target language (PT, SP, and FR), leveraging our collected datasets annotated for difficulty by experienced teacher annotators (see Section 4.1; additional details are in Appendix B.1). We explore multiple modeling strategies (see Section 4.2), including traditional Machine Learning (ML) approaches with engineered linguistic features, Deep Learning (DL) methods based on Transformer architectures, and hybrid models that integrate both feature-based and representation-based techniques.

3.2.2 Readability yardsticks

To provide users with a more fine-grained and actionable understanding of text complexity, our platform also evaluates texts along four key linguistic dimensions (see below). These yardsticks serve as diagnostic indicators, offering insight into specific aspects that contribute to the overall readability of a text. Leveraging the rich information returned by other components (see Section 3.2.3), we define the following four main yardsticks, which provide a quick and interpretable overview of the complexity of a given text:

Textual Structure: This yardstick captures surface-level properties and includes descriptive measures of the document, e.g., sentence, word length and word count.

Lexicon: Measures lexical complexity through word usage patterns, and is computed using features such as lexical complexity, lexical frequency,

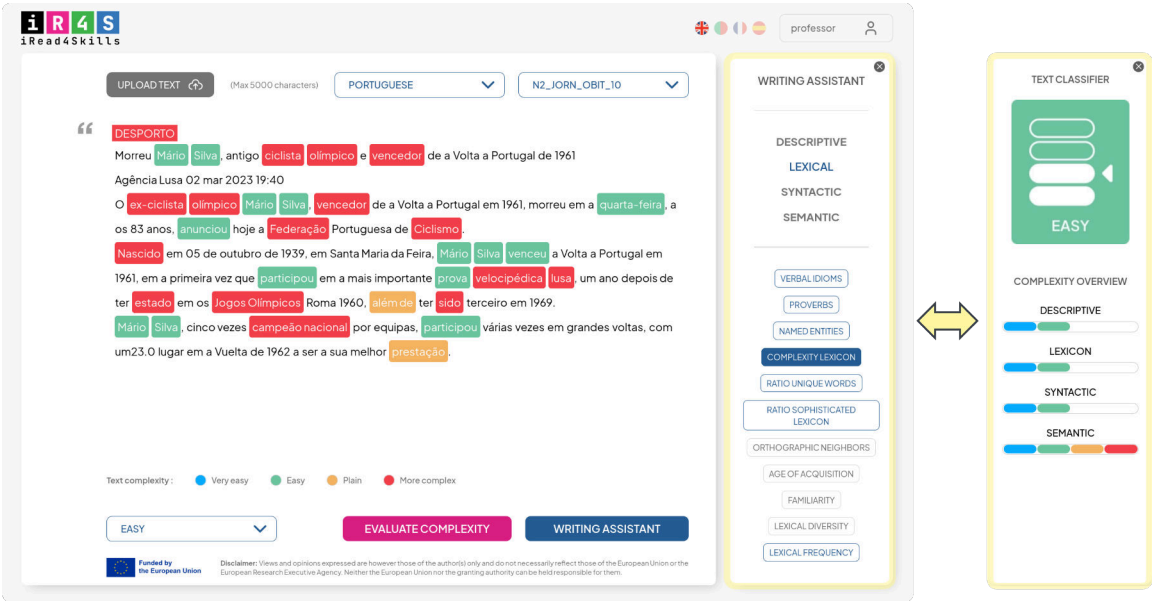


Figure 1: Interface of the iRead4Skills platform. The *Descriptive* yardstick shown is 'Textual Structure' (as defined in the paper). When using the writing assistant for in-text highlighting, the overall assessment view is replaced by the assistant's options. Both views are shown here together but correspond to different screens. The writing assistant will only be available to authorized users.

lexical diversity or age of acquisition.

Syntax: Reflects syntactic complexity, particularly the use of advanced constructions. It relies on parse depth as well as ratios of different syntactic phenomena such as subordination, coordination, auxiliary verbs, passive constructions, modifiers of noun and prepositional phrases.

Semantics: Accounts for semantic difficulty such as ambiguity or abstractness, and is measured using information from polysemy ratios and proportions of concrete or abstract words.

To compute a complexity score for each category, we compare two methods (threshold- and distribution-based) against a baseline (Section 4.3). The threshold-based approach uses predefined thresholds to assign complexity levels, with feature selection grounded in theoretical assumptions, while the distribution-based method models feature distributions using Gaussian Mixture Models (GMMs) (Reynolds, 2009) for each yardstick and complexity level.

3.2.3 Difficult phenomena annotation

This last component focuses both on identifying linguistic features that contribute to reading difficulties for low-literacy adults and highlighting specific phenomena relative to these features.

Annotator: To do this, we develop a heuristic annotator that leverages parse-based NLP models (Qi et al., 2020; Kitaev et al., 2019) to extract syntactic and semantic phenomena, alongside a curated set of readability features relevant to text complexity (see Table 5 in Appendix C). The annotation follows the BIO format (Ramshaw and Marcus, 1995), enabling structured labeling of multi-token and nested phenomena such as subordination and auxiliary verb chains. Scalar features, like sentence complexity or length, receive numerical annotations. Features are computed at multiple levels (document, sentence, and token) using metrics such as largest instance size, token coverage, and frequency relative to main verbs.

Highlighting: Following the text analysis, this module highlights linguistic phenomena identified as difficult based on the user's proficiency level. We computed statistical thresholds for each feature and complexity level using the annotated corpus (see Section 4.1) and an Interquartile Range (IQR)-based threshold ($Q3 + 1.5 \times (Q3 - Q1)$) to detect complex instances. Thresholds were applied per feature and complexity level, using upper bounds for most features (e.g., parse depth, word syllables) and lower bounds where lower values indicate complexity (e.g., familiarity, $Q1 - 1.5 \times (Q3 - Q1)$). Highlighting is triggered at document, sentence, or token level depending on the feature type. We are currently assessing, in collaboration with teachers, the most effective way to highlight difficult phenomena in text.

Language	Method	Accuracy	Adj. Acc.	Macro F_1
Portuguese	Feature-based (Random Forest)	53.89	91.36	52.17
	Fine-tuning (Albertina 100M)	51.87	92.99	50.79
	Hybrid (Prob. Distrib.)	**55.30**	**93.93**	**53.80**
Spanish	Feature-based (Decision Tree)	39.88	83.70	32.91
	Fine-tuning (BETO)	**47.47**	88.80	39.89
	Hybrid (Prob. Distrib.)	46.62	**89.88**	**43.24**
French	Feature-based (Random Forest)	62.77	98.05	47.78
	Fine-tuning (CamemBERT)	64.04	98.71	**60.36**
	Hybrid (Prob. Distrib.)	**67.32**	**99.14**	56.26

Table 2: Results of the top models in each category for the classification task across the three languages.

4 System Evaluation

We describe the experiments of classifying textual complexity and each yardstick, reporting the best results on each case.

4.1 Datasets

We evaluate the system using the iRead4Skills Dataset (Pintard et al., 2024), which includes written texts tailored to low-literacy adult readers, collected from a wide range of genres and text types. These documents were carefully selected to reflect the diversity of real-world reading materials and a subset of it was classified and validated by experts using the project's levels of reading complexity. Further details on the annotation process and dataset characteristics are in Appendix B.1.

Additionally, and as a contribution of this paper, we selected 60 documents in each target language, which were annotated by experts with the corresponding yardstick levels to evaluate this fine-grained level of readability classification. Appendix B.2 describes the annotation protocol.

4.2 Text Difficulty classification

As mentioned, we compared ML algorithms, Transformer-based DL models, and hybrid approaches for readability classification, with the best-performing models integrated into our tool.

Classical Machine Learning: To train ML models we use features from four large categories based on Wilkens et al. (2022): descriptive (length, counts and structure), lexical, syntactic, and discourse features; and also word-embeddings. We aggregate paragraph-, sentence-, and word-level features using statistical metrics like mean, max, and length. In addition to the complete set of features, we also experiment with feature selection methods to identify the most impactful features. Appendix A.1 describes the feature selection methods and the ML algorithms evaluated for this task.

Deep Learning models: We adopted the standard approach for fine-tuning transformer models[4] on our datasets. The list of models evaluated for each language can ben found in Appendix A.1.

Hybrid approaches: We implemented the soft-label architecture from Lee et al. (2021). We used our best performing fine-tuned DL models to predict classification probabilities for each difficulty class based on the input text. These probabilities were subsequently concatenated with the machine learning features to train hybrid models.

Results: To measure performance, we used common metrics in readability classification, namely Accuracy, Adjacent Accuracy, and Macro F_1. Accuracy reflects exact correctness, Adjacent Accuracy captures near-miss predictions between adjacent levels, and Macro F1 provides balanced evaluation across all classes. Table 2 presents the top-performing models across categories and languages. Overall, the hybrid approach achieves the highest performance. An exception occurs in French, where the fine-tuned model alone yields a higher F_1 score, albeit with a slightly lower precision.

Concerning generative LLMs, which are increasingly being used for readability assessment due to their flexibility and broad language capabilities (Jamet et al., 2024; Ribeiro et al., 2024), our experiments on the French and Portuguese datasets specifically designed for low-literacy adults revealed that, although LLMs show promising results, they still struggle to generalize effectively to

[4]We have also evaluated variants, such as partial fine-tuning, with similar or lower results.

texts outside their pretraining distribution and to adjust to readability scales tailored for this population. As demonstrated in our recent work (Aissa et al., 2025; Ribeiro et al., 2025), few-shot prompting improves LLM performance compared to zero-shot prompting, especially through a careful selection of examples. However, models specifically trained and fine-tuned for this readability task still outperform LLMs. For French, the top-performing LLM, DeepSeek-70B (DeepSeek-AI, 2025), achieved a macro-F1 score of 48.95%, which is below the 60.36% macro-F1 score obtained by our fine-tuned CamemBERT model. Similarly, for Portuguese, the top-performing LLM, GPT-4o mini (OpenAI et al., 2024), achieved 45.64% accuracy, 92.99% adjacent accuracy, and 45.44% macro-F1 score, all of which are below the results achieved by the hybrid model. Nonetheless, LLM capabilities for this task remain underexplored, offering an interesting avenue for future research.

4.3 Yardsticks evaluation

The targeted yardstick framework enables a more fine-grained understanding of the linguistic factors contributing to textual complexity. However, a key limitation to this analysis is that standard datasets are annotated only at the document level, and obtaining large-scale fine-grained annotations at each dimension would require significant human effort.

To overcome this challenge, we hypothesize that the correlation between each yardstick's complexity level and the corresponding features can be partially inferred from document-level annotations. Specifically, we partition the overall feature space into subspaces corresponding to each yardstick, as detailed in Section 3.2.2, and analyze each dimension separately assuming each yardstick inherits the document-level complexity annotation. Under this assumption, our aim is to identify correlations between feature distributions and complexity within each yardstick, thereby enabling dimension-specific analyses without the need for additional annotation. To validate this assumption, we rely on the documents annotated at the yardstick level (see Section 4.1 and Appendix B.2). This data allows us to assess whether the models (trained on document-level labels) can effectively capture variability in complexity across dimensions, particularly in cases where the global annotation does not fully reflect the complexity of individual yardsticks.

Methods: To assign a complexity level to each yardstick, we compare threshold- and GMM-based modeling approaches. This comparison allows us to evaluate the effectiveness of simple statistical thresholds against a probabilistic clustering technique for accurately classifying yardstick levels. Appendix A.2 describes each of the methods. As a baseline, we adopt a model in which each yardstick simply inherits the document-level complexity label, predicted by the best-performing document-level classifier available for each target language. This configuration approximates the best-case scenario for global complexity estimation in a real-world setting. If our yardstick-specific models outperform this baseline, it would indicate that the same annotated data can be used to extract more informative, dimension-specific insights, thus representing an information gain in the analysis of textual complexity.

Results: To compare the above methods, performance is evaluated using accuracy and Macro F_1; and QWK, which accounts for the ordinal nature of complexity levels and quantifies agreement between human annotations and model predictions. Results of the yardstick's classification are reported in Table 3.

For Portuguese, the threshold-based approach outperforms the baseline in predicting structure and semantic complexities, as indicated by higher accuracy, QWK scores and Macro F_1. When it comes to lexical complexity, the metrics indicate that our approach gets more labels correct, but its mistakes are more severe (e.g., predicting far-off classes). Lastly, it underperforms in syntactic complexity, likely due to difficulties in capturing the specific cues required for this dimension—an issue that corresponds to the low inter-annotator agreement (Krippendorff's α), suggesting that syntax is inherently more challenging to assess and model. Overall, performance improvements tend to occur in dimensions where annotators showed higher agreement, indicating a correlation between annotation reliability and model learnability. For Spanish, GMM-based methods consistently outperform the baseline across all dimensions, with particularly substantial gains in structure and more modest improvements in syntax, supporting the advantage of modeling each yardstick in its dedicated feature subspace. Similarly, in French, GMMs yield better results than the baseline in structure and lexical dimensions, while syntax shows mixed outcomes: the GMMs achieved higher QWK, indicating better ordinal consistency, but lower accuracy and F_1. In the semantic dimension, GMMs underperform,

	Method	Structure			Lexical			Syntax			Semantics		
		Acc	QWK	F_1	Acc	QWK	F_1	Acc	QWK	F_1	Acc	QWK	F_1
PT	baseline	0.350	0.445	0.299	0.383	**0.359**	**0.319**	**0.367**	**0.284**	**0.281**	0.417	0.408	0.382
	best (TR-based)	**0.633**	**0.702**	**0.564**	**0.467**	0.241	0.199	0.167	0.180	0.186	**0.433**	**0.424**	**0.388**
SP	baseline	0.350	0.243	0.316	0.350	0.421	0.342	0.366	0.262	0.260	-	-	-
	best (GMMs)	**0.433**	**0.616**	**0.442**	**0.416**	**0.545**	**0.407**	**0.400**	**0.337**	**0.350**	-	-	-
FR	baseline	0.467	0.530	0.363	0.400	0.568	0.363	**0.567**	0.655	**0.523**	**0.450**	**0.581**	**0.392**
	best (GMMs)	**0.550**	**0.646**	**0.476**	**0.567**	**0.661**	**0.458**	0.450	**0.686**	0.446	0.400	0.396	0.261

Table 3: Results per language and yardstick. Evaluation metrics include Accuracy (Acc), Quadratic Weighted Kappa (QWK), and Macro F_1. Semantic yardstick is omitted for Spanish due to temporary lack of external resources.

likely due to the yardstick's reliance solely on the concreteness ratio, which appears insufficient to capture semantic complexity.

5 Conclusions and future work

This paper presented the iRead4Skills ICA, a new multilingual tool specifically designed to assist educators and content developers in addressing the needs of low-literacy adults, by analyzing and diagnosing text complexity. The components build upon previous empirical findings from readability and text complexity analysis combined with current NLP techniques, and offers diagnosis through a text complexity analyzer. Crucially, the tool introduces four novel diagnostic yardsticks (textual structure, lexicon, syntax, and semantics) that provide actionable feedback on specific dimensions of textual complexity. Both the classification of texts and yardsticks were developed and validated through experimentation with annotated data. The platform is model-agnostic, allowing seamless updates with improved models over time. Currently, we are enhancing these models and conducting evaluations with teachers to identify the most effective and useful methods for highlighting difficult phenomena, aiming to support reading skill development and ensure lasting benefits for low-literacy adult education. Concretely, two main use cases will be examined: (1) evaluating complexity assessment and difficult phenomena annotation to help teachers select and design materials suitable for low-literate adults; and (2) evaluating the usefulness of readability predictions for directly recommending appropriate readings to low-literate adults. Moreover, further research is currently being conducted to extend the system beyond complexity analysis and provide writing assistance. For this, LLMs appear to be a natural choice, both for offering simplified text alternatives and for providing more general but targeted suggestions. Exploring these directions will be the focus of our next development stage. Future updates and the most recent version of the tool can be accessed at: https://iread4skills.com/.

Limitations

First, it is worth mentioning that some components of the platform are still in early stages of development; however, full functionality is expected in the coming months. While global complexity assessment is already quite mature, the yardsticks still require refinement, although they are already adding value and outperforming the baseline in most cases. Secondly, the user interface is being iteratively improved through user testing, with further changes expected to enhance usability and user experience. We are exploring the most effective way to highlight text annotations that indicate difficult phenomena to the user, so this aspect is expected to be improved. Thus, although initial experiments have shown promising results, more extensive testing is currently underway to validate and enhance the performance of the different modules. In this regard, in certain cases the data supporting the experiments is limited or it has relatively few annotations, which may affect model robustness. Finally, as commonly found in readability research, inter-annotator agreement sometimes remains low, reflecting the inherent challenges of subjective text complexity assessments.

Ethical considerations

The research activities necessary to build the machine learning models and develop the iRead4Skills webtool presented in this demo paper have followed well-established ethics high standards in

the academia. Ethics approval has been obtained from the University ethics committee, based on conformance of the protocols and user-centered research methods involved when involving stakeholders/users for either requirements elicitation and design feedback. The project started by establishing solid coordination and governance mechanisms to ensure ethical, legal, and data protection compliance, alongside effective collaboration among academic, technical, and field partners working across three target languages: Portuguese, Spanish, and French. A comprehensive needs analysis was then conducted, combining a review of existing research with newly designed surveys to identify the reading challenges faced by adult learners in vocational and lifelong education contexts.

Given the ethics guidelines for trustworthy Artificial Intelligence (AI) by the High-Level Expert Group on AI for the European Commission, some important rationale and decisions are set behind the construction of our models and tool design. Specifically, given the unique target users (low-literacy adults and their teachers), our corpora was specifically compiled and curated by linguists and experienced teachers who work with such target population. Texts are made available for research purposes. We rely on open source frameworks when necessary to ensure reliability and transparency. The interfaces are intended as supporting tools, allowing users to inspect and interpret the outputs, supporting human agency and decision making. Non deep learning models are also considered given the lower carbon footprint (as empirically measured in the constructions of our models in our internal reports). Thus, the administrator can choose the models available for each language. Finally, once the ML models are fully ethically tested in context, they will be released openly along with a statement on the scope and terms of use to ensure other developers/researchers understand their intended use, limitations and how to use them responsibly.

Acknowledgments

This work was supported by the European Commission (Project: iRead4Skills, Grant number: 1010094837, Topic: HORIZON-CL2-2022-TRANSFORMATIONS-01-07, DOI: 10.3030/101094837), by the Galician Government (ED431F 2021/01, ED431G 2023/04, and ED431B 2025/16), by a Ramón y Cajal grant (RYC2019-028473-I), and by Portuguese national funds through Fundação para a Ciência e a Tecnologia (FCT) (References: UIDB/50021/2020, DOI: 10.54499/UIDB/50021/2020 and UID/03213).

References

Wafa Aissa, Thibault Bañeras-Roux, Elodie Vanzeveren, Lingyun Gao, Rodrigo Wilkens, and Thomas François. 2025. Assessing french readability for adults with low literacy: A global and local perspective. In *The 2025 Conference on Empirical Methods in Natural Language Processing*.

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. 2010. Readability Assessment for Text Simplification. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9.

Raquel Amaro and Thomas François. 2023. iRead4Skills - Annotation Schema. Published October 30, 2023.

Raquel Amaro, Ricardo Monteiro, Thomas François, and Justine Nagant de Deuxchaisnes. 2024. iRead4Skills - Data Set 2: Annotated Corpora Report. Published November 30, 2024.

ANQEP. 2021. Referencial de competências-chave de educação e formação de adultos – nível básico. Technical report, Agência Nacional para a Qualificação e o Ensino Profissional, I.P. (ANQEP), Portugal.

Wissam Antoun, Francis Kulumba, Rian Touchent, Éric de la Clergerie, Benoît Sagot, and Djamé Seddah. 2024. CamemBERT 2.0: A Smarter French Language Model Aged to Perfection. *arXiv preprint arXiv:2411.08868*.

Yonathan A Arbel. 2024. The readability of contracts: Big data analysis. *Journal of Empirical Legal Studies*, 21(4):927–978.

Ion Madrazo Azpiazu and Maria Soledad Pera. 2019. Multiattentive recurrent neural network architecture for multilingual readability assessment. *Transactions of the Association for Computational Linguistics*, 7:421–436.

Kepa Bengoetxea and Itziar Gonzalez-Dios. 2021. Multiaztertest: a multilingual analyzer on multiple levels of language for readability assessment.

António Branco, João Rodrigues, Francisco Costa, João Silva, and Rui Vaz. 2014. Assessing automatic text classification for interactive language learning. In *International Conference on Information Society (i-Society 2014)*, pages 70–78. IEEE.

José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. 2020. Spanish Pre-Trained BERT Model and Evaluation Data. In *Proc. PML4DC at ICLR 2020*.

Council of Europe. 2001. *Common European Framework of Reference for Languages: Learning, Teaching, Assessment*. Cambridge University Press.

Pedro Curto, Nuno Mamede, and Jorge Baptista. 2015. Automatic Text Difficulty Classifier. In *Proceedings of the 7th International Conference on Computer Supported Education*, volume 1, pages 36–44.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Felice Dell'Orletta, Simonetta Montemagni, and Giulia Venturi. 2011. Read-it: Assessing readability of Italian texts with a view to text simplification. In *Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies*, pages 73–83.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Ding and Hanchuan Peng. 2003. Minimum redundancy feature selection from microarray gene expression data. In *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, pages 523–528.

European Association for the Education of Adults. 2021. Basic skills development in selected European countries: State of play report. Report, March 2021. `https://eaea.org/wp-content/uploads/2021/03/BLUESS-state-of-play_report.pdf`.

European Commission. 2021. Upskilling pathways – new opportunities for adults. Access link.

Asier Gutiérrez Fandiño, Jordi Armengol Estapé, Marc Pàmies, Joan Llop Palao, Joaquin Silveira Ocampo, Casimiro Pio Carrino, Carme Armentano Oller, Carlos Rodriguez Penagos, Aitor Gonzalez Agirre, and Marta Villegas. 2022. MarIA: Spanish Language Models. *Procesamiento del Lenguaje Natural*, 68.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233.

Thomas François, Adeline Müller, Eva Rolin, and Magali Norré. 2020. Amesure: a web platform to assist the clear writing of administrative texts. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 1–7.

Arthur Graesser, Danielle McNamara, Max Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers : a journal of the Psychonomic Society, Inc*, 36:193–202.

Robert Gunning. 1952. *The technique of clear writing*. McGraw-Hill, New York.

Henri Jamet, Maxime Manderlier, Yash Raj Shrestha, and Michalis Vlachos. 2024. Evaluation and simplification of text difficulty using LLMs in the context of recommending texts in French to facilitate language learning. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 987–992.

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.

Bruce W. Lee, Yoo Sung Jang, and Jason Lee. 2021. Pushing on text readability assessment: A transformer meets handcrafted linguistic features. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10669–10686, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fengkai Liu and John SY Lee. 2023. Hybrid models for sentence readability assessment. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 448–454.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de La Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a Tasty French Language Model. In *ACL 2020-58th Annual Meeting of the Association for Computational Linguistics*.

Matej Martinc, Senja Pollak, and Marko Robnik-Šikonja. 2021. Supervised and unsupervised neural approaches to text readability. *Computational Linguistics*, 47(1):141–179.

Nicholas Mcinnes and Bo JA Haglund. 2011. Readability of online health information: implications for health literacy. *Informatics for health and social care*, 36(4):173–189.

Ricardo Monteiro, Raquel Amaro, Susana Correia, Alice Pintard, Roser Gauchola, Michell Moutinho, and Xavier Blanco Escoda. 2023. iRead4Skills Complexity Levels. Project Deliverable D3.1, iRead4Skills.

Gleice Carvalho de Lima Moreno, Marco PM de Souza, Nelson Hein, and Adriana Kroenke Hein. 2022. ALT: um software para análise de legibilidade de textos em Língua Portuguesa. *arXiv preprint arXiv:2203.12135*.

Farah Nadeem and Mari Ostendorf. 2018. Estimating linguistic complexity for science texts. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 45–55.

OECD. 2013a. *The Survey of Adult Skills Reader's Companion*. OECD Publishing, Paris.

OECD. 2013b. *Technical Report of the Survey of Adult Skills (PIAAC)*. OECD Publishing, Paris.

OpenAI et al. 2024. GPT-4o System Card. *Computing Research Repository*, arXiv:2410.21276.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and 1 others. 2011. Scikit-learn: Machine learning in Python. *the Journal of Machine Learning Research*, 12:2825–2830.

Alice Pintard, Thomas François, Justine Nagant de Deuxchaisnes, Sílvia Barbosa, Maria Leonor Reis, Michell Moutinho, Ricardo Monteiro, Raquel Amaro, Susana Correia, Sandra Rodríguez Rey, Marcos Garcia González, Keran Mu, and Xavier Blanco Escoda. 2024. iRead4Skills Dataset 1: Corpora by Complexity Level for FR, PT and SP. Project Deliverable D3.2, iRead4Skills.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Andre Quispesaravia, Walter Perez, Marco Sobrevilla Cabezudo, and Fernando Alva-Manchego. 2016. Coh-Metrix-Esp: A complexity analysis tool for documents written in Spanish. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4694–4698, Portorož, Slovenia. European Language Resources Association (ELRA).

Lance Ramshaw and Mitch Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Proceedings of the Workshop on Very Large Corpora (VLC)*, pages 82–94.

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.

Douglas Reynolds. 2009. *Gaussian Mixture Models*, pages 659–663. Springer US, Boston, MA.

Eugénio Ribeiro, David Antunes, Nuno Mamede, and Jorge Baptista. 2025. Exploring Few-Shot Approaches to Automatic Text Complexity Assessment in European Portuguese. *Journal of the Brazilian Computer Society*, 31:690–710.

Eugénio Ribeiro, Nuno Mamede, and Jorge Baptista. 2024. Avaliação Automática do Nível de Complexidade de Textos em Português Europeu. *Linguamática*, 16(2):121–145.

João Rodrigues, Luís Gomes, João Silva, António Branco, Rodrigo Santos, Henrique Lopes Cardoso, and Tomás Osório. 2023. Advancing Neural Encoding of Portuguese with Transformer Albertina PT-*. In *Proceedings of the Portuguese Conference on Artificial Intelligence (EPIA)*, page 441–453.

Sandra Rodríguez Rey, André Bernárdez Braña, and Marcos Garcia. 2025. Exploring Linguistic Features in a New Readability Corpus for Spanish. *Procesamiento del Lenguaje Natural*, 74(0):221–239.

Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. 2015. Making It Simplext: Implementation and Evaluation of a Text Simplification System for Spanish. *ACM Trans. Access. Comput.*, 6(4).

Rodrigo Santos, João Rodrigues, Luís Gomes, João Ricardo Silva, António Branco, Henrique Lopes Cardoso, Tomás Freitas Osório, and Bernardo Leite. 2024. Fostering the Ecosystem of Open Neural Encoders for Portuguese with Albertina PT* Family. In *Proceedings of the Annual Meeting of the Special Interest Group on Under-resourced Languages (SIGUL)*, pages 105–114.

Carolina Evaristo Scarton and Sandra Maria Aluísio. 2010. Análise da inteligibilidade de textos via ferramentas de processamento de língua natural: adaptando as métricas do coh-metrix para o português. *Linguamática*, 2(1):45–61.

Mykola Trokhymovych, Indira Sen, and Martin Gerlach. 2024. An Open Multilingual System for Scoring Readability of Wikipedia. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6296–6311, Bangkok, Thailand. Association for Computational Linguistics.

Zarah Weiss, Sabrina Dittrich, and Detmar Meurers. 2018. A linguistically-informed search engine to identifiy reading material for functional illiteracy classes. In *Proceedings of the 7th Workshop on NLP for Computer Assisted Language Learning*, pages 79–90.

Rodrigo Wilkens, David Alfter, Xiaoou Wang, Alice Pintard, Anaïs Tack, Kevin P. Yancey, and Thomas François. 2022. FABRA: French aggregator-based readability assessment toolkit. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1217–1233, Marseille, France. European Language Resources Association.

Rodrigo Wilkens, Patrick Watrin, Rémi Cardon, Alice Pintard, Isabelle Gribomont, and Thomas François. 2024. Exploring hybrid approaches to readability: experiments on the complementarity between linguistic features and transformers. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2316–2331, St. Julian's, Malta. Association for Computational Linguistics.

A Technical details

A.1 Text classification

Experimental Data For Portuguese and Spanish we selected the best performing model based on test set evaluation with model selection guided by performance on validation data. For French, due to the considerably smaller size of the annotated dataset (see Table 4 in Section B.1), experiments were conducted using stratified 5-fold cross-validation with a split of 70% for training, 10% for validation, and 20% for testing.

Feature selection methods: To train ML models for readability classification we experimented, in addition to the complete set of features, with feature selection methods to identify the most impactful features, aiming to improve model accuracy and generalization for readability prediction. Specifically, we explored (i) the *minimum Redundancy Maximum Relevance* feature selection algorithm (Ding and Peng, 2003), (ii) Spearman ρ correlation between the features and the levels of the training data, (iii) k-Best, by selecting the most relevant k features according to the ANOVA F value, and (iv) the Recursive Feature Elimination approach.

ML algorithms: Regarding algorithms, we evaluate a selection of algorithms from various families, including Random Forest, SVM, Decision Trees, kNN, and Gradient Boosting. We tuned hyperparameters using grid search in each case.

Transformer models: For French, we employed variants of CamemBERT, including CamemBERT-base[5] (Martin et al., 2020), CamemBERT-v2-base[6] (Antoun et al., 2024), and a Sentence-BERT[7] (Reimers and Gurevych, 2019) model based on CamemBERT-base.

For Portuguese, we relied on the Albertina PT-PT[8] family of foundation models (Rodrigues et al., 2023; Santos et al., 2024).

For Spanish, we used a few variants of the BERT model, notably multilingual BERT[9] (Devlin et al., 2019), BERT for Spanish[10] (Cañete et al., 2020) and RoBERTa by MarIA[11] (Fandiño et al., 2022).

A.2 Yardstick analysis

Threshold-based: We use the statistical thresholds computed for different complexity levels. For each yardstick, we select relevant linguistic features and compare their values to predefined thresholds to estimate the complexity level. These levels are converted into numerical scores and aggregated to yield a final overall score for the yardstick. We identified the most informative features and aggregation methods through experiments with various theoretically motivated combinations, evaluated against a baseline. This approach balances linguistic theory with data-driven insights from annotated corpora.

Gaussian Mixture Models (GMMs): We adopt a probabilistic modeling approach by fitting a GMM using the Scikit-learn[12] library (Pedregosa et al., 2011) for each complexity level across the different yardsticks. Each model is trained under the assumption that the global document-level label can be projected onto all dimensions. Additionally, to accommodate documents of varying lengths, features at both the sentence and the token level are aggregated at the document level by using statistical descriptors such as mean, standard deviation, skewness, etc., resulting in fixed-size feature vectors suitable for modeling.

B Data and Annotation Protocol

B.1 Datasets

Table 4 presents the size of each dataset by language. For Portuguese and Spanish (Rodríguez Rey et al., 2025), all documents (2,933 and 2,563 respectively) were annotated by linguists, with a

[5] https://huggingface.co/almanach/camembert-base

[6] https://huggingface.co/almanach/camembertv2-base

[7] https://huggingface.co/dangvantuan/sentence-camembert-large

[8] https://huggingface.co/collections/PORTULAN/albertina-66a39cf7e2460605f3f1a9c2

[9] https://huggingface.co/google-bert/bert-base-multilingual-cased

[10] https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased

[11] https://huggingface.co/PlanTL-GOB-ES/roberta-base-bne

[12] https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

subset (428 and 406) further validated by additional language experts and used as the test set (for Spanish, only documents with majority votes were selected). For French, only the validated subset (461 documents) was annotated. For detailed information on the annotation process, including the annotation guidelines and decisions used for classifying text complexity, we refer to the iRead4Skills annotation schema (Amaro and François, 2023). The annotated corpora report (Amaro et al., 2024) provides concrete examples of labeled texts, inter-annotator agreement statistics, and coverage across multiple languages, illustrating the application of the schema in practice.

	Corpus	Validated	Train	Dev	Test
PT	2933	428	1986	519	428
SP	2563	406	1765	442	356
FR	2200	461	322	46	93

Table 4: Dataset splits by language: number of texts per corpus and dataset.

B.2 Yardsticks Annotation Protocol

All three languages used comparable descriptors for each complexity level as defined in the iRead4Skills Complexity Levels (Monteiro et al., 2023).

Portuguese: We selected 60 texts: 40 chosen at random (10 from each level) and 20 (5) selected based on the highest predicted probability of belonging to the target level. Three professional linguists, each with extensive experience in language proficiency assessment and language teaching, were then asked to read the texts carefully and assign a difficulty level to each yardstick according to descriptors developed within the project. Krippendorff's alpha (ordinal) was used to calculate inter-annotator agreement for each yardstick. The agreement scores were as follows: Structure: 0.650, Lexical: 0.495, Syntax: 0.320, Semantics: 0.397.

Spanish: We randomly selected 60 texts (15 for each level), and asked a professional linguist to read it carefully, assess it, and assign a level for each yardstick and text, based on the detailed descriptors of the project. Overall, the annotation process was smooth, with the annotator encountering no significant difficulties in assigning yardstick levels. For future work, we plan to engage additional annotators to facilitate the assessment of inter-annotator agreement.

French: We randomly selected 60 texts (15 per level) and hired two linguistics students, who were compensated, to perform the annotation. Prior to the annotation process, annotators were briefed on the project objectives and task specifications, and were provided with an annotation guide. A follow-up meeting was held after completion of the first set to address questions and identify potential difficulties; no major issues were reported. Additionally, a member of the project team served as a third annotator. The time required to annotate each set ranged from fifteen minutes to one hour. We used Krippendorff's alpha (ordinal) to assess inter-annotator agreement. The agreement scores were as follows: Structure: 0.422, Lexical: 0.398, Syntax: 0.497, and Semantics: 0.420.

C Annotated features

Table 5 displays the features implemented for each language. Features are based on those described by Wilkens et al. (2022).

feature	**PT**	**SP**	**FR**
Auxiliary verbs	✓	✓	✓
Passive construction	✓	✓	✓
Subordination	✓	✓	✓
Coordination	✓	✓	✓
Clitic pronouns	✓	✓	✓
NP/PP modifiers	✓	✓	✓
Depth of parse tree	✓	✓	✓
Support-verb constructions	✓	✗	✗
Causative operator-verb	✓	✗	✗
Vocative	✓	✗	✓
Echo complements	✓	✗	✗
Verbal idioms	✓	✗	✗
Proverbs	✓	✗	✗
Named Entities	✓	✓	✓
Complexity	✓	✓	✓
Number of sentences	✓	✓	✓
Sentence length	✓	✓	✓
Word length	✓	✓	✓
Word length (in syllables)	✓	✗	✓
Ratio Hapax	✓	✓	✓
Ratio of surface forms [P0-P75]	✓	✓	✗
Ratio of lemmas [P0-P75]	✓	✓	✗
Ratio Sophisticated Words	✓	✗	✓
Orthographic Neighbors (Nb.)	✓	✗	✓
Orthographic Neighbors (Cum. Freq.)	✓	✗	✓
Age of acquisition	✓	✓	✓
Word familiarity	✓	✓	✓
Ratio of abstract words	✓	✗	✗
Ratio of concrete words	✓	✗	✓
Ratio of Polysemous Words	✓	✗	✗
Nb. of words before verb	✓	✓	✓
Nb. of words after verb	✓	✓	✓
Dialogue quote	✓	✗	✗
Lexical diversity (MATTR)	✗	✗	✓
Lexical frequency	✗	✓	✓

Table 5: Summary of the implemented features per language.

AIPOM: Agent-aware Interactive Planning for Multi-Agent Systems

Hannah Kim, Kushan Mitra, Chen Shen, Dan Zhang, Estevam Hruschka
Megagon Labs
{hannah, kushan, chen_s, dan_z, estevam}@megagon.ai

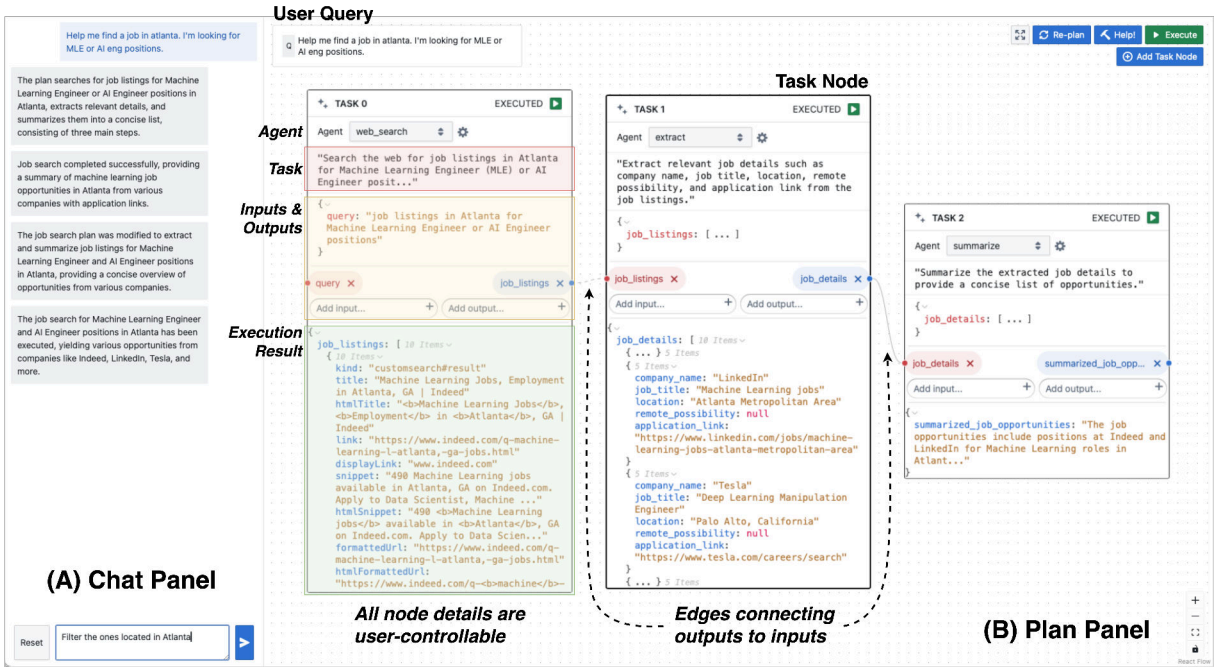


Figure 1: AIPOM enables transparent and controllable planning in multi-agent workflows through conversational and graphical interfaces that support human–LLM collaboration. **(A) The Chat Panel** allows users to define or update the planning goal, provide high-level feedback, and receive updates or explanations. **(B) The Plan Panel** displays the generated plan as an editable graph, enabling users to directly manipulate task nodes, agent assignments, data flow, and execution outputs.

Abstract

Large language models (LLMs) are being increasingly used for planning in orchestrated multi-agent systems. However, existing LLM-based approaches often fall short of human expectations and, critically, lack effective mechanisms for users to inspect, understand, and control their behaviors. These limitations call for enhanced transparency, controllability, and human oversight. To address this, we introduce AIPOM, a system supporting human-in-the-loop planning through conversational and graph-based interfaces. AIPOM enables users to transparently inspect, refine, and collaboratively guide LLM-generated plans, significantly enhancing user control and trust in multi-agent workflows. Our code and demo video are available at https://github.com/megagonlabs/aipom.

1 Introduction

Orchestrated Multi-Agent Systems (OMAS) have emerged as a powerful framework for handling complex tasks across diverse domains (Kandogan et al., 2024; Zaharia et al., 2024). These systems consist of multiple specialized agents, each responsible for performing specific subtasks upon request. The agents are systematically orchestrated, with their outputs propagating through successive agents to collaboratively resolve a given task. Recently, these modular workflows have been enhanced by the integration of large language models (LLMs), external tools, and domain-specific models, leading to improved performance and adaptability in tackling complex, real-world tasks (Schick et al., 2023; Chen et al., 2024).

A key component of OMAS is planning, i.e.,

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 85–96
November 4-9, 2025

the process of breaking down high-level goals into structured sequences of subtasks and assigning them to appropriate agents. LLMs are increasingly being used for planning (Huang et al., 2022; Wang et al., 2023b; Singh et al., 2023), owing to their ability to perform complex reasoning, generalize across domains, leverage world knowledge, reflect on their own planning decisions, and operate directly through natural language (Renze and Guven, 2024; Zhang et al., 2025a). These capabilities make LLMs well-suited for orchestrating multi-agent interactions without task-specific training.

Despite these strengths, LLM-based planning presents several challenges. First, in domain-specific or high-stakes scenarios, LLMs may generate outputs that are inaccurate, incomplete, or misaligned with expert knowledge (Valmeekam et al., 2023; Huang et al., 2024). Second, in many OMAS settings, users are presented only with the final output of the system, without visibility into the underlying plan structure or the intermediate outputs produced by agents. This lack of transparency makes it difficult to understand, verify, and trust the system's behavior. Finally, these systems are typically accessed through chat interfaces, which offer limited controllability and make it difficult for users to inspect, refine, or debug plans at a granular level. These limitations make human oversight not only necessary but central to the planning phase, underscoring the need for interfaces that allow users to actively engage with and guide the planning and execution processes to ensure outcomes align with their intentions (Union, 2024).

To address these challenges, we present AIPOM (Agent-aware Interactive Planning for Orchestrated Multi-agent systems), a novel system that enhances transparency and controllability in OMAS through human-in-the-loop planning. AIPOM combines natural language interaction with a graph-based interface that represents plans as editable workflows in a visual programming environment. Through direct manipulation (Shneiderman, 1983), users can inspect and modify the plan structure–including agent assignments, data flow, and execution order–by interacting directly with nodes and edges in the plan graph. Additionally, users can invoke LLM assistance to suggest completions, resolve issues, or fill in missing details. This mixed-initiative (Horvitz, 1999) model enables flexible, collaborative planning, combining human insight and expertise with LLM-driven reasoning to iteratively build and refine executable plans. Our contributions are as follows:

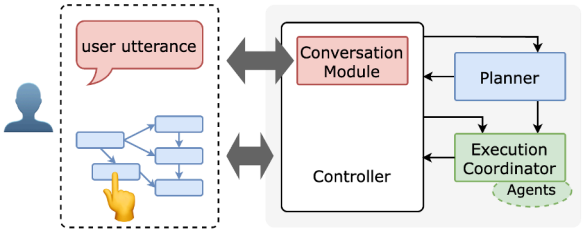


Figure 2: System overview. AIPOM supports human-in-the-loop planning through natural language interaction and direct manipulation on a plan graph.

- AIPOM, a novel system combining conversational and graph interfaces, providing fine-grained plan exploration and control.
- A mixed-initiative planning approach enabling human-LLM collaboration for plan construction and refinement for OMAS.
- Experiments and a pilot study demonstrating how AIPOM improves transparency and controllability in LLM-based planning.

2 AIPOM System

2.1 System Overview

AIPOM consists of four key modules (Fig. 2): a planning module responsible for converting user request into logical plans (§ 2.2), a conversation module that interprets user utterances and extract intent, an execution coordinator that manages subtask dispatch across agents, and a controller that orchestrates communication between them.

Users interact with AIPOM through a dual-panel interface that combines a chat panel (§ 2.3.2) for natural language interaction and a plan panel (§ 2.3.1) for exploring and editing the plan iteratively. The controller translates user inputs (both natural language feedback and graph edits) into system-level operations that update the plan and coordinate execution.

Implementation details are listed in Appendix A.

Plan Model A plan is a structured workflow of subtasks and dependencies, represented as a directed acyclic graph (DAG) (Zhuo et al., 2024; Zhang et al., 2025b). Each node in the graph corresponds to a subtask assigned to an agent, specifying its task description, assigned agent, expected inputs, and outputs. Edges define data dependencies from outputs from one node to inputs of another, thereby establishing execution order and information flow.

This plan representation differs from some prior work, which models plans as node-level DAGs

without explicit data mappings or as linear sequences of subtask descriptions. In contrast, our setting requires *coordinating multiple external agents with defined input/output interfaces*, making it essential to *track how outputs of one step connect to inputs of the next*. The DAG structure supports this fine-grained dependency modeling and enables reliable multi-agent execution.

Agents Agents (which can be LLM-based, built on top of proprietary models or APIs, or rely on simple tools and function calling) available to the system are described in an agent registry, which defines their names, capabilities, and input/output specifications. This registry serves as a shared source of truth for both the planner and the execution coordinator.

2.2 LLM-based Planner

AIPOM uses an LLM to generate and refine plans in an *agent-aware* manner. The planner constructs plans based on agent capabilities and input/output requirements defined in the agent registry.

2.2.1 Plan Generation

Plan generation is triggered whenever the conversation module identifies a new user query, representing the user's latest intent. This query is passed to the planner along with the agent registry. The LLM planner is prompted to generate a structured, executable plan that decomposes the user's goal into subtasks, assigns each subtask to an appropriate agent, and defines dependencies between them.

2.2.2 Plan Refinement via User Feedback

After a plan is generated, users can refine it either through natural language (NL) feedback or through direct manipulation on plan graphs.

1. **NL Feedback** Users can provide textual feedback. The planner is then re-prompted with the current plan state, the agent registry, and the user's feedback to produce an updated plan.
2. **Direct Manipulation** Alternatively, users can directly edit the plan graph by adding or deleting nodes or edges, modifying task descriptions, reassigning agents, adjusting input/output fields, or updating agent configurations. These changes are immediately reflected in the plan.
3. **LLM Fix** Users may invoke LLM assistance after making partial edits, prompting the planner to complete, validate, or fix the current plan.

We posit that NL feedback is well-suited for high-level guidance, such as shaping the overall structure or intent of the plan. In contrast, direct manipulation is more effective for precise or localized adjustments where users aim to retain most of the existing plan. This *mixed-initiative* workflow supports flexible and efficient human-LLM collaboration, leveraging the complementary strengths of NL interaction and structured editing.

2.3 Interface

AIPOM provide a dual-panel interface that supports both natural language interaction and direct manipulation of a structured plan. This layout enables users to switch fluidly between conversational input and direct edit, supporting a mixed-initiative workflow for human-LLM collaborative planning.

2.3.1 Plan Panel

The plan panel (Fig. 1(B)) displays the generated plan as a directed graph, with the current user query shown in the top-left corner. The plan is visualized as a node-link diagram, where each node represents a task and edges represent data dependencies.

Each node is rendered as a card containing subtask details, including the assigned agent, task description, input/output fields, and execution status. Once a task is executed, its output is shown at the bottom of the node card. Edges are rendered as directional arrows connecting output fields of one node to input fields of another, making data flow across the plan explicit. A green button inside each node card triggers single node execution.

The plan is fully editable via direct manipulation. Users can add new nodes using the "Add Node" button and create edges by dragging from an output to a compatible input. Nodes and edges can be removed by selecting and pressing the delete key. Subtask details (e.g., task descriptions, assigned agents, agent configurations, and input/output variables) can be modified directly within each node card. Task nodes can also be re-positioned freely to improve plan layout. Additionally, intermediate outputs can be manually edited without modifying the plan structure, allowing downstream subtasks to be re-executed with custom inputs.

Control buttons in the top-right corner allow users to execute the entire plan (Execute All), generate a new plan for the current query (Re-plan), or request LLM assistance to complete or fix the current plan (Help).

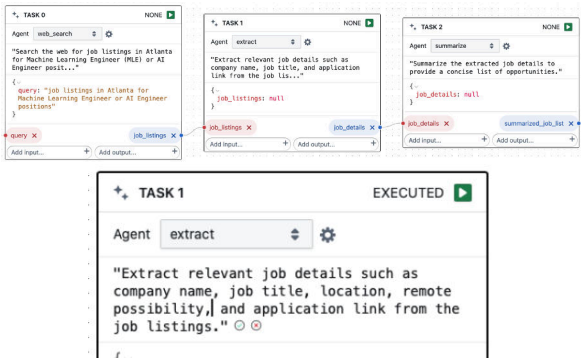


Figure 3: Initial plan generated for job search example (top) and editing task description (bottom).

2.3.2 Chat Panel

The chat panel (Fig. 1(A)) provides a conversational interface where users interact with the system using natural language. It supports a range of high-level inputs, such as initializing a new plan, modifying the current query, refining an existing plan, or triggering execution.

User messages and system responses are displayed as chat bubbles, forming a clear and traceable interaction history. When a new plan is generated, an execution is triggered, or a plan is refined, the system not only updates the plan panel but also responds with natural language explanations in the chat panel. This conversational interface complements the plan panel by enabling users to steering the planning process using high-level language, while simultaneously observing plan updates and execution results in context.

3 Usage Scenarios

3.1 Searching for a Job

Misty is seeking MLE or AI engineering roles in Atlanta. She begins with a query: "Help me find a job in Atlanta. I'm looking for MLE or AI eng positions." AIPOM responds with a three-step plan using a web search agent, an extract agent, and a summarization agent (Fig. 3, top). Misty executes the plan. Intermediate outputs appear in each node, allowing her to observe they contribute to the final answer. When she notices that the search agent returns only five postings, she adjust the agent settings to return 10 results and re-execute the plan. Next, Misty edits the extract agent's task to include location and remote possibility (Fig. 3, bottom), then re-runs only the modified node and its dependents. Noticing some jobs are outside Atlanta (e.g., Tesla, Palo Alto, is included in Node 1's execution result), she provides feedback via chat: "Filter out jobs that are not in Atlanta" (Fig. 1). AIPOM updates the plan by inserting a filtering step. After re-running the updated plan, Misty is satisfied with the results and proceeds to explore the application links. This scenario highlights AIPOM 's support for iterative refinement, transparent execution, and granular control.

3.2 Solving a Math Problem

Brock tries solve a math word problem involving full-priced and discounted glasses (see screenshots in Appendix D). The initial plan fails to compute the number of each type, leading to an incomplete solution. To fix this, Brock adds a placeholder node with a natural language description and two expected outputs, then clicks the Help button to invoke LLM assistance (Fig. 5, top). During execution of fixed plan, a multiply node produces an incorrect result: multiplying the cost per glass by 60 instead of interpreting it as 60%. Brock replaces the node with an LLM-based multiply agent to handle the percentage correctly and adds a missing edge to fix a data dependency (Fig. 5, middle). After these edits, the updated plan successfully solves the problem (Fig. 5, bottom). This example shows how AIPOM supports plan repair, agent substitution, and user-driven debugging in a mixed-initiative way.

4 Evaluation

We conduct a quantitative experiment to evaluate plan refinement performance, alongside a pilot study that compares different plan representation formats and feedback modalities.

4.1 Experiment Setting

Datasets and Tasks Our experiments utilize two datasets focused on math reasoning: GSM8K (grade-school-level word problems, Cobbe et al., 2021) and Multi-Step Arithmetic from BIG-Bench Hard (complex equation-format problems, Suzgun et al., 2022). We select math problems because their solutions have limited variability in the correct plan structure, unlike other tasks that may have multiple correct approaches involving different sets of agents, making them easier to evaluate.

For each dataset, we randomly sample 50 tasks and manually generate a correct plan p_1, which is then validated by the authors. We then *artificially modify* each correct plan by randomly applying one

Model	Dataset	Feedback	Add Node			Remove Node			Add Edge			Remove Edge			Modify (Agent)			Modify (I/O)		
			Acc	ISO	GED	Acc	ISO	GED	Acc	ISO	GED	Acc	ISO	GED	Acc	ISO	GED	Acc	ISO	GED
GPT-4o	GSM8K	**Detailed**	70.97	96.77	0.05	96.77	93.50	0.26	93.55	100	0.00	93.55	100	0.00	95.16	100	0.00	98.38	93.54	0.19
		Vague	67.74	82.25	0.73	90.32	88.71	0.44	95.16	95.16	0.08	83.9	91.9	0.13	93.5	87.1	0.52	95.16	90.3	0.35
		DM + Fix	**93.54**	90.32	0.31	100	100	0	100	100	0	100	100	0	100	100	0	**96.77**	91.94	0.29
	Multi-step	**Detailed**	19.6	95.2	0.07	52	68.4	0.79	81.2	100	0.00	80.8	99.6	0.01	74	93.2	0.07	60	82.4	0.21
		Vague	6.4	27.2	1.83	41.6	53.2	1.71	74	96.4	0.07	79.6	97.6	0.02	54	65.6	1.58	38.4	88	0.16
		DM + Fix	**11.2**	43.8	4.42	100	100	0	100	100	0	100	100	0	100	100	0	**37.2**	50.6	4.28
Llama-3.3-70B	GSM8K	**Detailed**	72.58	90.32	0.58	51.61	95.16	0.13	75.81	95.16	0.10	74.19	93.55	0.18	74.19	93.55	0.12	71.77	92.74	0.29
		Vague	64.51	85.48	0.53	50.0	61.29	1.29	74.19	91.94	0.19	72.58	88.71	0.39	70.97	82.26	1.08	66.94	83.87	0.85
		DM + Fix	**77.05**	65.00	1.87	100	100	0	100	100	0	100	100	0	100	100	0	**90.32**	83.33	0.83
	Multi-step	**Detailed**	5.60	8.50	8.33	19.60	72.43	0.59	64.40	85.20	1.14	65.60	88.28	0.80	37.6	74.13	2.08	33.0	66.60	2.62
		Vague	9.20	13.97	5.69	0.40	1.61	8.63	59.60	83.60	1.32	18.0	24.24	6.69	33.10	71.06	2.59	13.30	20.36	10.44
		DM + Fix	**12.6**	23.41	6.69	100	100	0	100	100	0	100	100	0	100	100	0	**26.87**	25.20	7.02

Table 1: Plan refinement performance across operation types for different feedback formats and models. Metrics include execution accuracy (Acc ↑), isomorphic subgraph match (ISO ↑), and graph edit distance (GED ↓). Highlighted are the **DM+Fix** and baseline Detailed Feedback performance for complex operations.

operation (e.g., adding or removing a node or edge, or altering a subtask specification) to produce an incorrect version p_0. We assess the planner's ability to refine p_0 back to the correct plan p_1 using three kinds of feedback formats: detailed natural language feedback, vague/underspecified natural language feedback, and partial manipulation with LLM assistance. After the planner generates a refined plan p_1', we compare it to the original correct plan p_1. The list of modification operations and example feedbacks are included in Appendix B.

Metrics We evaluate refined plans using the execution accuracy of refined plans and graph similarity to the original correct plans. These metrics capture functional correctness (whether the task is solved) and structural correctness (alignment with the original plan). For graph similarity, we employ: (1) isomorphism (ISO), which measures whether the graphs are structurally identical with matching agent assignments; (2) graph edit distance (GED), the minimum number of edit operations required to transform one graph into the other.

4.2 Experiment Results

Table 1 compares the effectiveness of feedback formats across plan refinement operations using the GPT-4o and Llama-3.3-70B-Instruct models.

Compared to vague NL feedback, detailed NL feedback achieves higher performance across nearly all refinement operations, confirming that precise and explicit instructions enable the LLM planner to reliably recover correct plans. However, this assumes that users are both able and willing to articulate details, which can impose cognitive burden, especially in complex or unfamiliar domains. Vague NL feedback, by contrast, is less effective: its ambiguity limits the planner's ability to accurately infer users' refinement intent. These results highlight that while natural language interactions are useful, their effectiveness depends heavily on the specificity of user input. As a result, they cannot be solely relied upon for plan refinement, especially when user intent is implicit, ambiguous, or difficult to express in language.

Direct manipulation with LLM assistance (DM+Fix) offers a practical alternative, allowing users to make partial edits on plan while relying on the LLM to complete and correct the plan. For simple, single-step operations (e.g., remove node, add or remove edge, and modify agent assignment), direct manipulation alone achieves near-perfect accuracy. For more complex operations that involve multiple interdependent changes (e.g., adding a new node and connecting its dependencies), DM+Fix outperforms vague feedback and performs comparably to detailed feedback, while requiring less user effort.

We also observe lower performance on the Multi-Step Arithmetic dataset due to the complexity of its generated plans. Multi-step plans require intricate output-input dependencies between nodes. Modification operations can easily disrupt these links, making accurate refinement challenging.

Overall, GPT-4o consistently outperforms Llama-3.3-70B, often by a significant margin. However, both models exhibit similar performance trends, indicating comparable behavior despite differences in absolute metrics.

4.3 Pilot Study

We conducted a small-scale pilot study to explore how users provide feedback to refine LLM-generated plans. The study compared plan repre-

Phase (Plan→ Feedback)	Completion Time (sec)	Word Count	Interaction Count	False Acceptance	Post-Feedback Accuracy
① Text→❶ Text	173.72	18.09	-	22.22%	80.56%
② Graph→❶ Text	149.98	12.39	-	11.11%	86.11%
② Graph→❷ DM	155.37	-	2.16	0%	88.89%

Table 2: User study results across phases where participants were presented with plans (textual or graph) and provided feedback (text or direct manipulation of graph). Average task completion time (seconds), textual feedback word count, direct manipulation interaction count, false acceptance rate, and post-feedback accuracy are reported.

sentation formats (① text vs. ② graph) and feedback modalities (❶ textual comments vs. ❷ partial graph edits). Participants were presented with flawed plans and provided feedback across three phases combining these conditions. Detailed study design and results are provided in Appendix C.

Table 2 shows that execution accuracy of refined plans is higher when the original plan is presented as a graph rather than text, and when feedback is given via direct manipulation (followed by LLM fix assistance) rather than natural language. Additionally, participants completed tasks faster and provided more concise textual feedback when plans were presented in graph format compared to textual plans. Participants also accepted incorrect plans as correct twice as often when working with textual plans. These findings align with survey results in which users found graph presentations easier to understand and debug and preferred graph editing over textual feedback (see Appendix C.2). These results highlight the benefits of our graph visualization for transparency and interpretability over conventional chat-based agentic systems.

5 Related Works

Multi-Agent Systems Our work builds on recent trends in multiple specialized agents AI systems, referred to as multi-agent systems (MAS), compound AI, agentic workflows, AI pipelines, etc. While traditional MAS emphasize agent autonomy, cooperation, and distributed decision-making (Wooldridge, 2009; Stone and Veloso, 2000), our focus is on a class of *centrally orchestrated* systems that coordinate pre-defined agents, i.e., each implementing a modular function or service. In this *orchestrated* MAS setting, agents are not proactive or autonomous; instead, they execute assigned tasks upon request. This design aligns with recent notions of compound AI (Kandogan et al., 2024) and agentic workflows (Qiao et al., 2025).

LLM-Based Planning LLM-based planning has gained popularity due to language models' ability to reason step-by-step and decompose tasks without domain-specific training (Valmeekam et al., 2023; Huang et al., 2024). Many systems interleave planning and execution, generating and executing one step at a time based on observed outcomes (Yao et al., 2023; Schick et al., 2023; Prasad et al., 2023; Wang et al., 2023a). This paradigm enables flexible adaptation but lacks a global, inspectable plan structure. In contrast, our system adopts a plan-then-execute approach: it generates a complete multi-agent plan upfront, enabling granular inspection and refinement by humans.

Interactive AI Workflow Systems Our system shares common goals with LLM chains / ML workflow systems (Wu et al., 2022; Cheng et al., 2024; Arawjo et al., 2024; Lin and Martelaro, 2024), which offer visual programming interfaces for assembling modular components into executable chains or ML pipelines. While these systems are conceptually similar to plans in OMAS, they typically require users to manually construct plans from scratch or rely on predefined templates.

InstructPipe (Zhou et al., 2025), ChainBuddy (Zhang and Arawjo, 2024), and Low-code LLM (Cai et al., 2024) use LLMs to generate structured pipelines/workflows from NL descriptions. While our interface shares similarities in combining NL with visual interactions, our work targets OMAS, where planning must be agent-aware, with explicit data flow across agents. Unlike systems focused on initial generation, AIPOM supports mixed-initiative refinement, allowing users to collaboratively build and update plans.

6 Conclusion

We presented AIPOM, a system addressing key limitations in current LLM-based planning for orchestrated multi-agent systems. By combining conversational and graph-based interfaces, AIPOM enhances transparency and controllability through flexible human-in-the-loop collaboration. Preliminary results demonstrate its effectiveness in inter-

active plan refinement.

Future work includes applying AIPOM to high-stakes domains like healthcare and finance, where precise and controllable planning by domain experts is essential. We plan to expand user interactions beyond basic graph edits to support operations such as freezing, merging, splitting, replacing tasks, and enforcing structural constraints (e.g., "A must precede B, but not coincide with C"). To improve scalability, we aim to enhance LLM assistance in verifying plans and executions, enabling users to focus on ambiguous or problematic areas (Sung et al., 2025). Finally, real-world deployments and user studies will help us assess which interactions users prefer, how they impact trust, and how to further refine the system for practical use.

Ethics Statement

We promote the collaboration of LLM-based planners and humans, which can be beneficial for various tasks. It is important to take note of the responsible use of such systems. Over-reliance on LLM-based planners may expose inherent biases present within such models which could influence decision-making in real-world scenarios. Furthermore, it is important for humans to be accountable of their actions, that is, bad actors may exploit such systems to refine plans to suit their own benefits and biases.

We also emphasize the need for privacy and data protection. If the planner handles personal or sensitive data, human intervention may introduce privacy risks. Such issues may be mitigated by using role-based access to such systems as well as data anonymization.

As LLMs continue to be utilized for planning, it is important to do so with responsible human monitoring which ensures planning and decision-making is transparent, accountable and unbiased.

References

Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA. Association for Computing Machinery.

Yuzhe Cai, Shaoguang Mao, Wenshan Wu, Zehua Wang, Yaobo Liang, Tao Ge, Chenfei Wu, WangYou WangYou, Ting Song, Yan Xia, Nan Duan, and Furu Wei. 2024. Low-code LLM: Graphical user interface over large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pages 12–25, Mexico City, Mexico. Association for Computational Linguistics.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yu Cheng, Jieshan Chen, Qing Huang, Zhenchang Xing, Xiwei Xu, and Qinghua Lu. 2024. Prompt sapper: A llm-empowered production tool for building ai chains. *ACM Trans. Softw. Eng. Methodol.*, 33(5).

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, page 159–166, New York, NY, USA. Association for Computing Machinery.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *Preprint*, arXiv:2402.02716.

Eser Kandogan, Sajjadur Rahman, Nikita Bhutani, Dan Zhang, Rafael Li Chen, Kushan Mitra, Sairam Gurajada, Pouya Pezeshkpour, Hayate Iso, Yanlin Feng, Hannah Kim, Chen Shen, Jin Wang, and Estevam Hruschka. 2024. A blueprint architecture of compound ai systems for enterprise. *Preprint*, arXiv:2406.00584.

David Chuan-En Lin and Nikolas Martelaro. 2024. Jigsaw: Supporting designers to prototype multimodal applications by chaining ai foundation models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA. Association for Computing Machinery.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2023. Adapt: As-needed decomposition and planning with language models. *arXiv*.

Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei

Huang, and Huajun Chen. 2025. Benchmarking agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shneiderman. 1983. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530.

Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.

Yoo Yeon Sung, Hannah Kim, and Dan Zhang. 2025. Verila: A human-centered evaluation framework for interpretable verification of llm agent failures. *Preprint*, arXiv:2503.12651.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

European Union. 2024. Eu artificial intelligence act. article 14: Human oversight. https://artificialintelligenceact.eu/article/14/. Accessed: 2024-06-13.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models - a critical investigation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *Preprint*, arXiv:2305.16291.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023b. Describe, explain, plan and select: Interactive planning with LLMs enables open-world multi-task agents. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Michael Wooldridge. 2009. *An Introduction to MultiAgent Systems*, 2nd edition. Wiley Publishing.

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA. Association for Computing Machinery.

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*.

Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. 2024. The shift from models to compound ai systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/.

Jin Zhang, Flood Sung, Zhilin Yang, Yang Gao, and Chongjie Zhang. 2025a. Learning to plan before answering: Self-teaching LLMs to learn abstract plans for problem solving. In *The Thirteenth International Conference on Learning Representations*.

Jingyue Zhang and Ian Arawjo. 2024. Chainbuddy: An ai-assisted agent system for helping users set up llm pipelines. In *Adjunct Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST Adjunct '24, New York, NY, USA. Association for Computing Machinery.

Shiqi Zhang, Xinbei Ma, Zouying Cao, Zhuosheng Zhang, and Hai Zhao. 2025b. Plan-over-graph: Towards parallelable llm agent schedule. *Preprint*, arXiv:2502.14563.

Zhongyi Zhou, Jing Jin, Vrushank Phadnis, Xiuxiu Yuan, Jun Jiang, Xun Qian, Kristen Wright, Mark Sherwood, Jason Mayes, Jingtao Zhou, Yiyi Huang, Zheng Xu, Yinda Zhang, Johnny Lee, Alex Olwal, David Kim, Ram Iyengar, Na Li, and Ruofei Du. 2025. Instructpipe: Generating visual blocks pipelines with human instructions and llms. *Preprint*, arXiv:2312.09672.

Hankz Hankui Zhuo, Xin Chen, and Rong Pan. 2024. On the roles of llms in planning: Embedding llms into planning graphs. *Preprint*, arXiv:2403.00783.

A Implementation Details

AIPOM is built using a React frontend and a Python backend with FastAPI for communication. The planner and conversation module are powered by OpenAI's GPT-4o. To simulate an OMAS setting, we created specialized agents using Python functions, APIs, and LLMs, based on a curated subtask taxonomy.[1]

A.1 List of Prompts

We provide the prompts used by our planner and conversation module, with the system prompt listed at the top and the user prompt at the bottom.

Prompt used for planing

```
You are a planner responsible for creating high-level plans to solve any tasks using a set of agents.
Your goal is to break down a given task into a sequence of subtasks that, when executed correctly by the appropriate agents, will lead to the correct solution.
A plan should have at least 2 steps.

For each step in the plan:
1. Describe the subtask the agent must perform.
2. Provide a brief, self-contained description of the expected inputs and outputs. Do not include any specific values or examples.
3. Generate an instruction prompt for the agent.

Represent your plan as a graph where each node corresponds to a step, and each edge represents a dependency between two steps i.e., a step's output is used as an input for a subsequent step.
If a node requires the output from a previous node as an input, ensure it is included in the edge list.
An input variable for a node represented is a tuple, where the first item is an input description, the second item is the value of the variable if it can be predetermined without executing the plan.
If is dependent upon preceding nodes, set null. DO NOT INFER THE VALUE. DO NOT EXECUTE THE STEPS.
The output should be structured in the following JSON format:
{
'nodes': <list of JSON nodes 'id': <node id as integer>, 'name': <assigned agent name>, 'task': <instruction prompt>, 'input': <list of tuple (input var, its value)>, 'output': <list of outputs>>,
'edges': <list of JSON edges 'src_node': <source node id>, 'dest_node': <destination node id>, 'src_output': <output variable name>, 'dest_input': <input variable name>>
}

eg.
{plan demonstration examples}

Here are the available agents:
```
{agent registry}
```

For identify_operands, ensure you repeat the query in the task. Sometimes, the query may require a multiplier eg. "..twice of", divisor eg. "divide by x", percentage, in a later task. Ensure all such operations are also captured in identify_operands.
There may be multiple inputs from one node to another.
In that case, ensure you define separate edges from one node to the other.
For some agents, ensure that input order is correct, e.g., when calculating profit, revenue - cost is different from cost - revenue. so input should be [revenue, cost] order.

---

{task query}
```

Prompt used for refining plan

```
<same system prompt as planning>

---

Given the original plan, refine it according to user feedback

Original Plan:
{prev plan}

User Feedback:
{feedback}
```

Prompt used for completing/fixing plan

```
<same system prompt as planning>

---

Given a query, an initial plan will be given to you. The initial plan may be incomplete or incorrect.
Your job is to complete or fix the plan. Stay as true to the initial plan as you can.

Query:
{query}

Intial Plan:
{plan}
```

Prompt used for response generation

```
You are a natural language interface for a multi-agent system.
This system creates a plan to answer a user query and executes it using AI agents.
Your task is to explain the actions triggered by the user input and clearly communicate the system's output in a very short (max 1-2 line) response.
Do not mention anything else. Write down only plain text.

---

Case 1: Response after new plan generation
Generate a very short (max 1-2 line) response to a user query to generate a plan. The response should simply provide a high level response of what the plan does, and minor details such as number of steps.
User Query: {query}
Plan: {plan}

Case 2: Response after plan execution
Generate a very short (max 1-2 line) response to a user query to execute a plan or a single node.
The response should simply provide a high level response of the execution, and minor details such as final result.

User Query: {query}
Plan: {plan}

Case 3: Response after feedback-based refinement
Generate a very short (max 1-2 line) response to a user query to interact with a plan.
The response should simply provide a high level response of the plan which was interacted with and what change took place.

Interaction Type: {interaction}
Plan: {plan}
```

[1] While a full OMAS system would involve explicit models, tools, and predefined agents, simulating the agents in this way simplifies the setup and allows us to demonstrate the core interactive planning functionality. AIPOM can be easily incorporated with actual agents by adding them to the agent registry and wiring them into the execution flow.

Modification	Refinement Ops.	Detailed NL Feedback	Vague NL Feedback	Direct Manipultaion
Remove arbitrary node	**Add Node**	Add a `{agent}` node connecting `{prev}` to `{next}`	Add a `{agent}` node	Add `agent` node + Fix
Add arbitrary node	**Remove Node**	Remove a superfluous `{agent}` node	Remove a superfluous node	Remove a specific node
Delete arbitrary edge	**Add Edge**	Add an edge between `{source id}` and `{target id}`	Add a missing edge	Add an edge connecting nodes
Add arbitrary edge	**Remove Edge**	Remove a superfluous edge between `{source id}` and `{target id}`	Remove a superfluous edge	Remove a specific edge
Incorrect agent in node	**Modify (Agent)**	Update node `{id}` to have the correct agent	Assign a correct agent	Change the assigned agent
Random I/O change	**Modify (I/O)**	Update node `{id}` with valid inputs and outputs	Set valid inputs and outputs	Add/remove an I/O + Fix

Table 3: Modifications performed to test plan refinement capabilities, along with corresponding templates for detailed & vague natural language feedbacks and direct manipulation with LLM assistance.

B Modification and Feedback Templates

We apply single-step modifications to a correct plan p_1 to generate an incorrect plan p_0. The planner's task is to refine p_0 back to p_1 using three types of feedback, as shown in Table 3. NL feedback is generated from templates and fed to the planner, whereas direct manipulation feedback is applied via graph edits, followed by LLM-assisted fixes. Although the associated refinement operation is included in the table for clarity, it is not revealed to the planner (i.e., the planner must infer it).

C Pilot Study Details

C.1 Study Design

Participants We recruited nine participants from an industry research laboratory, comprising interns, engineers, and research scientists. All participants were proficient in English, based in the United States, held at least a graduate-level degree, and had prior experience working with LLMs. The study objectives and how their input would be used were clearly explained to the participants.

Tasks We sampled 12 math word problems from the GSM8K dataset and constructed imperfect plans for each, following Table 3. To control task difficulty, we included both easy and medium tasks: easy tasks were created by applying a single modification to a gold (reference) plan, while medium tasks involved two or more modifications.

In each task, participants were presented with a flawed plan and asked to provide feedback to improve it. Plans were shown in one of two formats: ① textual descriptions in the chat panel, or ② graph representations in the plan panel. Participants provided feedback through two modalities: ❷ natural language comments and ❶ partial graph edits, including adding or deleting nodes or edges and modifying input/output variables. Graph edits were intended as partial signals to guide the planner, rather than complete corrections.

Procedure The study followed a within-subjects design, with tasks evenly divided across three phases: (1) participants received textual plans and provided textual feedback (①→❶); (2) participants received plan graphs and provided textual feedback (②→❶); and (3) participants received plan graphs and performed partial graph edits to guide the LLM planner (②→❷). Each phase included four tasks. Both the assignment of tasks to phases and the order of phases were randomized for each participant to control for ordering effects.

Note that participants were not asked to confirm or reject the actual refinements based on their feedback; rather, their input was collected and post-processed to assess whether it led to improvements.

Post-Study Survey After completing all tasks, participants answered an exit survey comparing the two plan representation formats (text and graph) and the two feedback modalities (textual feedback and direct manipulation of the graph). The survey assessed ease of understanding and issue detection for plan representations, as well as ease of use, cognitive effort, and preferences for feedback modalities. The full list of questions are:

- The plan was easy to understand in text format.
- The plan was easy to understand in graph format.
- It was easy to detect issues or flaws in the text plan.
- It was easy to detect issues or flaws in the plan graph.
- It was easy to provide useful feedback by writing textual feedback.
- It was easy to provide useful feedback by partially editing the plan graph.
- Providing feedback by writing textual feedback required a lot of mental effort.
- Providing feedback by partially editing the plan graph required a lot of mental effort.
- I would prefer to use text feedback for future tasks.
- I would prefer to use partial graph editing for future tasks.

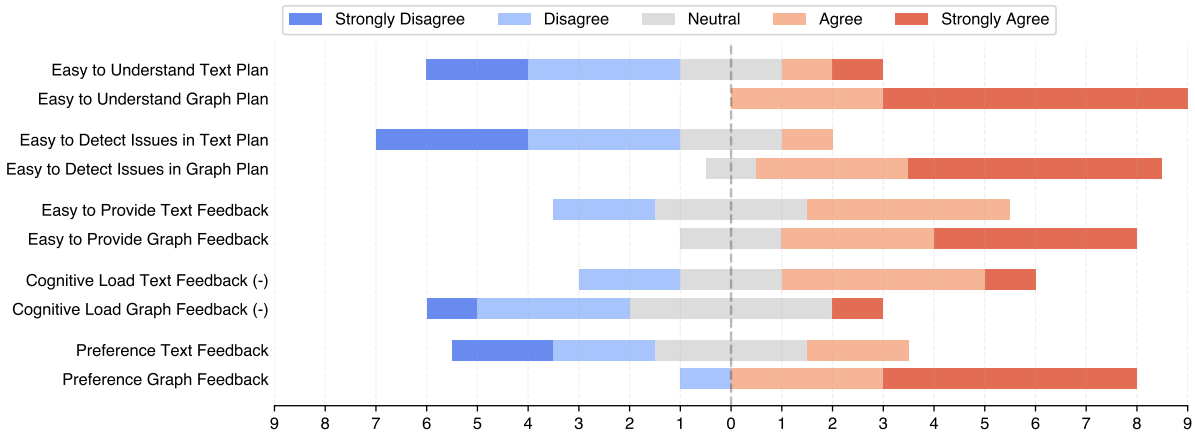


Figure 4: Participant responses from the exit survey, with each row representing 9 responses. Graph-based plan representations were perceived as significantly easier to interpret and debug than textual plans. Participants also preferred partial graph editing over textual feedback, finding it easier to provide and less mentally demanding.

C.2 Exit Survey Results

Figure 4 summarizes participants' responses from the exit survey, capturing perceived usability and preferences across plan representation formats and feedback modalities.

Plan Representations Participants found graph-based plans considerably easier to interpret and debug than textual plans. One participant noted, "*[...] if graph visualization is provided, issues like missing edges are easy to be detected immediately.*" This suggests that the visual structure of the graph helped users reason about dependencies and execution flow between agents.

Feedback Modalities Participants rated partial graph editing more favorably than textual feedback in terms of ease of use and lower mental effort. Also, participants expressed a strong preference for using graph edits in future tasks. One participant commented, "*Textual seems more helpful for high-level feedback and graph editing is more suitable for detailed editing [...].*" These responses indicate that users perceive direct manipulation as a complementary and intuitive addition to conversational feedback for guiding LLM planners.

D Additional Screenshots

In this section, we present additional screenshots of our system, captured while following the usage scenario described in § 3.2.

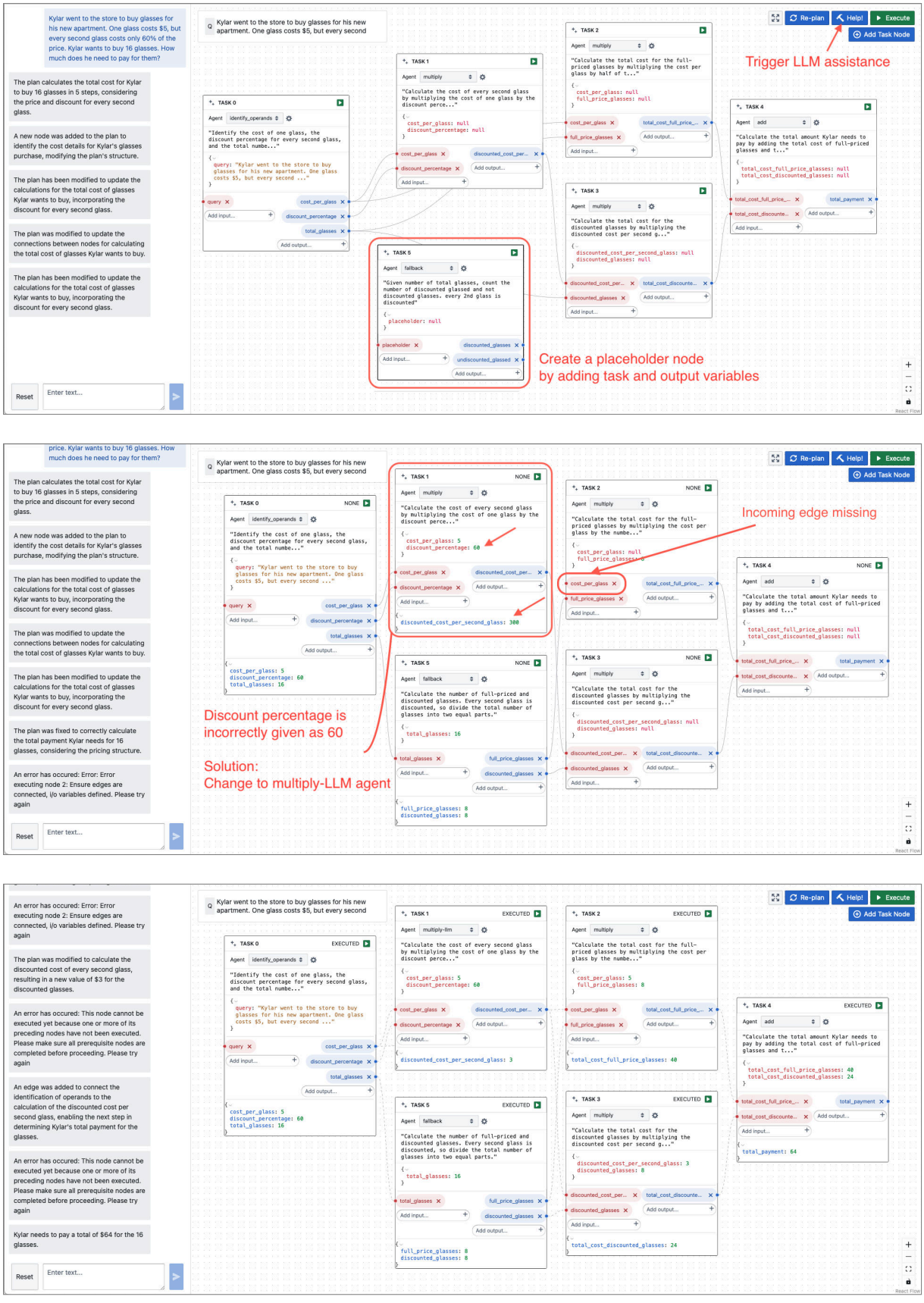


Figure 5: Screenshots providing a walkthrough of the use case scenario described in § 3.2.

LAD: LoRA-Adapted Diffusion

Ruurd J. A. Kuiper[1], Lars de Groot[2], Bram van Es[2], Maarten van Smeden[1], Ayoub Bagheri[2]

[1]University Medical Center Utrecht. [2]Utrecht University.
r.j.a.kuiper@umcutrecht.nl

Abstract

Autoregressive models dominate text generation but suffer from left-to-right decoding constraints that limit efficiency and bidirectional reasoning. Diffusion-based models offer a flexible alternative but face challenges in adapting to discrete text efficiently. We propose LAD (LoRA-Adapted Diffusion), a framework for non-autoregressive generation that adapts LLaMA models for iterative, bidirectional sequence refinement using LoRA adapters. LAD employs a structural denoising objective combining masking with text perturbations (swaps, duplications and span shifts), enabling full sequence editing during generation. We aim to demonstrate that LAD could be a viable and efficient alternative to training diffusion models from scratch, by providing both validation results as well as two interactive demos directly available online:
https://ruurdkuiper.github.io/tini-lad/
https://huggingface.co/spaces/Ruurd/tini-lad
Inference and training code:
https://github.com/RuurdKuiper/lad-code

1 Introduction

In recent years, the field of natural language generation (NLG) has been transformed by the introduction of large language models (LLMs), predominantly based on transformer models generating text using the autoregressive (AR) paradigm (Vaswani et al. 2017). While these models excel at sequential prediction, their inherent left-to-right generation process presents limitations in efficiency, flexibility, and bidirectional reasoning (Zhu and Zhao 2023; Yi et al. 2024; Zou, Kim, and Kang 2023; Nie et al. 2025). Recently, diffusion models (Sohl-Dickstein et al. 2015), initially achieving state-of-the-art performance in continuous domains like image and audio synthesis, have emerged as a promising alternative to AR models. Diffusion models operate via iterative denoising, progressively refining a noisy initial sample towards a novel sample that mimics the original distribution, offering potential advantages in parallel generation and bidirectional information transfer. Applying diffusion principles to the discrete nature of text, however, requires significant adaptation, which has led to an emerging field of active exploration within the NLP community (Zhu and Zhao 2023; Y. Li et al. 2023).

Research in text diffusion has primarily followed two trajectories: modifying the diffusion process to operate directly on discrete token spaces, often involving masking or absorbing states (He et al. 2022; Nie et al. 2024; Sahoo et al. 2024; Ye et al. 2023; Shi et al. 2024; von Rütte et al. 2025; Austin et al. 2021; Gong et al. 2024; Schiff et al. 2024; Cardei et al. 2025; Lou, Meng, and Ermon 2023; Yuan et al. 2022), or embedding discrete text into continuous latent spaces where standard Gaussian

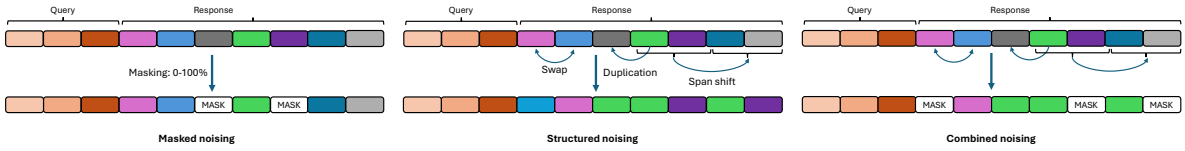


Figure 1: **The noising process applied during training.** The left illustration shows **masked** noising, where noising is applied by masking a random percentage of tokens. The middle shows **structured** noising, where tokens are randomly swapped, duplicated or spans of tokens are shifted. On the right **combined** noising is shown, where both masking and structured noising is applied, which is the noising strategy used to train all models in this study. Noising is applied only on the response tokens.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 97–110

diffusion can be applied (Zhang et al. 2023; Lovelace et al. 2022; X. L. Li et al. 2022).

Recent innovations that have improved diffusion model performance include linguistically informed masking strategies (Chen et al. 2023), simplified continuous-time objectives (Shi et al. 2024) and novel loss functions like score entropy (Lou, Meng, and Ermon 2023). Furthermore, recognizing the computational cost of training large models from scratch, recent studies have often employed existing pretrained AR models, adapting them for diffusion-based generation (Gong et al. 2024; Nie et al. 2024), demonstrating competitive performance (Nie et al. 2025). Post-training techniques like reinforcement learning are also being explored to enhance reasoning in these models (Zhao et al. 2025).

Despite recent progress, several challenges limit the adoption of text diffusion models. Training large models, either from scratch or via full-parameter tuning, is computationally expensive, making efficient adaptation strategies attractive. Inference methods like those in Masked Diffusion Models (MDMs) often fix tokens once unmasked (Sahoo et al. 2024; Schiff et al. 2024; Nie et al. 2025, 2024; Shi et al. 2024), limiting the model's ability to perform holistic refinement or recover from early errors. Efficiently using the knowledge encoded within large pretrained AR models for diffusive generation remains an open goal(Han et al. 2024).

We introduce LAD (LoRA-Adapted Diffusion), a method for non-autoregressive generation that adapts pretrained AR models—Llama 3.2 1B/3B and Llama 3.1 8B (Grattafiori et al. 2024) using bidirectional attention and lightweight Low-Rank Adaptation (Hu et al. 2021), trained using a novel structural noising scheme. LoRA enables parameter-efficient finetuning, preserving pretrained knowledge while avoiding 'catastrophic forgetting' (Luo et al. 2023; Goodfellow et al. 2013).

Additionally, instead of relying solely on mask tokens such as typical MDMs, we also apply a structured corruption process using token swaps, duplications, and span shifts. These perturbations mimic common diffusive generation errors, helping the model learn corrections and enabling updates even when no mask tokens are present. Finally, LAD is trained directly on instruction data, unifying diffusion adaptation and instruction-tuning without requiring a separate pretraining phase.

Our main contributions can thus be summarized as follows:

- A method for adapting AR LLMs to diffusion using LoRA and bidirectional attention.
- A combined structural-masking noising scheme enabling full-sequence editing.
- Unified diffusion adaptation and instruction tuning on instruction data.
- Two real-time demo interfaces exposing key generation parameters like re-noising and max steps.

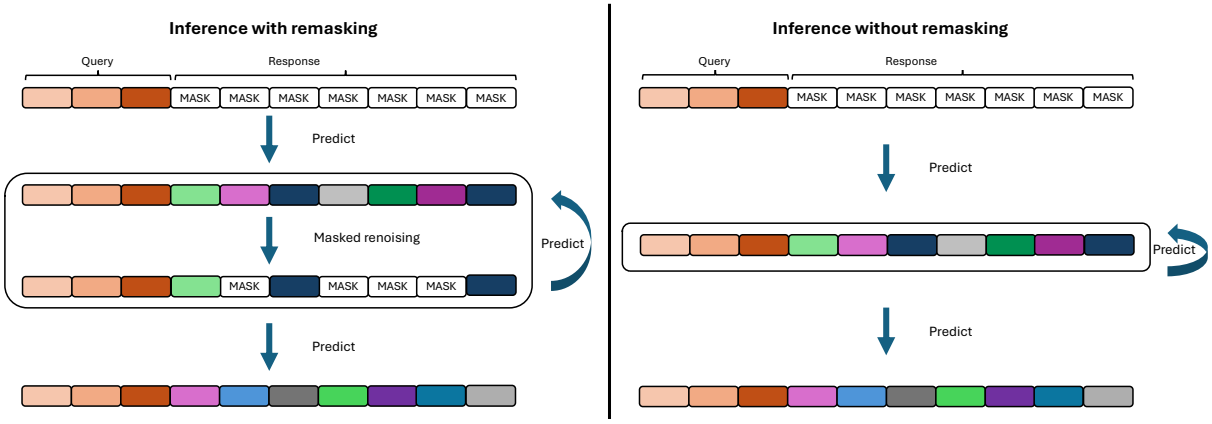


Figure 2: **The inference process.** The left column illustrates inference with intermediate remasking: tokens are predicted for each position in every iteration. After each prediction, part of the generated text is remasked. For inference without remasking, tokens are never remasked. However, the model still refines the output because it has been trained to correct structurally corrupted text as well.

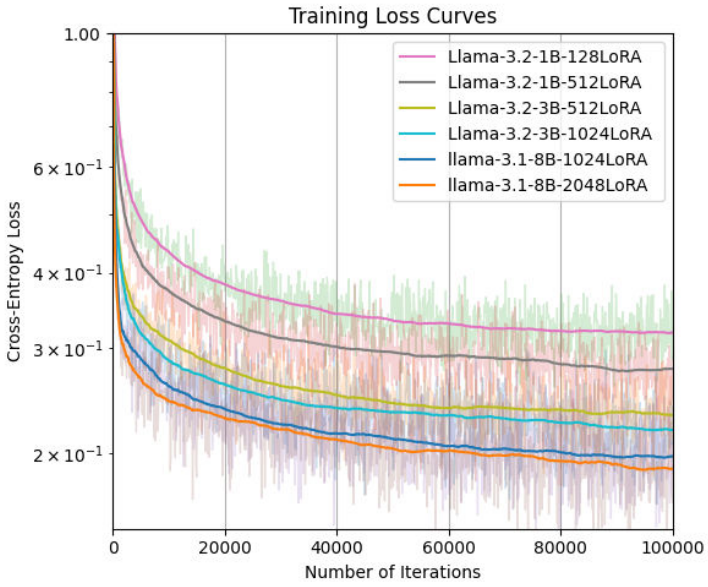


Figure 3: **Training cross-entropy loss.** The lightly colored lines show raw loss values, while the bold lines represent smoothed losses using a progressive moving average.

2 Methods

2.1 Models

All LAD models are based on Llama 3.2 1B/3B-Instruct and Llama 3.1 8B-Instruct (Grattafiori et al. 2024). To enable non-autoregressive generation, we replaced causal attention with bidirectional masks and applied LoRA fine-tuning, freezing all weights except the output layer. Following (Hu et al. 2021) we limited LoRA to the query and value projections, which capture most of its performance benefits, and set $\alpha = r$ across models. We trained six LAD variants: two 1B models (r=128, 512), two 3B models (r=512, 1024), and two 8B models (r=1024, 2048), to study scaling effects of model size and LoRA rank. Model naming and details can be found in Table 1, while detailed hyperparameters are in Appendix A.

2.2 Data

Because we used pre-trained models, our training corpus consisted solely of instruction finetuning datasets. Three general question-answering datasets were used: tatsu-lab/alpaca (Taori et al. 2023) , vicgalle/alpaca-gpt4 (Peng et al. 2023) and a filtered subset of crumb/Clean-Instruct-3M (Crumb 2023). We found that 'catastrophic forgetting' (Goodfellow et al. 2013) would occur when the model was not also trained on more specialized tasks, such as multiple-choice questions, coding, and math. Therefore, task-specific corpora were included. Only the training partition of each of these datasets was used for training. For multiple choice question answering, we used MMLU (Hendrycks et al. 2020), ARC-Easy (Clark et al. 2018) and HellaSwag (Zellers et al. 2019). For math, the GSM8K (Cobbe et al. 2021) and ORCA (Mitra et al. 2024) datasets were used. For coding, the OpenCoder (Huang et al. 2024) dataset was used. All datasets were filtered so the prompt and output together were less than 896 characters. This resulted in a combined dataset of 951k examples, which was subsequently split into training (98%), validation (1%), and test (1%) sets. All strings were tokenized, either truncated or padded with 'end-of-sequence' tokens to a length of 256, and formatted using standard Llama instruction templates. Splits were saved for consistent comparison of loss curves.

Table 1: Comparison of the trained models. The model naming reflects both the size of the base model and LoRA rank, which determines the number of trainable parameters.

Model (LAD-)	Total params	Trainable params	Trainable (%)	LoRA Rank	Training (hours)
1B-r128	1.2B	13.6M	1.1	128	2.5
1B-r512	1.2B	54.4M	4.2	512	2.8
3B-r512	3.4B	146.8M	4.4	512	7.5
3B-r1024	3.5B	293.6M	8.4	1024	8.1
8B-r1024	8.5B	436.2M	5.1	1024	16.0
8B-r2048	8.9B	872.4M	9.8	2048	16.5

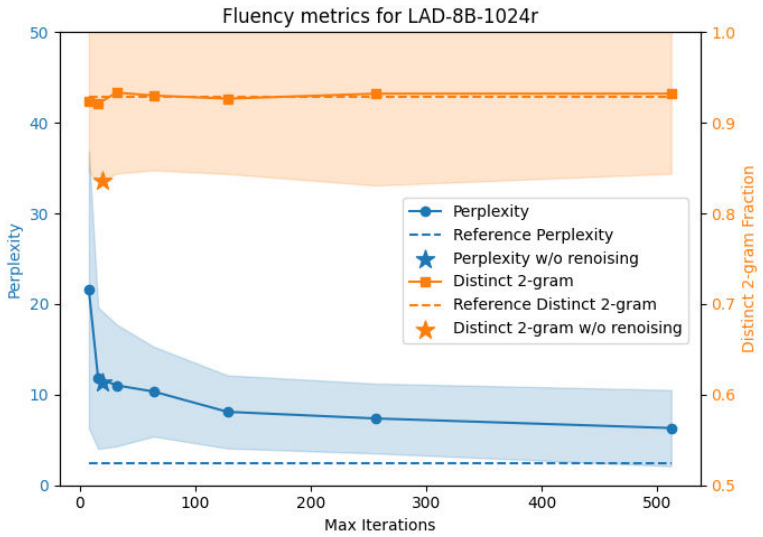


Figure 4**: Median perplexity and distinct 2-gram fraction** of LAD-8B-1024 model outputs across varying max iterations. Shaded areas show interquartile ranges. Dashed lines mark reference answer metrics. Stars show results without renoising.

2.3 Noising

When applying diffusion models to text, we found that intermediate outputs often retain useful lexical content yet suffer from structural issues, such as repeated words, misordered phrases, and fragmented sentences, that require correction. To train a model capable of identifying and correcting such errors, we introduced similar perturbations into the training data through structured noising. However, using only structured noising, there is no well-defined end-point at which a sample would be 'fully noised'. For MDMs, this is commonly represented by a fully masked sequence (Austin et al. 2021; Lou, Meng, and Ermon 2023; Nie et al. 2025; Ou et al. 2024). Therefore, we also incorporated the more regularly employed strategy of incrementally adding 'mask' tokens. The combination enables the model to denoise from an entirely masked input while iteratively refining partially coherent sequences, as illustrated in Figure 1.

Unlike formal MDMs, which use an invertible, probabilistic corruption process and derive a variational bound on the data likelihood, our process, based on local swaps, duplications, and span shifts, is non-invertible. Therefore, we do not model the reverse process in a likelihood-based framework and instead train the model directly using a standard cross-entropy loss.. More details about the training and noising schedule are provided in Appendix B.

2.4 Training

For computational efficiency, all models were trained using float16 mixed precision. The AdamW optimizer (Loshchilov and Hutter 2017) was used with an initial learning rate of 1×10^{-5}, a cosine learning rate schedule with 1000 warmup steps, and a weight decay of 0.01. The batch size was set to 8. Gradient clipping was used with a maximum norm of 0.5. All models were trained for 100k iterations using a batch size of 8 and a context window of 256 tokens to adhere to memory constraints. This means that a total of 205 million tokens were used to train each model. Training was performed on a single NVIDIA A100 GPU (40GB). Full model checkpoints were saved every 10,000 steps to inspect the training process and estimate convergence. All models were trained to minimize the cross-entropy loss between the output logits and the original, uncorrupted token sequence.

2.5 Inference

At inference time, the models can generate text using two modes: a scheduled denoising approach analogous to traditional MDMs, and a self-refining approach that relies on the model's ability to correct sequences without explicit renoising. Both generation strategies are illustrated in Figure 2.

The first method follows a conventional iterative denoising schedule. Given a prompt, the process for generating a response sequence, $\mathbf{x}$, begins with

a fully masked initial state, $\mathbf{x}^{(0)}$, of a predefined length. The model then enters a refinement loop for a maximum of N steps. At each iteration i, the model takes the current sequence $\mathbf{x}^{(i-1)}$ as input and produces a new, complete sequence prediction, $\hat{\mathbf{x}}^{(i)}$.

Following this prediction, a fraction $\eta(i)$ of the tokens in $\hat{\mathbf{x}}^{(i)}$ are randomly re-masked to produce the input for the next iteration, $\mathbf{x}^{(i)}$. This fraction is determined by an exponentially decreasing noise-schedule, $\eta(i)$. The noise schedule is defined as:

$$\eta(i) = \eta_0 \frac{e^{-s\cdot(i/N)} - e^{-s}}{1 - e^{-s}}$$

Here, s is a hyperparameter controlling the sharpness of the decay, and η_0 is the initial noise level at $i = 0$. This process continues until the final iteration N, at which point no tokens are re-masked.

The second method used the model's capability to correct text directly without masking tokens. Similar to the scheduled approach, generation begins with an initial prediction $\hat{\mathbf{x}}^{(1)}$ from a fully masked response sequence. However, in all subsequent iterations ($i > 1$), the explicit renoising step is omitted. Instead, the full, unmasked output from the previous step, $\hat{\mathbf{x}}^{(i-1)}$, is fed directly back into the model to produce the next refinement, $\hat{\mathbf{x}}^{(i)}$. In this mode, the model both identifies misplaced or incorrect tokens, and replaces these by a better fitting alternative.

It should be noted that both methods are not equivalent to standard MDM generation. Firstly, any token can be re-masked when the first method is used. Moreover, for both methods the model can amend any token in the sequence, not just those that were explicitly masked.

Lastly, both methods employ an early stopping criterion. The generation process is considered to have converged and is terminated before the preset maximum number of iterations if the output sequence remains identical for three consecutive iterations.

2.6 Evaluation

We evaluated all models on five benchmarks: ARC-Easy and ARC-Challenge (Clark et al. 2018), MMLU (Hendrycks et al. 2020), HellaSwag (Zellers et al. 2019), and GSM8K (Cobbe et al. 2021), using GPT-4o to judge the correctness of generated answers. For most benchmarks, purely diffusive generation was used, except for GSM8K, where semi-autoregressive generation was used. Inference details can be found in Appendix C. For each benchmark, 100 samples were tested per model. To assess text fluency, we computed perplexity using the larger and more recent Phi-4 14B model (Abdin et al., 2024). Output diversity was quantified using the distinct 2-gram fraction, which measures the proportion of unique consecutive word pairs in the generated text. We evaluated six diffusion-based models and three autoregressive base models, all limited to a maximum output length of 256 tokens.

Finally, we also measured the impact of the maximum number of iterations on output fluency for the LAD-8B-1024r model.

2.7 User interface

To demonstrate the test time compute flexibility of the LAD models, we developed two interfaces for

Table 2: Summary of benchmark scores (% correct) for varying model and LoRA sizes. Generative perplexity and distinct-2 gram fractions are also included. Results for the three Llama basemodels, evaluated using our own protocol, are shown for comparison.
Underline = best diffusion model; * = best model for its size; **bold** = best overall model

	Model	**LAD**						**Llama**		
	- Size	1B		3B		8B		1B	3B	8B
	- LoRA rank	128	512	512	1024	1024	2048	-	-	-
Benchmarks	ARC-Easy	56	*58	91	*93	93	93	57	91	***94**
	MMLU	29	30	43	*48	54	53	*48	*48	***65**
	ARC-Challenge	*42	35	68	74	79	***80**	40	*77	***80**
	HellaSwag	27	26	51	*58	***80**	76	*38	55	62
	GSM8K	5	4	37	43	44	45	*49	*72	***80**
Intrinsic metrics	Perplexity	17.1	14.7	11.0	10.1	9.1	9.5	*2.9	*2.8	***2.7**
	Distinct 2-grams	0.88	0.90	0.92	0.94	0.94	0.94	0.92	0.98	0.94

interactive experimentation (Figure 5). The simple interface allows a user to provide a prompt, set the maximum number of generation iterations, and toggle between the two inference methods. A pause function is included to visualize the step-by-step generation process.

The advanced interface provides more detailed control over the generation process. It allows tuning of hyperparameters for the noise schedule (initial fraction, decay sharpness), sampling strategy (top-k, top-p), and bias towards end-of-sequence tokens. This interface also enables alternative methods such as confidence-guided noising, which preferentially re-masks low-confidence tokens, and semi-autoregressive generation, where text is produced in smaller, sequential blocks.

3 Results

Exact model sizes and training durations for the diffusion models are shown in Table 1. The 1B, 3B, and 8B models were trained in approximately 3, 8, and 16 hours, respectively.

Figure 3 shows the training loss curves for all models. Lower final cross-entropy loss values are observed for models with larger parameter sizes and higher LoRA ranks.

The fluency metrics for the LAD-8B-1024r model across different numbers of diffusion iterations are shown in Figure 4. Perplexity decreases as the number of iterations increases, while the distinct 2-gram fraction remains stable and similar to that of the reference text. The figure also includes results for the same model run without intermediate re-noising. In this setting, perplexity remains comparable to the model with re-noising for the same number of iterations, while the distinct 2-gram fraction is slightly lower.

Table 2 presents benchmark scores for all models across five datasets, along with perplexity and distinct 2-gram fraction. Results are reported for diffusion models of varying size and LoRA rank, as well as for the original Llama foundation models. Across most benchmarks—ARC-Easy, MMLU, ARC-Challenge, and HellaSwag—the diffusion models perform comparable to the base models, also for similar sized smaller models. On GSM8K, however, the foundation models show clearly higher accuracy. Perplexity values are higher for all diffusion models compared to the base models, while the distinct 2-gram fractions are similar across all models.

Finally, a short quantitative and qualitative study of diffusive generation without renoising can be seen in Appendix D and E.

4 Discussion

The results demonstrate that LAD effectively adapts pretrained autoregressive LLMs into diffusion-style models with competitive performance and efficient training.

The LAD models were fine-tuned using only 100k iterations, which amounted to 200 million training tokens. For comparison, the base model LLaMA 3.1-8B (Grattafiori et al. 2024) was trained on 15 trillion tokens, and comparable diffusion models such as LlaDa (Nie et al. 2025), used 2.3 trillion tokens. This means LAD used just 0.001% and 0.008% of their respective token counts.

Model performance scales with both model size and LoRA rank, although diminishing returns were seen for large LoRA ranks. This was apparent from both the training loss curves (Figure 3) as well as the benchmark and text fluency results (Table 2). Base model size plays a dominant role in performance, validating the choice to use pre-trained frozen autoregressive backbones with lightweight bidirectional LoRA adapters.

Increasing the number of diffusion refinement iterations reduced perplexity (Figure 4), meaning that increasing ‘test time compute’ increased the performance of the model. This behavior could enable the user to make choices on whether speed or quality is more important, which is a unique feature that is not available through autoregressive generation. The diminishing returns observed beyond a certain number of iterations do suggest practical limits for speed-quality trade-off. We also showed that the model can be used without intermediate re-noising (Figure 6 and Figure 7). This strategy ensures no intermediate information generated by the model is lost, as the model has access to all information generated in the previous iteration.

Benchmark (Table 2) results further showed that LAD models achieve comparable accuracy to base autoregressive models on most datasets despite their non-autoregressive nature, although performance lags on complex reasoning tasks like GSM8K.

Finally, the interactive user interfaces (Figure 5) concretely demonstrate the flexibility enabled by LAD’s diffusion framework. By exposing control over generation iterations, noise scheduling,

sampling strategies, and novel techniques like confidence-guided noising and semi-autoregressive generation, users can tailor compute and output quality trade-offs in real time.

5 Conclusion

LAD provides an efficient and flexible framework to adapt pretrained autoregressive models for diffusion-based generation, demonstrating for the first time that LoRA finetuning alone can enable this transformation. Our unique structured noising strategy makes diffusion without denoising possible, though it does not yet outperform diffusion with renoising. While LAD matches base models on some tasks, complex reasoning challenges like GSM8K remain areas for improvement. Overall, LAD lays a foundation for scalable, controllable diffusion models, with further evaluation needed on diffusion without renoising.

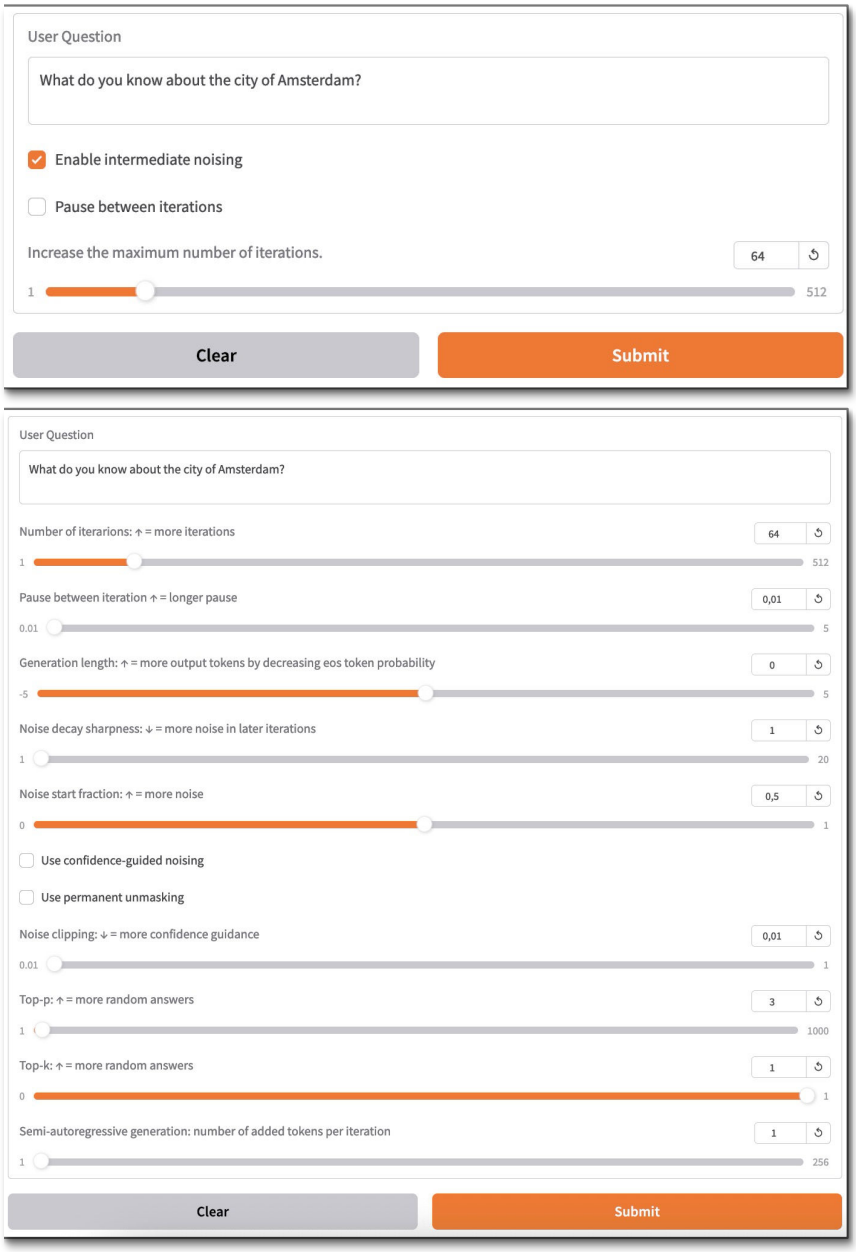


Figure 5: **Interfaces of the LAD model used for inference.** Top shows the interface of the simple demo and decide the inference method. Bottom shows the extended demo which includes a larger number of customization options

Limitations

Our work shares part of its motivation with (von Rütte et al. 2025) who enabled sequence-level correction in diffusion models by augmenting masking with discrete uniform noise (random token replacement). They noted this method, while promising, often degraded benchmark performance. They attributed this to the increased task complexity requiring larger models. We observed similar performance drops using our models, reinforcing the idea that larger models might be necessary to achieve similar results as autoregressive models. However, using the LAD training paradigm might provide a solution to this problem.

Firstly, by applying LoRA to large, pre-existing models, we enable the use of larger networks with only a fraction of the compute required for training from scratch. This parameter-efficient approach training approach could thus offer a viable approach to further test the scalability hypothesis posed by von Rütte et al. Second, we see that fewer iterations are necessary to achieve similar results when using larger models. This could offset the increased compute cost of using the larger models.

Comprehensive evaluation of large language models requires substantial effort, as there are many dimensions to consider—ranging from reasoning ability and factual accuracy to multilingual performance and robustness. While we have attempted to test key aspects of the LAD framework, a full exploration of all capabilities was beyond the scope and length constraints of this study. Nonetheless, the following limitations highlight important areas for future research.

Firstly, diffusion models uniquely enable bidirectional attention and editing, which we did not explicitly study here. Measuring bidirectional reasoning behavior is non-trivial and should be included when validating these models further, as it is a desirable feature of diffusion models.

We were also not able to compare perplexity scores between LAD and other diffusion-based text models. This was primarily due to mismatched generation settings: most diffusion models in prior work are evaluated under unconditional generation, while LAD was trained for instruction-following. Direct comparison would thus be misleading, as conditional perplexity is typically lower.

Furthermore, evaluation was limited to five benchmarks and two intrinsic metrics (perplexity and distinct-2). While these provide a useful first indication of model quality, future work should expand to a broader range of tasks, particularly ones that stress reasoning, factuality, or multilingual capabilities. This will help to better judge the strengths and limitations of diffusion-based text generation.

Likewise, we have so far tested LAD on models up to 8B parameters. While this is already on par with the largest open-source diffusion LLMs to date (Nie et al. 2024, 2025), scaling further should be possible using LoRA finetuning. However, one of our goals was to show that AR models could be adapted for diffusive generation using minimal resources. Even with our parameter-efficeint finetuning methods, models larger than 8B will require access to GPUs with more than 40GB VRAM or multi-GPU setups.

Finally, while our structural noising strategy was designed to simulate typical generation errors, it remains hand-engineered and not learned. This meant that the training samples do not necessarily reflect the intermediate samples generated during inference.

Finally, we provide interactive interfaces for tuning inference hyperparameters, which enables users to intuitively find settings that work best for them. However, a deeper exploration of optimal settings (e.g. for top-k, sharpness, or re-noising schedule) was outside the scope of this paper. These choices may have a strong effect on output quality, and merit further systematic study.

Ethics Statement

This work follows the general principles of the ACM Code of Ethics. No human or sensitive data was used, and we relied solely on publicly available datasets. Ethical risks were considered in line with standard practice for language model research.

Acknowledgments

This research was supported by the Utrecht University Focus Area Applied Data Science. We thank the ADS steering committee for their support.

References

Austin, Jacob, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. "Structured Denoising Diffusion Models in Discrete State-Spaces." *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2107.03006.

Cardei, Michael, Jacob K. Christopher, Thomas Hartvigsen, Brian R. Bartoldson, Bhavya Kailkhura, and Ferdinando Fioretto. 2025. "Constrained Language Generation with Discrete Diffusion Models." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2503.09790.

Chen, Jiaao, Aston Zhang, Mu Li, Alex Smola, and Diyi Yang. 2023. "A Cheaper and Better Diffusion Language Model with Soft-Masked Noise." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2304.04746.

Clark, Peter, 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 [Cs.AI]*. arXiv. http://arxiv.org/abs/1803.05457.

Cobbe, Karl, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, et al. 2021. "Training Verifiers to Solve Math Word Problems." *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2110.14168.

Crumb, A. I. 2023. "Clean-Instruct-3M." https://huggingface.co/datasets/crumb/Clean-Instruct-3M.

Gong, Shansan, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, et al. 2024. "Scaling Diffusion Language Models via Adaptation from Autoregressive Models." *ArXiv [Cs.CL]*. arXiv. https://github.com/HKUNLP/DiffuLLaMA.

Goodfellow, Ian J., Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. "An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks." *ArXiv [Stat.ML]*. arXiv. http://arxiv.org/abs/1312.6211.

Grattafiori, Aaron, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. 2024. "The Llama 3 Herd of Models." *ArXiv [Cs.AI]*. arXiv. http://arxiv.org/abs/2407.21783.

Han, Kehang, Kathleen Kenealy, Aditya Barua, Noah Fiedel, and Noah Constant. 2024. "Transfer Learning for Text Diffusion Models." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2401.17181.

He, Zhengfu, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. "DiffusionBERT: Improving Generative Masked Language Models with Diffusion Models." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2211.15029.

Hendrycks, Dan, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. "Measuring Massive Multitask Language Understanding." *ArXiv [Cs.CY]*. arXiv. http://arxiv.org/abs/2009.03300.

Hu, Edward J., 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 [Cs.CL]*. arXiv. http://arxiv.org/abs/2106.09685.

Huang, Siming, Tianhao Cheng, J. K. Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J. Yang, et al. 2024. "OpenCoder: The Open Cookbook for Top-Tier Code Large Language Models." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2411.04905.

Li, Xiang Lisa, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. "Diffusion-LM Improves Controllable Text Generation." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arXiv.2205.14217.

Li, Yifan, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. "Diffusion Models for Non-Autoregressive Text Generation: A Survey." *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2303.06574.

Loshchilov, Ilya, and Frank Hutter. 2017. "Decoupled Weight Decay Regularization." *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/1711.05101.

Lou, Aaron, Chenlin Meng, and Stefano Ermon. 2023. "Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution." *ArXiv [Stat.ML]*. arXiv. http://arxiv.org/abs/2310.16834.

Lovelace, Justin, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. 2022. "Latent Diffusion for Language Generation." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2212.09462.

Luo, Yun, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. "An Empirical Study of Catastrophic Forgetting in Large Language Models during Continual Fine-Tuning." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2308.08747.

Mitra, Arindam, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. "Orca-Math: Unlocking the Potential of SLMs in Grade School Math." *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2402.14830.

Nie, Shen, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2024. "Scaling up Masked Diffusion Models on Text." *ArXiv [Cs.AI]*. arXiv. http://arxiv.org/abs/2410.18514.

Nie, Shen, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-

Rong Wen, and Chongxuan Li. 2025. “Large Language Diffusion Models.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2502.09992.

Ou, Jingyang, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. 2024. “Your Absorbing Discrete Diffusion Secretly Models the Conditional Distributions of Clean Data.” *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2406.03736.

Peng, Baolin, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. “Instruction Tuning with GPT-4.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2304.03277.

Rütte, Dimitri von, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. 2025. “Generalized Interpolating Discrete Diffusion.” *ArXiv [Cs.CL]*. arXiv. https://github.com/dvruette/gidd/.

Sahoo, Subham Sekhar, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. “Simple and Effective Masked Diffusion Language Models.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2406.07524.

Schiff, Yair, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. 2024. “Simple Guidance Mechanisms for Discrete Diffusion Models.” *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2412.10193.

Shi, Jiaxin, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. 2024. “Simplified and Generalized Masked Diffusion for Discrete Data.” *ArXiv [Cs.LG]*. arXiv. http://arxiv.org/abs/2406.04329.

Sohl-Dickstein, Jascha Narain, Eric A. Weiss, Niru Maheswaranathan, and S. Ganguli. 2015. “Deep Unsupervised Learning Using Nonequilibrium Thermodynamics.” Edited by Francis Bach and David Blei. *International Conference on Machine Learning*, Proceedings of Machine Learning Research, abs/1503.03585 (March): 2256–65.

Taori, Rohan, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori Hashimoto. 2023. “Stanford Alpaca: An Instruction-Following LLaMA Model.” https://github.com/tatsu-lab/stanford_alpaca.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/1706.03762.

Ye, Jiasheng, Zaixiang Zheng, Yu Bao, Lihua Qian, and Quanquan Gu. 2023. “Diffusion Language Models Can Perform Many Tasks with Scaling and Instruction-Finetuning.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2308.12219.

Yi, Qiuhua, Xiangfan Chen, Chenwei Zhang, Zehai Zhou, Linan Zhu, and Xiangjie Kong. 2024. “Diffusion Models in Text Generation: A Survey.” *PeerJ. Computer Science* 10 (e1905): e1905.

Yuan, Hongyi, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2022. “SeqDiffuSeq: Text Diffusion with Encoder-Decoder Transformers.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2212.10325.

Zellers, Rowan, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. “HellaSwag: Can a Machine Really Finish Your Sentence?” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/1905.07830.

Zhang, Yizhe, Jiatao Gu, Zhuofeng Wu, Shuangfei Zhai, Josh Susskind, and Navdeep Jaitly. 2023. “PLANNER: Generating Diversified Paragraph via Latent Language Diffusion Model.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2306.02531.

Zhao, Siyan, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. 2025. “D1: Scaling Reasoning in Diffusion Large Language Models via Reinforcement Learning.” *ArXiv [Cs.CL]*. arXiv. http://arxiv.org/abs/2504.12216.

Zhu, Yuansong, and Yu Zhao. 2023. “Diffusion Models in NLP: A Survey.” *ArXiv [Cs.CL]*. arXiv. https://doi.org/10.48550/arxiv.2303.07576.

Zou, Hao, Zae Myung Kim, and Dongyeop Kang. 2023. “A Survey of Diffusion Models in Natural Language Processing.” *ArXiv [Cs.CL]*. arXiv. https://www.semanticscholar.org/paper/4ca824e792f3a2c777ffa5896a2e7cdf11b9518d.

A Model hyperparameters

All models were trained using the same training framework. The foundation models used were the Llama 3.2 1B- and 3B-Instruct and Llama 3.1 8B-Instruct models. For computational efficiency, all training was performed using 16-bit floating-point precision. The `AdamW` optimizer was used with a `cosine` learning rate scheduler, starting with a learning rate of `1e-5` after 100 warmup steps. We used a weight decay of 0.01 and a maximum gradient norm of 0.5. The models were trained for a single epoch with a per-device batch size of 8. All input sequences were tokenized and then padded or truncated to a fixed length of 256 tokens.

To analyze the impact of parameter-efficient fine-tuning, we evaluated several model variants by applying Low-Rank Adaptation (LoRA) to the base model. As summarized in the main text, we varied the LoRA rank (`r`) across values of 128, 512, 1024 and 2048, also depending on the model size. For all LoRA configurations, the scaling parameter `lora_alpha` was set equal to the rank, and LoRA was applied to the query (`q_proj`) and value (`v_proj`) matrices of the attention mechanism. No dropout was used in the LoRA layers. This approach allowed us to create a range of models with varying parameter counts while keeping the core architecture and training hyperparameters consistent.

B Training noising schedule

Our approach is motivated by the empirical observation that when applying diffusion-style models to text, intermediate outputs often contain relevant lexical content but exhibit structural issues, such as repeated words, incorrect word order, or sentence fragments. To train a model capable of correcting such errors, we introduce a hybrid corruption process that combines token masking with structured, non-local perturbations. Let $\mathbf{x} = (x_1, x_2, \ldots, x_L)$ represent a clean sequence of tokens from the data distribution $p_{\text{data}}(\mathbf{x})$. Our corruption process, $C(\mathbf{x})$, is a stochastic function that produces a corrupted sequence $\tilde{\mathbf{x}}$. This process is a composition of two distinct noising strategies:

- **Token masking (C_{mask}):** We employ a masking operator that replaces a fraction of tokens with a special [MASK] token. This is analogous to the forward process in Masked Diffusion Models (MDMs). Let $\mathbf{m} \in \{0,1\}^L$ be a binary mask vector sampled from a uniform random distribution, where $m_i = 1$ indicates masking. The masking process can be defined as:
$$C_{\text{mask}}(\mathbf{x}, \mathbf{m})_i = \begin{cases} [\text{MASK}] & \text{if } m_i = 1 \\ x_i & \text{if } m_i = 0 \end{cases}$$
- **Structural perturbation (C_{struct}):** We apply a set of non-invertible, structure-altering

Algorithm 1: Masked and Structured Noising for Denoising Training

Require: token sequence x = [x_1, ..., x_n], masking token MASK, noise probability p ∈ [0, 0.5], RNG

1: With 50% probability: # Masking
2: Sample mask fraction f ~ Uniform(0, 1)
3: Let m ← floor(f × n)
4: Randomly select m indices I_mask ⊆ {1, ..., n} without replacement
5: For each i ∈ I_mask, set x_i ← MASK
6: For each position i ∈ {1, ..., n−1}: # Swapping
7: With probability p/4, swap $\hat{x}_i$ and $\hat{x}$_{i+1}
8: For each position i ∈ {1, ..., n}: # Duplication
9: With probability p/4, do:
10: Sample direction d ∈ {−1, +1}
11: If i + d ∈ [1, n], set $\hat{x}_i$ ← $\hat{x}$_{i+d}
12: With probability p/4: # Span shift
13: Sample span length s ∈ {1, 2, 3}
14: Sample shift distance δ ∈ {1, ..., 4} and direction d ∈ {−1, +1}
15: Let start ∈ {1, ..., n − s + 1}
16: Let span ← x_{start : start + s − 1}
17: Let target ← clamp(start + d × δ, 1, n − s + 1)
18: Overwrite x_{target : target + s − 1} ← span
19: Return x

transformations, including local token swaps, duplications, and random shifts of token spans. These operators are designed to mimic the specific structural artifacts we observe during iterative generation.

The full corruption $\tilde{\mathbf{x}} = C(\mathbf{x})$ is a probabilistic application of these operators. This hybrid approach enables us to create a distribution of corrupted samples that trains the model to correct both content-level (via masking) and structure-level errors.

Our methodology departs from the usual formal probabilistic framework of Masked Diffusion Models (MDMs) as presented by, among others, (Shi et al. 2024; Ou et al. 2024; Nie et al. 2025; Sahoo et al. 2024). A typical MDM defines a time-indexed forward process $q(\mathbf{x}_t|\mathbf{x}_0)$ that gradually masks tokens, where $t \in [0,1]$ represents the noise level, and each token is independently masked with probability t. This process allows for the derivation of a reverse process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and an objective function $\mathcal{L}(\theta)$ that serves as a variational upper bound on the negative log-likelihood of the data:

$$-\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log p_\theta(\mathbf{x})] \leq \mathcal{L}(\theta)$$

This provides a principled, likelihood-based training framework. In contrast, our corruption process $\mathcal{C}(\mathbf{x})$ is not defined as a time-indexed, reversible Markov chain. The structural perturbations in C_{struct} are deterministic operations applied stochastically which are not directly invertible. Consequently, we cannot define a corresponding probabilistic forward model or derive a tractable variational bound on the data likelihood.

Given the nature of our corruption process, we frame the training as a direct denoising auto-encoding task rather than likelihood maximization. The model parameterized by θ, is trained to reconstruct the original sequence $\mathbf{x}$ from its corrupted version $\tilde{\mathbf{x}}$. The objective is to minimize the standard cross-entropy loss between the model's output distribution and the original clean sequence:

$$\mathcal{L}_{\text{CE}}(\theta) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\tilde{\mathbf{x}}\sim\mathcal{C}(\mathbf{x})}\left[-\sum_{i=1}^{L}\log p_\theta(x_i|\tilde{\mathbf{x}})\right]$$

While this prevents tractable likelihood evaluation, it allows the model to learn a practical denoising function. The masking component serves two critical roles within this framework. First, a fully masked input ($\tilde{\mathbf{x}}$ generated with a 100% mask rate) provides a well-defined starting point for generation from pure noise, analogous to $\mathbf{x}_1$ in formal diffusion. Second, partial masking provides stochasticity during the inference process that enables more diverse and natural responses, which complements the convergence to grammatically sound sentences offered by the structured corruptions. This combined approach trains a single model capable of both iterative refinement of flawed text and conditional generation from a corrupted prompt.

To simulate intermediate inference steps, we apply a structured noising function to each sequence of tokens. For each sequence, a noise probability $p\in[0.0,0.5]$ is sampled uniformly and used to parameterize three types of corruption. First, with 50% probability, a random fraction (between 0% and 100%) of the tokens is replaced with a special `MASK` token. Second, adjacent tokens are randomly swapped with probability $p/4$. Third, tokens are duplicated either from the previous or next position, also with probability $p/4$. Finally, with probability $p/4$, a short span of 1 to 3 tokens is copied and shifted to a new location within the sequence, moved by 1 to 4 positions in either direction. These noising operations are applied during preprocessing via a batched mapping function and are only performed on sequences of at least two tokens. The resulting corrupted input serves as the model input, while the original sequence is used as the target for supervised training. An algorithmic notation of the noising process is shown in Algorithm 1.

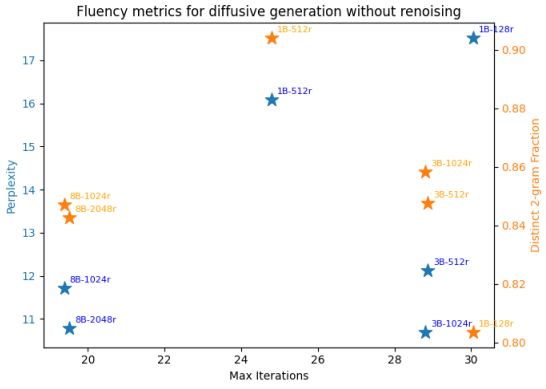


Figure 6: **Perplexity and distinct 2-gram fraction for multiple LAD models without intermediate re-noising.** Each star represents a model's performance at its average convergence iteration, shown on the x-axis.

C Evaluation details

We evaluated model performance on five standard benchmarks: ARC-Easy, MMLU, ARC-Challenge, HellaSwag, and GSM8K. For each benchmark, we sampled 100 representative examples per model. Answers were generated using a custom diffusion-based decoding process, with settings tailored to each benchmark.

For ARC-Easy, MMLU, ARC-Challenge, and HellaSwag, model responses were generated using 16 diffusion steps, with a noise schedule starting at 1.0, a maximum generation length of 256 tokens, top-k sampling set to 1, and top-p sampling set to 0.1. These settings encourage the model to refine highly noised inputs into coherent answers over a fixed number of iterations.

For GSM8K, which involves multi-step mathematical reasoning, we used a different configuration: the generation process was allowed up to 256 diffusion steps, with no initial noise (i.e., noise start = 0.0), a maximum generation length of 256 tokens, top-k set to 1, and top-p set to 1.0. In addition, we used semi-autoregressive generation, allowing the model to generate and refine 4 tokens at a time. A noising sharpness value of 1.0 was used to guide the refinement process. We followed the semi-autoregressive implementation described by (Nie et al. 2025).

Answer correctness was evaluated using GPT-4o, prompted in a zero-shot setting to determine whether each model's response correctly answered the original question. All diffusion-based models were capped at a maximum of 256 generated tokens. In addition to the six diffusion-tuned models, we also evaluated three autoregressive base models under identical constraints.

To assess fluency, we calculated perplexity using the Phi-4 14B model (Abdin et al., 2024), a large autoregressive language model trained for strong generalization and language modeling quality. We selected Phi-4 for this purpose under the assumption that stronger models produce more reliable and meaningful evaluations of fluency. Evaluating weaker or mid-sized models with a more capable evaluator helps prevent underestimation of fluency due to evaluator limitations.

To measure lexical diversity, we computed the distinct 2-gram fraction for each model output. This metric represents the proportion of unique consecutive word pairs (bigrams) in the generated

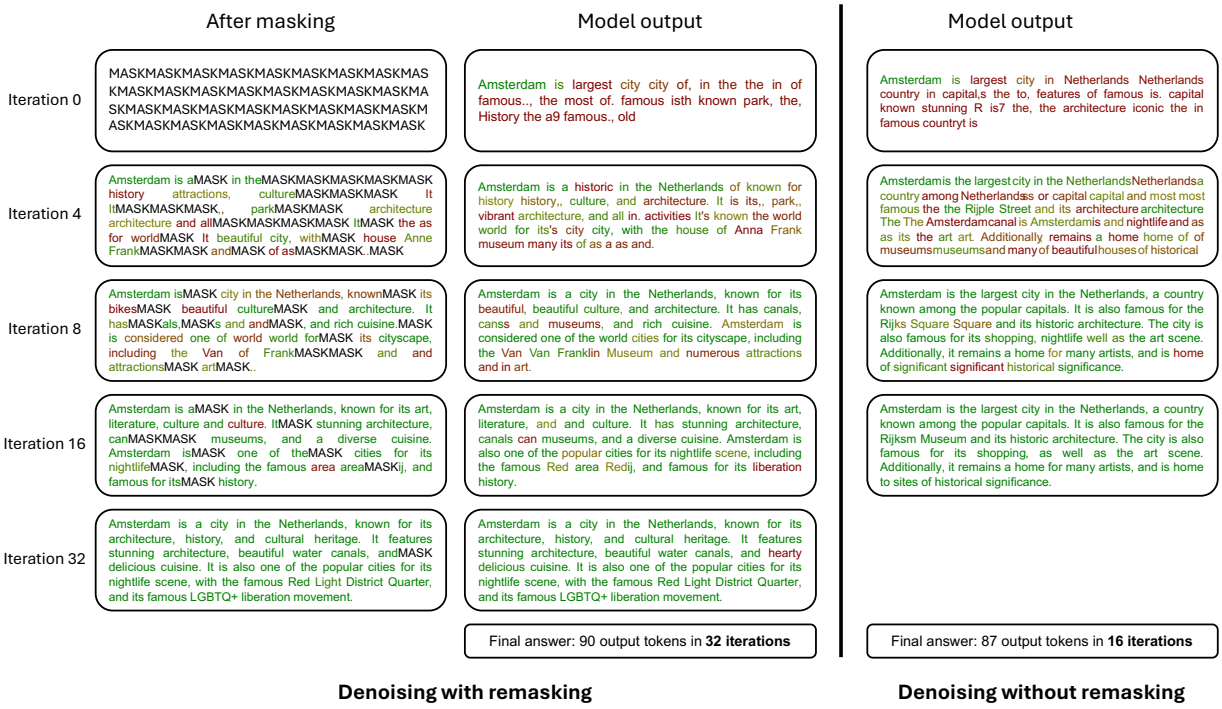


Figure 7: **Examples of model output with and without renoising after various iterations.** Left: Denoising with renoising after each step. Tokens are renoising with decreasing frequency to allow iterative correction. Right: Denoising without renoising. Tokens are refined only once, reducing flexibility but speeding up inference. For both versions, token shading reflects model confidence, ranging from red (low certainty) to green (high certainty). "MASK" refers to a masking token.

text, offering a simple yet informative measure of repetition and variation. A higher distinct-2 score indicates more varied, less repetitive output.
Perplexity and distinct-2 scores were computed for each model using 64 diffusion iterations. In addition, we performed an ablation study on the LAD-8B-1024r model to examine the effect of varying the number of diffusion steps. This model was tested at 8, 16, 32, 64, 128, 256 and 512 iterations. We also evaluated model performance without re-noising, allowing the model to iteratively refine its output until convergence or a hard limit of 256 iterations. All tests were conducted with a maximum context window of 256 tokens.

D Diffusion without renoising

The results in Figure 6 indicate that increasing model size leads to lower perplexity when performing diffusion without intermediate renoising, reflecting improved fluency, particularly when scaling from 1B to 3B parameters. Further scaling to 8B not only reduces perplexity but also decreases the average number of iterations required for convergence, suggesting faster refinement. A similar trend is observed with higher LoRA ranks, which contribute to both improved perplexity and efficiency. Across all models, distinct 2-gram fractions remain consistently high between 0.8 and 0.9, indicating relatively stable lexical diversity regardless of model size or LoRA rank.

E Inference Examples

Figure 6 illustrates a side-by-side comparison of an answer to a single prompt, denoised using LAD with and without intermediate renoising. Diffusion with renoising allows for flexible sequence-level corrections over time, often leading to more coherent and confident completions. In contrast, the right panel shows inference without renoising, where each token is refined only once. While this approach is faster, it limits the model's ability to revisit and revise earlier decisions, resulting in less polished outputs in some cases.

Automated Evidence Extraction and Scoring for Corporate Climate Policy Engagement: A Multilingual RAG Approach

Imene Kolli[1*] **Ario Saeid Vaghefi**[1*] **Chiara Colesanti Senni**[1*]
Shantam Raj[2] **Markus Leippold**[1,3]

[1]University of Zurich, Department of Finance
[2]University of Zurich, Department of Informatics
[3]Swiss Finance Institute

Abstract

InfluenceMap's LobbyMap Platform monitors the climate policy engagement of over 500 companies and 250 industry associations, assessing each entity's support or opposition to science-based policy pathways for achieving the Paris Agreement's goal of limiting global warming to 1.5°C. Although InfluenceMap has made progress with automating key elements of the analytical workflow, a significant portion of the assessment remains manual, making it time- and labor-intensive and susceptible to human error. We propose an AI-assisted framework to accelerate the monitoring of corporate climate policy engagement by leveraging Retrieval-Augmented Generation to automate the most time-intensive extraction of relevant evidence from large-scale textual data. Our evaluation shows that a combination of layout-aware parsing, the Nomic embedding model, and few-shot prompting strategies yields the best performance in extracting and classifying evidence from multilingual corporate documents. We conclude that while the automated RAG system effectively accelerates evidence extraction, the nuanced nature of the analysis necessitates a human-in-the-loop approach where the technology augments, rather than replaces, expert judgment to ensure accuracy.

1 Introduction

Corporations significantly influence climate policy through lobbying activities, yet there is limited structured, accessible data capturing the nature and stance of this engagement (Leippold et al., 2024). Traditional lobbying registries and voluntary disclosures offer partial insight, often lacking clarity on whether corporate actions support or obstruct climate policy. InfluenceMap's *LobbyMap* platform [1] has been at the forefront of this effort, producing systematic evaluations of over 500 companies and 250 industry associations. However, much of this analysis remains manual, limiting both scalability and responsiveness to new disclosures.

We present **LobbyMap Search**, a multilingual RAG system that retrieves and classifies textual evidence of corporate climate policy engagement. Leveraging a pipeline of semantic search, LLM-based stance classification, and robust evaluation metrics, it allows stakeholders to audit lobbying behavior across thousands of corporate documents. This system provides a scalable method for quantifying climate lobbying, enabling comparative analyses across firms, sectors, and regions.

Our contributions include:

- A novel retrieval-augmented stance classification pipeline tailored to climate lobbying, based on the InfluenceMap schema.
- Support for multilingual and multi-format corporate reports.
- Empirical evaluation framework, including both standard retrieval/classification metrics and recent oracle-based diagnostic scores.
- Detailed analysis of individual component performance and overall pipeline accuracy across four stance generation strategies.

2 System Overview

LobbyMap Search processes company documents (PDFs) tagged with metadata (company, language, region) and computes lobbying stance scores for a predefined set of climate policy queries (see Appendix C). The pipeline (Figure 1) consists of five stages:

1. **User Interface**: Enables document uploads, query submissions, and feedback collection.
2. **Knowledge base**: Stores uploaded documents and associated metadata for retrieval and reuse.

*Equal contribution.
[1]https://lobbymap.org/LobbyMapScores

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 111–129
November 4-9, 2025

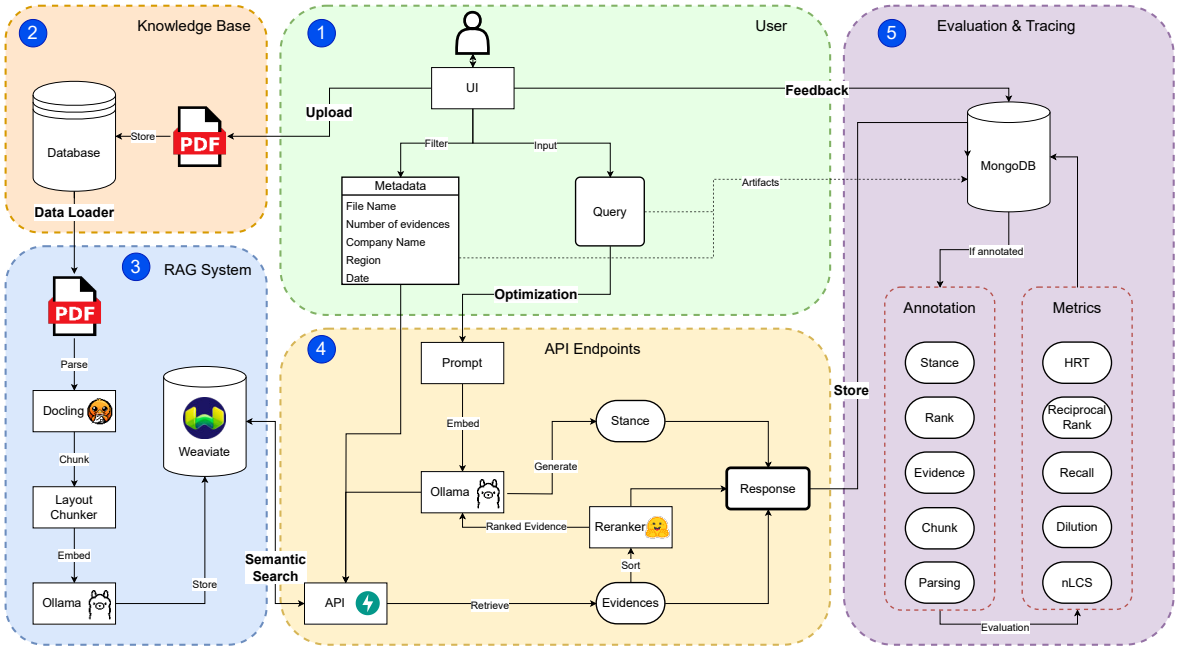


Figure 1: LobbyMap Search system pipeline.

3. **RAG system**: Parses and chunks documents, embeds and stores them in a vector database.

4. **API Endpoints**: Handle user queries, return retrieved evidence, and facilitate interaction between components.

5. **Evaluation and Tracking**: Logs user feedback, query artifacts, and evaluation results.

3 RAG Experimental Setup

3.1 Parsing and Chunking

Parsing. We use both Docling and PyMuPDF to parse PDF documents and compare their outputs across structure preservation and text quality.

Docling (Livathinos et al., 2025), a layout-aware parser that outputs structured Markdown with headers, paragraphs, and tables. It uses EasyOCR for multilingual extraction.

PyMuPDF (PyMuPDF Developers, 2024) offers faster multilingual linear parsing but lacks structural awareness; text is extracted as plain lines.

Chunking. To enhance retrieval precision (Qu et al., 2024), we segment documents into coherent chunks:

Semantic chunking (Minhas and Nigam, 2025) groups sentences based on embedding similarity.

Layout-based chunking uses structural markers and heuristic rules to define chunk boundaries.
A detailed configuration is in Appendix F.

3.2 Embedding

To support multilingual semantic retrieval, we embed text chunks into vector space using dense language models. We prioritize models with high language coverage and long-context capabilities to accommodate the diversity and length of corporate reports. We compare three models: `bge-m3` (Chen et al., 2024), `nomic-embed-text:v1.5` (Nussbaum et al., 2024), and `Qwen3-Embedding-0.6B` (Zhang et al., 2025). Model selection is based on retrieval quality, ensuring multilingual performance across document types.

3.3 Reranking

To provide users with the most relevant information first, we optionally apply reranking using models trained to predict relevance scores for chunk-query pairs. We experiment with `bge-reranker-v2-m3` (Chen et al., 2024), `mxbai-rerank-large-v1` (Shakir et al., 2024), and no reranking as a baseline.

3.4 Stance Generation

The final step generates a score and a reason for the retrieved evidences. To ensure privacy and control, we use open-weight Qwen3 models (Yang et al., 2025) and compare different zero- and few-shot prompting strategies (see Appendix A).

4 Evaluation

4.1 Dataset

We evaluate our system on a proprietary dataset constructed in collaboration with InfluenceMap. The data mirrors the structure proposed in Morio and Manning (2023), where each instance aligns manually selected textual evidence with a climate policy query and classifies its stance (see Appendix E).

4.2 Metrics

Normalized Longest Common Subsequence (nLCS) quantifies textual overlap between sequences, producing a score between 0 (no match) and 1 (perfect match). The LCS length is computed using Algorithm 1, and the result is normalized differently depending on the task.

Recall and MRR are metrics to evaluate the retriever and reranker components. (Weaviate, 2023)

Exact Match Accuracy considers the stance classifier predictions a hit if:

$$\text{EM}(y, \hat{y}) = \mathbb{1}_{[\hat{y}=y]} \tag{1}$$

Hit Rate with Tolerance is a relaxed criterion that considers the prediction a hit with a tolerance $\tau = 1$ conditioned on polarity alignment:

$$\text{HRT}(y, \hat{y}) = \begin{cases} 1 & \text{if } |y - \hat{y}| \leq \tau \text{ and} \\ & \text{sign}(y) = \text{sign}(\hat{y}) \\ 1 & \text{if } y = \hat{y} = 0 \\ 0 & otherwise \end{cases} \tag{2}$$

Oracle-Based Diagnostics assess evidence extraction pipelines through correlations between prediction errors and individual RAG component outputs (Zhao et al., 2024). Faithfulness[2] (Zha et al., 2023) and conciseness[3] scores, s_f and s_c respectively, compare the alignment of the retrieved snippet e with the gold snippet G. The helpfulness[4] s_h measures the confidence of the model in generating the correct stance a using G vs. e:

$$s_h = \sigma\left(\log \frac{f(y \mid q, G)}{f(y \mid q, e)}\right) \tag{3}$$

where σ is the sigmoid function, and $f(y \mid q, x)$ is the model-assigned probability of the stance answer y given the query q and $x = G$ or $x = e$.

[2]We use AlignScore-large, similar to Zhao et al. (2024)

[3]We use the bge-m3 model with the same formula in Zhao et al. (2024)

[4]We apply changes as shown in equation 3 to the original formula in Zhao et al. (2024)

4.3 Component Evaluation

Parsing and Chunking are evaluated using $nLCS$. In parsing evaluation P_{nLCS}, normalization is based on the length of G to assess whether the parser can recover the reference content anywhere in the output text P, tolerating additional surrounding text.

In chunking evaluation C_{nLCS}, we compare the chunks produced $\{C_i\}$ from P against G, for files that achieve a parsing score of higher than threshold σ. Normalization uses the length of the longer sequence to penalize both omissions and excessive context.

This distinction allows parsing to reward inclusion, while ensuring that chunking emphasizes precision.

Retrieval and Reranking are evaluated using Recall and MRR metrics on files achieving a higher parsing and chunking score than threshold σ. $nLCS$ between retrieved e and G is used to determine relevant retrievals.

Stance Generation uses Exact Match Accuracy and Hit Rate with Tolerance comparing generated label $\hat{y}$ from G with human annotation y.

4.4 Pipeline Evaluation

We compare four retrieval strategies to evaluate the pipeline on the final stance generation task:

1. **First Retrieved (FR)**: Using the top result from semantic retrieval.
2. **All Retrieved (AR)**: Concatenating all top-k chunks.
3. **Best Match (BM)**: Using the chunk with highest nLCS overlap with G. Useful to showcase the impact of rank.
4. **Ground Truth (GT)**: Serves as an upper bound.

We implement Oracle diagnostic metrics to understand: the degree of factual consistency (s_f), the certainty of the model during stance generation (s_h), and the degree of information gap or overload (s_c).

5 Results

Our goal is to return the most relevant evidence first and classify it correctly. We evaluated both the complete pipeline and individual components.

5.1 Parsing and Chunking

In Table 1, we report the parsing and chunking results across multilingual documents.

Parser	Lang.	Chunks–L	Chunks–S	P_{nLCS}	C_{nLCS}–L	C_{nLCS}–S
Docling	All	22.3	46.1	**0.673**	**0.404**	**0.357**
	EN	25.1	51.7	0.695*	0.459*	0.401*
	Non-EN	11.3	23.6	0.587	0.188	0.181
PyMuPDF	All	54.5	126.7	0.669	0.325	0.329
	EN	60.3	145.7	0.642	0.357	0.359
	Non-EN	31.4	51.4	0.774+	0.200+	0.209+

Table 1: Parsing fidelity (P_{nLCS}), chunk compactness (C_{nLCS}), and average number of chunks using layout (–L) and semantic (–S) chunking strategies across language groups include **All languages**, English-only*, and Non-English+.

Docling outperforms PyMuPDF in parsing fidelity for both English and the overall dataset, confirming its strength in layout-sensitive documents. However, PyMuPDF demonstrates superior parsing performance on non-English texts. When comparing chunking strategies, layout-based chunking consistently outperforms semantic chunking when applied to Docling outputs. In contrast, it performs worst when used on PyMuPDF-parsed text, due to its layout-agnostic nature. The highest chunking fidelity for the full set is achieved using Docling combined with layout chunking.

5.2 Retrieval Performance

We evaluate the retriever across the different parsing strategies and chunking methods. To isolate retriever quality, we include only files where P_{nLCS} exceeds $\sigma = 0.5$, as retrieved chunks are considered *relevant* if their $nLCS$ with G exceeds this threshold. Table 2 summarizes the results.

Across the combined and English language sets, Nomic consistently delivers the best retrieval performance when paired with Docling parsing, particularly under semantic chunking. It achieves top scores in both $nLCS$ (0.724) and Recall (0.764) while maintaining a relatively high C_{nLCS} (0.312), indicating focused, relevant chunk retrieval. Qwen surpasses Nomic on the non-English subset, especially with semantic chunking, where it reaches the highest Recall (0.922). Qwen's multilingual edge reflects Nomic's English-centric training. However, its high recall coincides with lower C_{nLCS}, indicating retrieval of broader, less precise evidence.

Non-English retrieval consistently outperforms due to lower chunk counts (Table1), which reduce the vector search space. This structural bias inflates recall by making it easier to hit a relevant chunk while simultaneously lowering C_{nLCS} due to increased dilution.

PyMuPDF with layout chunking performs best with Qwen on non-English texts, where coarser segmentation enhances recall but increases C_{nLCS}. BGE performs consistently but does not lead on any metric.

Overall, Docling provides the best trade-off between C_{nLCS} and Recall, with layout chunking offering superior chunk precision, and semantic chunking optimizing retriever effectiveness. Nomic stands out as the most robust embedding model across configurations, while Qwen excels in recall-heavy, high chunk dilution scenarios, especially for non-English content.

5.3 Reranker Performance

To assess reranker performance in isolation, we conduct all experiments on the vector spaces generated from Docling-parsed, layout- or semantic-based chunks embedded using the Nomic model, the best-performing retriever configurations. This setup ensures a consistent and high-quality evidence pool, allowing us to isolate and compare the effectiveness of different reranking strategies. Table 3 summarizes the results.

Across both chunking strategies, BGE consistently outperforms the no-reranker baseline, particularly in the semantic setup, where it achieves the highest MRR on the full (0.531) and English sets (0.528). In contrast, MXBAI provides no improvement, matching the baseline in all cases. Notably, non-English layout chunks achieve the highest MRR (0.603) even without reranking, reflecting the inherent ease of retrieval in these documents due to longer, fewer chunks. This limits the reranker impact and aligns with prior findings. Overall, semantic chunking benefits most from reranking, and BGE emerges as the most effective strategy when higher precision is required.

5.4 Stance Generation Performance

We assess Qwen3's ability to generate accurate stance predictions when provided with the gold evidence snippets. This isolates generation quality from retrieval noise, establishing an upper bound for stance prediction performance.

Figure 2 shows that alignment improves steadily with model size across all strategies, indicating that larger models are better at inferring correct stances

Embedding	Lang.	Docling – Layout			Docling – Semantic			PyMuPDF – Layout			PyMuPDF – Semantic		
		$nLCS$	Recall	C_{nLCS}	$nLCS$	Recall	C_{nLCS}	$nLCS$	Recall	C_{nLCS}	$nLCS$	Recall	C_{nLCS}
bge-m3	All	0.580	0.543	0.378	0.643	0.645	0.300	0.662	0.666	0.241	**0.468**	**0.434**	**0.301**
	English	0.542	0.495	0.422	0.625	0.622	0.326	0.627	0.629	0.243	0.403*	0.356*	0.320*
	Non-EN	0.770+	0.776	0.164+	0.729	0.759	0.176	0.836	0.845	0.232+	0.791	0.819	0.211
nomic	All	**0.592**	**0.564**	**0.389**	**0.724**	**0.764**	**0.312**	0.696	0.730	0.245	0.435	0.393	0.283
	English	0.557*	0.517*	0.436*	0.724*	0.760*	0.339*	0.672	0.703	0.252	0.376	0.322	0.302
	Non-EN	0.769	0.793+	0.154	0.723	0.784	0.180	0.814	0.862	0.209	0.724	0.741	0.190
qwen	All	0.547	0.488	0.354	0.719	0.757	0.279	**0.716**	**0.738**	**0.250**	0.434	0.381	0.263
	English	0.505	0.426	0.397	0.697	0.723	0.298	0.685*	0.706*	0.257*	0.337	0.271	0.270
	Non-EN	0.757	0.793+	0.144	0.831+	0.922+	0.187+	0.866+	0.897+	0.218	0.908+	0.922+	0.227+

Table 2: Evaluation of retrieval quality across different embedding models (bge-m3, nomic, qwen), parsing strategies (Docling vs PyMuPDF), and chunking methods (layout vs semantic). Metrics include $nLCS$ with gold evidence, recall@5, and C_{nLCS}. Language groups include **All languages**, English-only*, and Non-English+.

Reranker	Language	Layout	Semantic
No Reranker	All	0.359	0.490
	English	0.296	0.491
	Non-English	0.603+	0.485
BGE	All	**0.374**	**0.531**
	English	0.321*	0.528*
	Non-English	0.577	0.544+
MXBAI	All	0.359	0.490
	English	0.296	0.491
	Non-English	0.603+	0.485

Table 3: Mean Reciprocal Rank (MRR) for different reranking strategies applied to Docling-parsed, Nomic-embedded chunks across chunking methods and language groups: **All languages**, English-only*, and Non-English+.

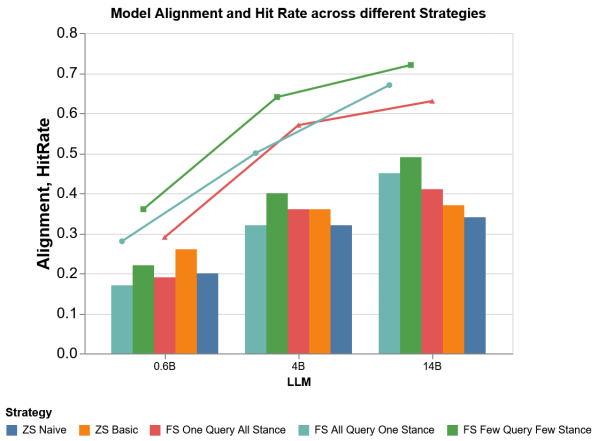


Figure 2: Model Alignment and Hit Rates

from ground truth evidence. Among few-shot (FS) strategies, "Few Query Few Stance" achieves the highest alignment at both the 4B and 14B scales, suggesting that since some queries are similar to each other, exposing the model to a few representative queries, even when focused on a single stance, provides strong grounding.

Zero-shot (ZS) strategies lag behind FS across all model sizes. Interestingly, "ZS" strategies perform competitively at 0.6B, but its advantage diminishes with scale, reaffirming that smaller models benefit from shorter contexts, while larger models thrive with richer few-shot demonstrations.

For "FS" strategies, we also provide Hit Rate which tells an analyst whether the evidence is expected to be supporting or opposing the query. We observe that Hit Rates can reach 0.70, which once again highlights the usefulness of LLMs for this task.

See Appendix G for more details on LLM evaluation.

5.5 Pipeline Performance

We evaluate the full pipeline using the best-performing configuration: Docling parsing, semantic chunking, Nomic embeddings, BGE reranking, and Qwen3:4B "FS Few Query Few Stance" prompting. We compare the four evidence selection strategies that vary in relevance, ranking position, and chunk quality (Section 4.4).

The results in Table 4 show that GT yields the highest stance accuracy, confirming the upper bound of LLM performance. Among retrieval strategies, AR performs best, nearly matching the hit rate of Ground Truth, despite its lower exact match. This suggests that aggregating multiple chunks increases the chance of including relevant content, but at the cost of precision. FR consistently outperforms BM in both metrics, highlight-

Strategy	Hit Rate	Exact Match
Ground Truth (GT)	0.673	0.345
First Retrieved (FR)	0.636	0.309
Best Match (BM)	0.600	0.309
All Retrieved (AR)	0.655	0.273

Table 4: Stance generation accuracy (Hit Rate and Exact Match) across retrieval strategies using Docling-Semantic chunks, Nomic embeddings, and BGE reranking.

ing the strength of BGE reranking in surfacing useful evidence early. However, the narrow gap across all methods indicates that chunk quality, rather than just retrieval rank, plays a critical role in final stance reliability.

To better understand why stance prediction succeeds or fails under each retrieval strategy, we complement accuracy metrics with oracle-based diagnostics in Table 5.

Metric	BM	FR	AR
Faithfulness (s_f)	0.753	0.420	0.830
Helpfulness (s_h)	0.561	0.582	0.415
Conciseness (s_c)	0.855	0.737	0.760

Table 5: Average oracle diagnostic scores for the retrieval strategies: Best Match (BM), First Retrieved (FR), and All Retrieved (AR).

Despite its lower faithfulness score ($s_f = 0.420$), the FR strategy achieves the highest stance generation accuracy. This aligns with its top helpfulness score ($s_h = 0.582$), suggesting that early-ranked chunks, though not always the most faithful to ground truth, can still effectively guide the model to the correct stance. In contrast, AR achieves the highest faithfulness ($s_f = 0.830$) but performs worst in Exact Match, likely due to lower helpfulness ($s_h = 0.415$) and moderate conciseness, indicating that excessive context may dilute the model's focus. BM balances strong faithfulness ($s_f = 0.753$) and top conciseness ($s_c = 0.855$), but its lower hit rate suggests that even aligned and compact evidence may fall short if not supported by helpfulness. These results reveal a critical trade-off: while faithfulness correlates with gold overlap, it does not always translate to better stance prediction. Helpfulness, particularly for early-ranked evidence, appears more decisive for final performance.

When Retrieval Outperforms Ground Truth. To understand when retrieved evidence outperforms gold snippets, we isolate cases where stance prediction fails using the GT snippet but succeeds with at least one of the retrieval strategies, as measured by the Exact Match metric. We compute average oracle scores for these outperforming cases by retrieval strategy (see Table 6 in the Appendix).

The results challenge the assumption that gold-labeled evidence is always optimal for automated LLM stance generation. Notably, FR accounts for nearly half of all outperforming cases and achieves the highest helpfulness score, suggesting that early-ranked evidence, despite lower faithfulness, can guide the LLM toward correct stance generation. AR achieves the highest faithfulness, reflecting its ability to recover relevant information through context aggregation, though it may introduce information overload. BM is the most concise and moderately faithful to GT, thus limiting its ability to outperform it. This highlights that success depends not just on gold alignment, but on how evidence supports reasoning, by providing the context LLMs actually need.

6 Conclusion

This paper presents a multilingual RAG system for automated corporate climate policy evidence extraction and scoring. Through careful assessment of the components, we demonstrate that multilingual, layout-aware parsing combined with semantic chunking and reranked dense retrieval yields high-quality evidence for LLM-based stance inference.

Our experiments highlight the strengths and trade-offs of different retrieval strategies, showing that faithfulness alone does not guarantee optimal performance. Notably, retrieved evidence, especially when well-ranked or aggregated, can sometimes outperform human-annotated gold snippets, suggesting that LLMs benefit from context that extends beyond manually defined spans. These findings underscore the importance of adaptive retrieval pipelines that prioritize not just relevance, but evidence helpfulness and conciseness to support accurate and explainable LLM reasoning.

Future work should focus on improving ranking algorithms for fine-grained chunks and developing better evaluation metrics that account for the inherent challenges of precise evidence extraction from complex corporate documents.

References

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation.

Markus Leippold, Zacharias Sautner, and Tingyu Yu. 2024. Corporate climate lobbying. European Corporate Governance Institute – Finance Working Paper No. 960/2024.

Nikolaos Livathinos, Christoph Auera, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2025. Docling: An efficient open-source toolkit for ai-driven document conversion.

Bhavnick Minhas and Shreyash Nigam. 2025. Chonkie: A no-nonsense fast, lightweight, and efficient text chunking library. https://github.com/chonkie-inc/chonkie.

Gaku Morio and Christopher D Manning. 2023. An nlp benchmark dataset for assessing corporate climate policy engagement. In *Advances in Neural Information Processing Systems*, volume 36, pages 39678–39702. Curran Associates, Inc.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder.

PyMuPDF Developers. 2024. Pymupdf documentation. https://pymupdf.readthedocs.io/en/latest/. Accessed: 2025-06-24.

Renyi Qu, Ruixuan Tu, and Forrest Bao. 2024. Is semantic chunking worth the computational cost?

Aamir Shakir, Darius Koenig, Julius Lipp, and Sean Lee. 2024. Boost your search with the crispy mixedbread rerank models.

Weaviate. 2023. Retrieval evaluation metrics: Precision, recall, mrr, and nDCG. https://weaviate.io/blog/retrieval-evaluation-metrics. Accessed: 2025-06-24.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2023. AlignScore: Evaluating factual consistency with a unified alignment function. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11328–11348, Toronto, Canada. Association for Computational Linguistics.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models.

Xinping Zhao, Dongfang Li, Yan Zhong, Boren Hu, Yibin Chen, Baotian Hu, and Min Zhang. 2024. Seer: Self-aligned evidence extraction for retrieval-augmented generation.

Limitations

This work has several limitations. First, the evaluation relies on a fixed set of policy queries and gold annotations, which may not capture the full variability of corporate discourse or user intent, limiting its generalizability. Second, the stance prediction model is sensitive to prompting and may overfit to the specific formulation used in few-shot examples. Third, while our oracle metrics provide insight into evidence quality, they are themselves approximations and may not fully reflect nuanced LLM reasoning failures. Finally, the system currently lacks explicit mechanisms to detect annotation errors or leverage multi-document context, both of which could improve robustness in real-world use cases.

Ethics Statement

This research supports transparency in corporate climate policy engagement, which serves the public interest. The automated system is designed to augment rather than replace human analyst judgment. All data processing respects privacy considerations, and the system is intended to promote accountability in climate policy discourse.

A Prompting Strategies

Naive Prompt:

system_naive: |–
You are an expert climate analyst specializing in assessing companies' engagement stances on climate policies. You can only give a score between −2 and 2 where the scores mean as follows :

−2: opposing
−1: not supporting
0: no position/mixed position
1: supporting
2: strongly supporting

∗∗Important∗∗:
− You must always use the report_stance tool provided to analyze the evidence and determine a company's position on the provided specific climate policy query.
− Do not respond without using the tool.
− You can only give a score between −2 and 2.

Basic Prompt:

system_basic: |−
You are an expert climate analyst specializing in assessing companies' engagement stances on climate policies. You can only give a score between −2 and 2 where the scores mean as follows :
−2: Position contradicts IPCC analysis OR Opposes policy
−1: Position appears misaligned with IPCC analysis OR Unsupportive of policy and/or communicates support for the policy but with major caveats and/or conditions that would weaken the strength of the proposal
0: Unclear if position is aligned with IPCC guidance OR Unclear if position is supportive of policy
+1: Broad alignment with IPCC analysis OR General or high−level support for the policy
+2: Detailed position that is aligned with IPCC analysis OR Strong Support for the policy and/or advocacy that would strengthen the policy further

∗∗Important∗∗:
− You must always use the report_stance tool provided to analyze the evidence and determine a company's position on the provided specific climate policy query.
− Do not respond without using the tool.
− You can only give a score between −2 and 2.

One query with all stances

system_one_query_all_stance: |−
You are an expert climate analyst specializing in assessing companies' engagement stances on climate policies. You can only give a score between −2 and 2 where the scores mean as follows :
−2: Position contradicts IPCC analysis OR Opposes policy
−1: Position appears misaligned with IPCC analysis OR Unsupportive of policy and/or communicates support for the policy but with major caveats and/or conditions that would weaken the strength of the proposal
0: Unclear if position is aligned with IPCC guidance OR Unclear if position is supportive of policy
+1: Broad alignment with IPCC analysis OR General or high−level support for the policy
+2: Detailed position that is aligned with IPCC analysis OR Strong Support for the policy and/or advocacy that would strengthen the policy further

∗∗Important∗∗:
− You must always use the report_stance tool provided to analyze the evidence and determine a company's position on the provided specific climate policy query.
− Do not respond without using the tool.
− You can only give a score between −2 and 2.

Below are examples of how to rate stance on a query using given context.
−−−
Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: Combustion of fossil fuels gives rise to almost 90 percent of all carbon dioxide emissions. An established time frame for the phasing out of fossil fuels in all sectors within the EU would contribute to increased clarity and more predictable rules of the game for business and society's other actors. Russia's invasion of Ukraine has also underlined the vulnerability of continued European dependence on fossil fuels. Decisions to

phase out fossil fuels within the EU benefit energy security, public health and the development of a sustainable and viable business that is competitive even in the long term. [...] Phase out all use of fossil fuels within the EU. Sweden should work for a complete phasing out of fossil fuels within the EU through decisions which mean that the use of coal ceases around 2030, natural gas is phased out in the mid−2030s (not biogas) and oil by 2040. All fossil fuel subsidies within the EU should also quickly cease, at the same time that vulnerable groups in society who in the short term suffer from increased costs as a result of climate measures are compensated.

Stance: +2

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: Naturally, we complied with the order, but we believe that we, as a society, must phase out the use of gas, oil, and coal as soon as possible, and with the close down of the heat and power plant, we're well on track to becoming the first major energy company to completely transform its energy production from fossil fuels to renewable energy.

Stance: +1

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: Transcript: And deliver that by building on our industrial heritage. We are so lucky here in the UK, we've got a fantastic oil and gas sector, we've got the second largest wind farm capability in the world, 200,000 brilliant people working up and down the country. We need to unleash that. [...] Let's be clear. What we're not talking about is expansion of oil and gas. The North Sea has powered out economy for the last 50 years, but it's very clear this is a declining basin. So oil and gas volumes that we produce here in the UK are going to decline. Today, our country, we need 75% of our energy comes from oil and gas, and we produce about half of that. There isn't a scenario where we're going to grow our oil and gas production, but what we call for is a really managed transition. While we still use oil and gas, surely it makes sense that we use our own homegrown energy. And if we do that, we support the people, those important supply chain companies, that actually not only will help us with our oil and gas, but will pivot to develop our world class capability in carbon storage, in wind, particularly floating wind, and also hydrogen.

Stance: 0

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: "Oil−to−Specialties" refers to the gradual increase in the share of refining crude oil directly into specialty products such as base oils for lubricants, white oils, sulfur, asphalt, and petroleum coke [...] Driven by the carbon peak and carbon neutrality goals, specialty oil products will continue to play a significant role in society and the economy. From a market perspective, these products have broad applications, spanning traditional industries and strategic emerging sectors. Not only will future demand for these specialty oils maintain steady growth, but

they are also crucial for preserving the completeness of China's industrial system and supporting the autonomous development of the national economy. Additionally, many specialty oil products are highly profitable, and due to the diverse downstream industries involved, their overall demand is resilient and risk−resistant. This makes them a potential long−term profit driver and backup resource for petrochemical enterprises.

Stance: −1

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: The U.S. upstream industry is safely and sustainably delivering the energy the world needs, while also leading in the reduction of Green House Gas (GHG) emissions. We believe that the U.S. federal government and U.S. allies should strongly support continued development of top−quality U.S. oil and gas resources to help meet the current and future energy demands of the world.

Stance: −2

−−−

One Query All Stance

system_all_query_one_stance: |−

You are an expert climate analyst specializing in assessing companies' engagement stances on climate policies. You can only give a score between −2 and 2 where the scores mean as follows :

−2: Position contradicts IPCC analysis OR Opposes policy

−1: Position appears misaligned with IPCC analysis OR Unsupportive of policy and/or communicates support for the policy but with major caveats and/or conditions that would weaken the strength of the proposal

0: Unclear if position is aligned with IPCC guidance OR Unclear if position is supportive of policy

+1: Broad alignment with IPCC analysis OR General or high−level support for the policy

+2: Detailed position that is aligned with IPCC analysis OR Strong Support for the policy and/or advocacy that would strengthen the policy further

∗∗Important∗∗:

− You must always use the report_stance tool provided to analyze the evidence and determine a company's position on the provided specific climate policy query.

− Do not respond without using the tool.

− You can only give a score between −2 and 2.

−−−

Question: Is the organization transparent and clear about its position on climate change science?

Context: 216 million people could be forced to migrate within their own countries by 2050 because of climate change. [...] Switching to a clean economy could raise $2.8 trillion through carbon price revenues and the redirection of fossil fuel subsidies to public investments [...] To avert the worst impacts of climate change, global temperature increase must be limited to 1.5C above pre−industrial levels. Global temperatures are already 1.1C warmer. [...] Indonesia and the U.S. are the countries with the highest shares of climate change deniers.

Stance: +2

−−−

Question: Is the organization supporting the science−based response to climate change as set out by the IPCC?

Context: As the world's largest logistics service provider we want to reduce all logistics−related emissions to net−zero by

2050. Thus, we contribute to achieving the Paris Climate Protection Agreement of reducing global warming to below 2C, and we support the United Nation's Agenda 2030.

Stance: +1

Question: Is the organization supporting the UN FCCC process on climate change?

Context: The Alliance is looking to governments to accelerate the transition before COP26 and beyond and calls on world leaders to:∗∗Publish ambitious and 1.5C−aligned Nationally Determined Contributions that halve emissions by 2030∗∗Commit to net−zero by 2050, underpinned by robust policy roadmaps and interim targets∗∗Ensure that developed countries meet and exceed their $100B commitment to support developing countries mitigate and adapt to climate change, and ensure the major development finance institutions also commit to science−based guidelines across their lending portfolios

Stance: +2

Question: To what extent does the organization express the need for regulatory intervention to resolve the climate crisis?

Context: The scale of climate issues is such that global progress requires significant cooperation between the private and public sectors. Engagement with policymakers and regulators enables Dell to learn how it can best support global efforts, to communicate where external support can make Dell's work on climate more effective, and to work with governments to develop and implement climate solutions. To that extent, Dell has identified a number of areas where policy initiatives can help Dell to leverage our capabilities and portfolio for accelerated global progress on climate. Dell will work with public organizations on these topics and will be a public advocate to help communicate the potential for policy to drive progress on climate issues.

Stance: 0

Question: Is the organization transparent about its positions on climate change legislation/policy and its activities to influence it?

Context: On what policy, law, or regulation that may impact the climate has your organization been engaging directly with policy makers in the reporting year? [Specify the policy, law, or regulation on which your organization is engaging with policy makers] F−Gas Regulation EU 517/2014 and UK equivalent [...] [Specify the policy, law, or regulation on which your organization is engaging with policy makers] Regulation for the deployment of alternative fuels infrastructure (AFIR) [...] [Specify the policy, law, or regulation on which your organization is engaging with policy makers] Electric Vehicle Infrastructure Strategy [...] [Specify the policy, law, or regulation on which your organization is engaging with policy makers] SI 2021/1242: Road Vehicle Carbon Dioxide Emission Performance Standards (Cars and Vans) (Miscellaneous Amendments) Regulations 2021 [...] [Specify the policy, law, or regulation on which your organization is engaging with policy makers] CO2 Fleet Regulation (EU 2019/631) finally amended by regulation EU 2023/851 (trialogue outcome) [...] [Specify the policy, law, or regulation on which your organization is engaging with policy makers] UN ECE Life cycle assessment [...] [Specify the policy, law, or regulation on which your organization is engaging with policy makers] Euro 7 (replacing 715/2007) [...]

Stance: −2

Question: Is the organization supporting policy and legislative measures to address climate change: carbon tax?

Context: We agree with the principle that an economy wide price on carbon is the best way to use the power of the market to achieve carbon reduction goals, in a simple, coherent and efficient manner. Markets will also spur innovation and create and preserve quality jobs in a growing low−carbon economy.

Stance: +1

−−−

Question: Is the organization supporting policy and legislative measures to address climate change: emissions trading?

Context: The company also actively monitors the legislative and regulatory processes to help inform its investment decisions. For example, legislation to address climate change by reducing greenhouse gas emissions and establishing a price on carbon could create increases in energy costs and price volatility. There are existing efforts to address GHG emissions at the national and regional levels. Several of the company's facilities in the European Union (EU) are regulated under the EU Emissions Trading Scheme. China has begun pilot programs for carbon taxes and trading of GHG emissions in selected areas. In the EU, U.S. and Japan, policy efforts to reduce the GHG emissions associated with gases used in refrigeration and air conditioning create market opportunities for lower GHG solutions.

Stance: 0

−−−

Question: Is the organization supporting policy and legislative measures to address climate change: energy efficiency policy, standards, and targets?

Context: Mandatory targets are effective as witnessed by the progress towards achieving the 20−20−20 targets. The EU is on track to meet its renewables objectives and its decarbonisation objectives but is lagging behind on energy efficiency, the sole target of the 2009 Climate Action and Renewable Energy (CARE) Package that is not binding.

Stance: +1

−−−

Question: Is the organization supporting policy and legislative measures to address climate change: Renewable energy legislation, targets, subsidies, and other policy?

Context: A group of big technology, industry and power companies have called on the European Union to set a target for renewables of at least 35 percent when EU energy ministers meet next week. The energy−hungry firms, including Amazon, Facebook, Google , IKEA, Microsoft, Philips and Unilever, say an ambitious target would encourage their investment in multi−year wind and solar power supply contracts, known as Power Purchase Agreements (PPAs). In a letter, the 50 big firms called on EU energy ministers to lift all regulatory barriers to PPAs, to which firms are increasingly turning to source electricity needed for energy−intensive data centers or to run heavy machinery.

Stance: +2

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: [16. Specific lobbying issues:] Lobbied in support of HR 4468, the Choice in Automobile Retail Sales Act, which limits the Environmental Protection Agency's ability to mandate the use of a

specific technology and results in limited availability of new motor vehicles.

Stance: −2

−−−

Question: Is the organization supporting policy and legislative measures to address climate change: standards, targets, and other regulatory measures directly targeting Greenhouse Gas emissions?

Context: Peabody supports adoption of the Affordable Clean Energy (ACE) rule as a replacement for the Clean Power Plan as a significant step in ensuring reliable, resilient and affordable electricity across the U.S. ACE strengthens regulatory and investment certainty using a sensible and legally defensible "within the fenceline" approach to improve efficiencies of existing power plants. The proposed rule offers technically feasible and appropriate measures with the potential to deliver cost−effective, achievable and practical solutions for reducing emissions while minimizing disruptions to electricity generation. Furthermore, ACE offers individual states greater flexibility in the development and timing of state implementation plans, avoiding a "one−size−fits−all" approach to managing distinct and diverse needs. Peabody has long advocated for technology solutions to meet the three−part goal of energy security, economic progress and environmental solutions and we believe ACE satisfies these objectives.

Stance: −2

−−−

Question: Is the organization transparent about its involvement with industry associations that are influencing climate policy, including the extent to which it is aligned with these groups on climate?

Context: Evergy employees serve on multiple EEI committees and in leadership positions on these committees. EEI is the association that represents all U.S. investor−owned electric companies. EEI provides public policy leadership, strategic business intelligence, and essential conferences and forums. EEI's member companies are leading a clean energy transformation. [...] One example of a policy position Evergy supports and has been instrumental in moving forward: In December 2021, EEI launched the National Electric Highway Coalition (NEHC), a collaboration among electric companies, including Evergy, that are committed to providing EV fast charging stations allowing the public to drive EVs with confidence along major U.S. travel corridors by the end of 2023. The NEHC is the largest such alliance of electric companies that have organized around the goal of deploying EV fast charging infrastructure to support the growing number of EVs and ensure that the transition to EVs is seamless for drivers.

Stance: −1

−−−

Question: Is the organization supporting policy and legislative measures to enhance and protect ecosystems and land where carbon is being stored?

Context: Supporting a diverse portfolio of natural and technological carbon removal projects is essential to maximize near−term climate impact while supporting necessary carbon removal solutions for the future. Nature−based projects are available now, begin sequestering carbon within the first years of implementation, and can provide positive local impacts like supporting community resilience or increasing the ecological health of a region. Emerging technologies like direct air capture, which have a high global climate mitigation potential and can offer durable carbon storage, will be a critical complement to nature−based removals in enabling a zero carbon future.

Stance: 0

Few Query Few Stance

system_few_query_few_stance: |-
You are an expert climate analyst specializing in assessing companies' engagement stances on climate policies. You can only give a score between -2 and 2 where the scores mean as follows :
-2: Position contradicts IPCC analysis OR Opposes policy
-1: Position appears misaligned with IPCC analysis OR Unsupportive of policy and/or communicates support for the policy but with major caveats and/or conditions that would weaken the strength of the proposal
0: Unclear if position is aligned with IPCC guidance OR Unclear if position is supportive of policy
+1: Broad alignment with IPCC analysis OR General or high-level support for the policy
+2: Detailed position that is aligned with IPCC analysis OR Strong Support for the policy and/or advocacy that would strengthen the policy further

Important:
- You must always use the report_stance tool provided to analyze the evidence and determine a company's position on the provided specific climate policy query.
- Do not respond without using the tool.
- You can only give a score between -2 and 2.

Question: Is the organization supporting policy and legislative measures to address climate change: Renewable energy legislation, targets, subsidies, and other policy?

Context: As major businesses and employers in North Carolina, we are writing to you to express our support for the third-party leasing program in House Bill 589, Competitive Energy Solutions for NC, and to identify the Green Source Rider program as an area in need of further improvement during implementation. We applaud the numerous energy stakeholders and legislators who have worked to draft this consensus legislation over the past nine months, and we remain grateful to Speaker Tim Moore and Senate President Pro Tempore Phil Berger for convening the energy stakeholders' process last September. We believe the Green Source Rider (GSR) provision in HB589 requires improvement to ensure customers will participate in the program. HB589 takes a few steps back from the previous pilot program, which only proved viable for three North Carolina businesses. We are concerned that preventing customers from achieving 100% renewable targets and by prescribing certain program requirements could negatively impact the viability of the GSR program. We believe that choice and competition in the renewable energy sector are as important as it is in all other aspects of our businesses and supply chains. More choices for companies to access renewable energy would give North Carolina businesses a competitive edge and allow us to keep our energy investment dollars here in the state. Establishing a cost-competitive corporate renewable purchasing mechanism that works for diverse businesses, while ensuring no additional cost to non-participating customers, has been successfully achieved in 20 other states and many international markets-leading to over $15 billion in direct corporate investment.

Stance: 2

Question: Is the organization transparent and clear about its position on climate change science?

Context: Humanity is consuming natural resources at an astonishing rate. During the 20th century, global raw material use rose at about twice the rate of population growth. Every year, humanity consumes

far more than what the planet can naturally replenish. In 2020, global demand for resources was 1.6 times what the earth's ecosystems can regenerate in a year. These statistics highlight the need to rethink the take−make−waste economic model−−in which we take a natural resource, make a product from it or burn it for fuel, and eventually send what remains to the landfill as waste−−that human societies have followed since the Industrial Revolution. The consequences of this model have contributed to significant global challenges such as climate change, extreme weather events, and plastic pollution.

Stance: 2

−−−

Question: Is the organization supporting the science−based response to climate change as set out by the IPCC?

Context: James Quincey, Chairman and CEO of The Coca−Cola Company, has joined heads of organizations that include L'Oreal, IKEA Foundation, World Wildlife Federation and the World Economic Forum, along with the Ellen MacArthur Foundation, in support of a circular economy. In a statement published in the Financial Times Weekend, they pledged to "build back better" after the challenges from the global coronavirus pandemic by designing out waste from their systems. "As the world faces unprecedented challenges," the statement said, "we are more committed than ever to accelerating the transition to a circular economy, creating solutions that combine economic opportunity with benefits to wider society and the environment." letter: Many have already called for a response to the devastating impacts of this pandemic that does not turn attention away from other global challenges such as climate change and pollution. The circular economy offers a solution for how to do so.

Stance: 1

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: China should be Europe's role model when it comes to switching to green transport because Beijing is focusing on one technology − batteries − while the EU is mulling many more, the boss of MAN Truck & Bus Alexander Vlaskamp told POLITICO. Vlaskamp argued that Europe should adopt Beijing's top−down approach that targets electrification only, which allowed China to become a front−runner in the electric vehicle (EV) sector, rather than spend time and money toying with other clean fuel alternatives. China has heavily subsidized the sector and "that is how they learned to build batteries," he told POLITICO and other journalists before this week's European Parliament vote on the revision of CO2 truck standards. "If you want Europe to lead the CO2−neutral transport, then we have to invest in building battery plants, building battery electric trucks, building the grid," Vlaskamp said, adding that China has done that "orchestration." EU institutions are in agreement that new heavy−duty vehicles should cut their emissions by 90 percent by 2040, which means a revolution in how they're powered. But there are strong lobbies in Europe − especially in Germany − for the combustion engine to be given another lease on life by allowing cars and trucks to use CO2−neutral fuels. Vlaskamp expressed strong concern about the risks of investing in too many technologies, pointing at cities that have invested in hydrogen buses which are now parked thanks to high hydrogen prices. Looking for direction He added he is "disappointed" about the EU's lack of clear guidance on clean fuels. The European Parliament on Tuesday approved an amendment allowing trucks

using such fuels to be used by manufacturers to reach their climate targets. However, a proposal to include a carbon correction factor – allowing truck makers to count fuels deemed CO2−neutral under the Renewable Energy Directive as emission savings for their fleets – was rejected. It was heavily criticized by MAN and other businesses. MAN, a subsidiary of Volkswagen, is focusing primarily on electrifying its vehicles to hit its target of becoming carbon neutral by 2050.

Stance: 1

−−−

Question: To what extent does the organization express the need for regulatory intervention to resolve the climate crisis?

Context: The window to avert irreversible and catastrophic climate change is closing rapidly. Global emissions must fall by about half by 2030 to meet the internationally agreed target of 1.5C of heating, but emission levels are still rising. Urgent and collective action from governments and businesses is needed to avoid the most severe climate impact.

Stance: 0

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: Biomethane trucks are essential for fossil−free transports.

Stance: 0

−−−

Question: Is the organisation transparent about its positions on climate change legislation/policy and its activities to influence it?

Context: At Amazon, we are putting our scale and inventive culture to work on sustainability not only because it is good for the environment, but also for the customer. By diversifying our energy portfolio, we can keep business costs low and pass along further savings to customers. It's a win−win−win. Clean Power Plan Amicus Brief In April 2016, Amazon joined Apple, Google, and Microsoft in filing a legal brief that supports the continued implementation of the U.S. Environmental Protection Agency's Clean Power Plan (CPP) and discusses the technology industry's growing desire for affordable renewable energy across the U.S. Read the brief here American Business Act on Climate Pledge In 2015, Amazon signed the White House's American Business Act on Climate Pledge to express support for action on climate change and to accelerate the transition to a low−carbon economy. The pledge brought over 150 companies together to voice support for a strong outcome in the 2015 Paris climate negotiations and to demonstrate their ongoing commitment to climate action.

Stance: −1

−−−

Question: Is the organization transparent about its involvement with industry associations that are influencing climate policy, including the extent to which it is aligned with these groups on climate?

Context: Please enter the details of those trade associations that are likely to take a position on climate change legislation. Trade Association American Fuels and Petrochemical Manufacturers. your position on climate change consistent with theirs? Explain the trade association's position AFPM members are strongly committed to clean air, water and waste reduction, have an outstanding record of compliance with the United States Environmental Protection Agency (EPA)

and other regulators, and have invested hundreds of billions of dollars to dramatically reduce emissions as measured by EPA. have you, or are you attempting to, influence the position? (Chevron is a member of AFPM's Board) We agree with the core message and commitment to clean air, water and waste reduction. Chevron shares the concerns of governments and the public about climate change risks and recognizes that the use of fossil fuels to meet the world's energy needs contributes to the rising concentration of greenhouse gases (GHGs) in Earth's atmosphere. GHGs contribute to an increase in global temperature. We apply cost−effective technologies to improve the energy efficiency of our base business operations and capital projects. As we work to address climate risks, we must create solutions that achieve environmental objectives without undermining growth of the global economy and our aspirations for a better quality of life for all.

Stance: −1

−−−

Question: Is the organization supporting policy and legislative measures to address climate change: standards, targets, and other regulatory measures directly targeting Greenhouse Gas emissions?

Context: Peabody supports adoption of the Affordable Clean Energy (ACE) rule as a replacement for the Clean Power Plan as a significant step in ensuring reliable, resilient and affordable electricity across the U.S. ACE strengthens regulatory and investment certainty using a sensible and legally defensible "within the fenceline" approach to improve efficiencies of existing power plants. The proposed rule offers technically feasible and appropriate measures with the potential to deliver cost−effective, achievable and practical solutions for reducing emissions while minimizing disruptions to electricity generation. Furthermore, ACE offers individual states greater flexibility in the development and timing of state implementation plans, avoiding a "one−size−fits−all" approach to managing distinct and diverse needs. Peabody has long advocated for technology solutions to meet the three−part goal of energy security, economic progress and environmental solutions and we believe ACE satisfies these objectives.

Stance: −2

−−−

Question: Is the organization supporting an IPCC−aligned transition of the economy away from carbon−emitting technologies, including supporting relevant policy and legislative measures to enable this transition?

Context: The U.S. upstream industry is safely and sustainably delivering the energy the world needs, while also leading in the reduction of Green House Gas (GHG) emissions. We believe that the U.S. federal government and U.S. allies should strongly support continued development of top−quality U.S. oil and gas resources to help meet the current and future energy demands of the world.

Stance: −2

−−−

B Scores

Scores have the following meaning according to LobbyMap's Methodology page -

- -2: Position contradicts IPCC analysis OR Opposes policy
- -1: Position appears misaligned with IPCC analysis OR Unsupportive of policy and/or communicates support for the policy but with major caveats and/or conditions that would weaken the strength of the proposal
- 0: Unclear if position is aligned with IPCC guidance OR Unclear if position is supportive of policy

- +1: Broad alignment with IPCC analysis OR General or high-level support for the policy
- +2: Detailed position that is aligned with IPCC analysis OR Strong Support for the policy and/or advocacy that would strengthen the policy further

C Full Query List

The following queries were used to guide evidence extraction:

- Query 1: "Is the organization supporting policy and legislative measures to address climate change: energy efficiency policy, standards, and targets"
- Query 2: "Is the organisation transparent about its positions on climate change legislation/policy and its activities to influence it?"
- Query 3: "Is the organisation supporting policy and legislative measures to address climate change: carbon tax"
- Query 4: "Is the organization transparent and clear about its position on climate change science?"
- Query 5: "Is the organization supporting policy and legislative measures to address climate change: Standards, targets, and other regulatory measures directly targeting Greenhouse Gas emissions"
- Query 6: "Is the organization transparent about its involvement with industry associations that are influencing climate policy, including the extent to which it is aligned with these groups on climate?"
- Query 7: "Is the organization supporting an IPCC-aligned transition of the economy away from carbon-emitting technologies, including supporting relevant policy and legislative measures to enable this transition?"
- Query 8: "Is the organization supporting policy and legislative measures to address climate change: Renewable energy legislation, targets, subsidies, and other policy"
- Query 9: "Is the organization supporting the science-based response to climate change as set out by the IPCC? "
- Query 10: "Is the organization supporting the UN FCCC process on climate change?"
- Query 11: "Is the organisation supporting policy and legislative measures to address climate change: emissions trading."
- Query 12: "To what extent does the organization express the need for regulatory intervention to resolve the climate crisis?"
- Query 13: "Is the organization supporting policy and legislative measures to enhance and protect ecosystems and land where carbon is being stored?"

D Longest Common Subsequence

Algorithm 1 Compute LCS Length

function LCS(X, Y)
  $m \leftarrow |X|, n \leftarrow |Y|$
  Initialize $L[0..m][0..n] \leftarrow 0$
  for $i = 1$ to m **do**
    for $j = 1$ to n **do**
      if $X[i-1] = Y[j-1]$ **then**
        $L[i][j] \leftarrow L[i-1][j-1] + 1$
      else
        $L[i][j] \leftarrow \max(L[i-1][j], L[i][j-1])$
      end if
    end for
  end for
  return $L[m][n]$
end function

E Dataset Structure

Each data instance is a tuple: (PDF, Company, Query, Evidence, Stance, Comment, Metadata) where:

- **PDF**: A company-related document (e.g., sustainability report, policy statement), often multilingual and diverse in layout.
- **Query**: One of 13 predefined climate policy questions representing lobbying subtopics (see Appendix C).
- **Evidence**: A human-extracted text snippet relevant to the query, from the PDF.
- **Stance**: A discrete label from the set $\{-2, -1, 0, +1, +2\}$ representing opposition through support.

- **Comment**: A short justification written by an analyst explaining the stance assignment.
- **Metadata**: Includes the corporate entity, date, and region associated with the PDF

F Parsing and Chunking Configuration

Docling

The following settings were used for parsing all documents with the Docling toolkit:

- **Device:** `mps` with 8 threads.
- **OCR:** Enabled using `easyocr` with:
  - Full-page OCR forced
  - Minimum confidence threshold: 0.5
- **Table structure extraction:** Enabled with:
  - Cell matching disabled
  - TableFormer mode set to `accurate`

PyMuPDF

The PyMuPDF parser was used with the default settings.

Semantic Chunking

The semantic chunking module was implemented using the Python library `CHONKIE` (Minhas and Nigam, 2025), and configured with the following options:

- **Model:** `minishlab/potion-base-8M`
- **Similarity threshold:** 0.75
- **Double-pass merge:** Enabled
- **Chunk size:** 1536 tokens
- **Device:** `mps`

Layout Chunking

The layout chunking module was configured with the following options:

- **Minimum chunk length:** 30 words.
- **Chunk boundary heuristic rules:**
  - Headers are merged with the subsequent text block to preserve continuity.
  - Subsequent headers are concatenated into a single header.
  - Tables are treated as atomic units.
  - Paragraphs shorter than the Minimum chunk length are merged with previous paragraphs.
  - Paragraphs ending with colons are merged with previous paragraphs unless the latter begins with a header marker.

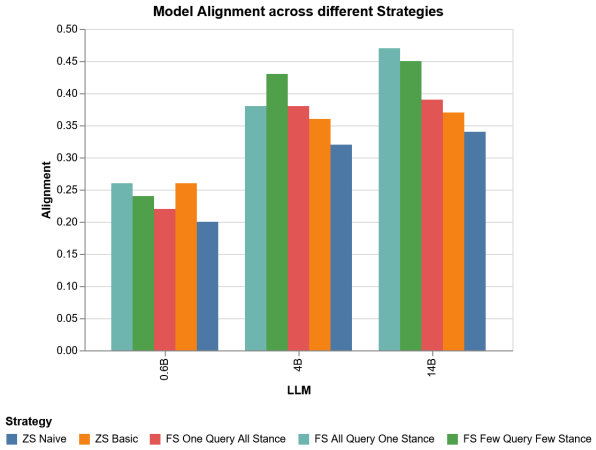


Figure 3: Model Alignment on balanced evaluation subset

G LLM Evaluation

The reported evaluations were performed on a sample of 2018 evidences extracted from 10 companies. This is an unbalanced evaluation set which is more representative of what an analyst might observe.

Figure 3 shows the same evaluation on a balanced set of 1000 samples where for each stance we have 200 samples. Interestingly we observe similar results.

H When Retrieval Outperforms Ground Truth Results

Metric	FR	AR	BM
Faithfulness	0.49	0.57	0.52
Helpfulness	0.65	0.58	0.54
Conciseness	0.77	0.82	0.88
Outperform (%)	46.2	30.8	23.1

Table 6: Average oracle scores for retrieval strategies that outperformed Ground Truth in stance generation.

GLiNER2: An Efficient Multi-Task Information Extraction System with Schema-Driven Interface

Urchade Zaratiana, Gil Pasternak, Oliver Boyd
George Hurn-Maloney, Ash Lewis
Fastino AI
{uz,gil,o8,g,ash}@fastino.ai

Abstract

Information extraction (IE) is fundamental to numerous NLP applications, yet existing solutions often require specialized models for different tasks or rely on computationally expensive large language models. We present GLiNER2, a unified framework that enhances the original GLiNER architecture to support named entity recognition, text classification, and hierarchical structured data extraction within a single efficient model. Built pretrained transformer encoder architecture, GLiNER2 maintains CPU efficiency and compact size while introducing multi-task composition through an intuitive schema-based interface. Our experiments demonstrate competitive performance across extraction and classification tasks with substantial improvements in deployment accessibility compared to LLM-based alternatives. We release GLiNER2 as an open-source pip-installable library with pre-trained models and documentation at github.com/fastino-ai/GLiNER2.

1 Introduction

Information extraction (IE) (Okurowski, 1993; Weischedel et al., 1996) represents one of the most fundamental and practically important tasks in natural language processing, involving the identification and extraction of structured information from unstructured text. While large language models (OpenAI, 2024) have demonstrated remarkable capabilities across various IE tasks (Wang et al., 2025; Han et al., 2024), their deployment presents significant practical challenges that limit their accessibility and adoption. Smaller models (Touvron et al., 2023; Jiang et al., 2023; Yang et al., 2025) (eg. *Llama-2-7b*) require GPU acceleration to achieve reasonable inference speeds, making CPU-based deployment prohibitively slow for production use. The availability of GPU resources can be a barrier for organizations and researchers operating under resource constraints.

Beyond computational requirements, LLMs present additional deployment challenges. Although API costs have decreased significantly in recent years, they can pose serious privacy and security concerns (Shanmugarasa et al., 2025; Wu et al., 2025), particularly when processing sensitive data containing personally identifiable information (PII), financial records, or proprietary business information. Many organizations in the healthcare, finance, and government sectors require on-premises deployment to maintain data sovereignty and comply with regulations such as GDPR, HIPAA, or industry-specific compliance requirements (Lareo, 2023; Zhang et al., 2024; CNIL, 2024). Furthermore, the recurring costs associated with API usage may remain prohibitive for researchers, startups, and practitioners in developing countries, creating inequitable access to advanced NLP capabilities. Our goal is to address these issues specifically in the context of information extraction.

GLiNER (Zaratiana et al., 2024) was proposed to address these fundamental limitations specifically for named entity recognition (NER) tasks. GLiNER (Zaratiana et al., 2024) introduced a paradigm shift by enabling zero-shot NER using a small transformer encoder architecture trained on diverse, LLM-annotated datasets (Zhou et al., 2024; Bogdanov et al., 2024). This approach yielded remarkable results, achieving performance that matched or surpassed contemporary LLMs while running efficiently on standard CPU hardware without requiring GPUs. Furthermore, it maintains a parameter count under 500 million, enabling deployment in edge computing scenarios, resource-constrained environments, and privacy-sensitive applications. GLiNER gained particular traction in the PII redaction domain (Segbroeck, 2024; Presidio, 2024), where its combination of competitive performance, CPU efficiency, and straightforward local deployment made it an ideal solution for han-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 130–140

Characteristic	GLiNER	GLiNER2	Open LLMs	Closed LLMs
Features				
Scope	NER only	Various IE & Classification	General	General
Label description	✗	✓	✓	✓
CPU Deployment	✓	✓	✗	✗
Privacy Preserving	✓	✓	✓	✗
No API Costs	✓	✓	✓	✗
Fine-tuning Support	✓	✓	✓	∼
Technical Specifications				
Parameters	195M	205M	7B-175B	Unknown
Model Architecture	Encoder	Encoder	Decoder	Decoder
Context Length	512 tokens	2048 tokens	2K-1M tokens	8K-10M tokens
Usage & Licensing				
License Type	Apache 2.0	Apache 2.0	Various	Proprietary
Commercial Use	✓	✓	∼	✓

Table 1: Comprehensive comparison across system categories. ✓= full support, ∼= partial/limited support, ✗= no support.

dling sensitive data.

Following GLiNER's release, several specialized adaptations emerged for different information extraction tasks. GLiREL (Boylan et al., 2025) extended the approach to relation extraction, while GLiClass (Knowledgator, 2025) adapted it for zero-shot text classification. Domain-specific variants included GLiNER-BioMed (Yazdani et al., 2025) for biomedical entity recognition, OpenBioNER (Cocchieri et al., 2025) for lightweight biomedical NER through entity type descriptions, and GLiDRE (Armingaud, 2025) for document-level relation extraction in French. However, each adaptation required *separate model development and deployment*, leading to fragmentation as practitioners needed multiple specialized models for comprehensive information extraction pipelines.

In this work, we introduce GLiNER2, a Python library that transforms the focused capabilities of its predecessors into a *universal information extraction system*. Rather than requiring separate models like GLiNER[1], GLiREL[2], and GLiClass[3], GLiNER2 unifies entity recognition, structured extraction, and text classification within a single architecture. GLiNER2 maintains the core efficiency, running on CPU, while dramatically expanding functionality beyond simple NER to support: entity recognition with natural language type descriptions and nested/overlapping spans, document-level classification with configurable single or multi-label outputs, and complex extraction schemas that capture hierarchical structures with parent-child relationships and repeated patterns. The library's Python API allows developers to define extraction schemas declaratively, compose multiple tasks in a single inference call, and deploy models with just a few lines of code. By unifying these capabilities, GLiNER2 replaces multiple specialized models with a *single efficient solution*. GLiNER2 is available through pip installation (`pip install gliner2`) with pre-trained weights hosted on Hugging Face, and it is released under the *Apache 2.0* license.

2 System design

Our architecture builds upon the foundational design principles of the original GLiNER (Zaratiana et al., 2024), which prompts a pretrained transformer encoder (Devlin et al., 2019; He et al., 2023) with entity types for zero-shot named entity recognition. We extend this prompting approach to handle more complex schemas that encompass multiple information extraction tasks. The core innovation lies in our unified input formulation that enables diverse extraction tasks through carefully designed prompt templates. The general input format follows:

`[Task Prompt]` ⊕ `[SEP]` ⊕ `[Input Text]`

where ⊕ denotes concatenation. The `[Task Prompt]` specifies what to extract (e.g., entity types like "person, location" or class labels like "positive, negative"), `[SEP]` is a special separator token, and `[Input Text]` is the text to be analyzed, which is a sequence of text tokens

[1] https://github.com/urchade/GLiNER
[2] https://github.com/jackboyla/GLiREL
[3] https://huggingface.co/knowledgator/GLiClass

Dataset	Task Type	# Labels	GPT-4o OpenAI (2024) >100B	GLiClass Knowledgator (2025) 190M	DeBERTa-v3 Laurer et al. (2024) 435M	GLiNER2 *Our model* 205M
SNIPS	Intent	7	0.97	0.80	0.77	0.83
Banking77	Intent	77	0.78	0.21	0.42	0.70
Amazon Intent	Intent	31	0.72	0.51	0.59	0.53
SST-2	Sentiment	2	0.94	0.90	0.92	0.86
IMDB	Sentiment	2	0.95	0.92	0.89	0.87
AG News	Topic	4	0.85	0.68	0.68	0.74
20 Newsgroups	Topic	20	0.68	0.36	0.54	0.49
Average	—	—	*0.84*	0.63	0.69	**0.72**

Table 2: Zero-shot text classification performance across various benchmarks.

$x_1, x_2, \ldots, x_N$. Complete task prompt formats for each extraction type are detailed in Appendix A.

GLiNER2 comprises the following tasks:

- **Entity Recognition:** We support entity type descriptions alongside labels, allowing richer semantic understanding through natural language definitions.
- **Hierarchical Structure Extraction:** We introduce structured schemas that capture parent-child relationships between entities and their attributes, enabling extraction of complex nested information.
- **Text Classification:** We add text classification capabilities with support for both single-label and multi-label, along with label description.
- **Task Composition:** Most significantly, we enable composition of multiple extraction tasks within a single forward pass, allowing simultaneous entity recognition, text classification, and structured extraction with shared contextual understanding.

This unified approach maintains the efficiency advantages of the original GLiNER while dramatically expanding its capabilities to handle diverse information extraction scenarios. Detailed architectural specifications and mathematical formulations are provided in Appendix A.

3 Experiments

3.1 Training Data

We trained our model on 254,334 examples, combining *real-world documents* and *synthetic data*, with balanced coverage of entity recognition, hierarchical extraction, and classification tasks. The real-world set consists of 135,698 documents from *news articles*, *Wikipedia*, *legal texts*, *PubMed abstracts*, and *ArXiv papers*, representing a wide range of writing styles and entity types. All documents were automatically annotated with *GPT-4o* using task-specific prompts and validated for quality. To address gaps and improve robustness, we generated 118,636 *synthetic examples* with *GPT-4o* targeting common business and personal use cases, including *email threads*, *text messages*, *professional documents*, *social media posts*, *transactional data*, and *domain-specific texts*; each synthetic example includes complete annotations for all tasks to support effective multi-task learning. Full dataset statistics and distribution are provided in Appendix B.1.

3.2 Results

We conducted comprehensive zero-shot evaluations on standard benchmarks for both text classification and named entity recognition to assess the effectiveness of our approach. Hierarchical structure extraction was not evaluated due to the absence of established zero-shot benchmarks for this task type, which we plan to address in future work. Details on all baseline models and evaluation protocols are provided in Appendix B.

We evaluate zero-shot classification on seven public benchmarks covering sentiment (*SST-2* (Socher et al., 2013), *IMDB* (Maas et al., 2011)), intent (*SNIPS* (Coucke et al., 2018), *Banking77* (Casanueva et al., 2020), *Amazon-Intent* (FitzGerald et al., 2022)), and topic classification (*AG News* (Zhang et al., 2016), *20 Newsgroups* (Lang, 1995)). **GLiNER2 achieves the highest average accuracy among open-source baselines**, outperforming GLiClass on five datasets and DeBERTa-v3 on three. It performs particularly well on intent

Dataset	GPT-4o	GLiNER-M	GLiNER2
AI	0.547	0.518	0.526
Literature	0.561	0.597	0.564
Music	0.736	0.694	0.632
Politics	0.632	0.686	0.679
Science	0.518	0.581	0.547
Average	0.599	0.615	0.590

Table 3: Zero-shot F1 scores on CrossNER benchmark.

classification, scoring 0.83 on *SNIPS* and 0.70 on *Banking77*, compared to DeBERTa's 0.77 and 0.42, respectively. On *Amazon-Intent* and *20 Newsgroups*, GLiNER2 trails DeBERTa-v3 slightly (by 6 and 5 points), but GLiNER2 is significantly faster as shown in Table 4. On sentiment benchmarks, GLiNER2 scores 0.86–0.87, close to DeBERTa-v3's 0.89–0.92 and within 10 points of GPT-4o. While GPT-4o leads across all tasks, this superior performance is expected given its substantially larger scale and extensive pretraining on diverse text corpora. Overall, GLiNER2 offers competitive accuracy across a range of classification settings and consistently closes the gap between task-specific baselines and large proprietary models.

For NER evaluation, we use the *CrossNER* benchmark (Liu et al., 2020), which measures zero-shot generalization across five specialized domains: *AI*, *Literature*, *Music*, *Politics*, and *Science*. As shown in Table 3, **GLiNER2 closely matches GPT-4o in overall F1 score** (0.590 vs. 0.599) and achieves higher scores in *AI* (0.526 vs. 0.547) and *Literature* (0.564 vs. 0.561). While GLiNER2 trails GLiNER-M in categories like *Science* and *Music*, it maintains strong performance in *Politics* (0.679), suggesting robustness across diverse entity types. Considering that GLiNER2 is a general-purpose model supporting multiple tasks, this level of NER performance with only modest drop-offs compared to a dedicated entity recognition system, demonstrates the effectiveness of our unified architecture.

3.3 Efficiency

We evaluate GLiNER2's computational efficiency by measuring inference latency on text classification tasks across different numbers of labels. All models are evaluated on *CPU* except GPT-4o, which uses the *OpenAI API*. Table 4 presents latency measurements in milliseconds for varying numbers of classification labels. GLiNER2 demonstrates **strong computational efficiency**, achieving latency comparable to GLiClass while providing significantly better performance than DeBERTa-based zero-shot classification. The key advantage becomes evident when comparing against DeBERTa, which performs a *separate forward pass for each label*, resulting in *linear scaling* with the number of labels and substantially higher latency (**6.8× slower** with 20 labels). In contrast, GLiNER2 processes *all labels simultaneously* in a single forward pass, maintaining consistent performance regardless of label count. Both GLiNER2 and GLiClass achieve approximately **2.6× speedup over GPT-4o** while running on standard CPU hardware, demonstrating the practical advantages of compact, specialized models for production deployment scenarios where *latency and computational resources* are critical considerations.

#L	GPT-4o	DeBERTa	GLiClass	GLiNER2
5	358	1714	**137**	**130**
10	382	3404	**131**	**132**
20	425	6758	**140**	**163**
50	463	16897	**190**	**208**
x	1.00×	0.10×	2.75×	2.62×

Table 4: CPU Latency (ms) comparison for text classification with varying number of labels.

4 Artifacts

4.1 Python Package

We provide a Python package that makes GLiNER2 accessible through an intuitive API. The `gliner2` library can be easily installed via pip and provides seamless integration with the Hugging Face ecosystem for model distribution and loading. The model can be loaded using the standard `.from_pretrained` method, with weights hosted on Hugging Face Hub for convenient access.

```python
# Installation: pip install gliner2
from gliner2 import GLiNER2

# Load from Hugging Face
extractor = GLiNER2.from_pretrained("gliner/gliner2-base")
```

Figure 1: Model loading.

Named Entity Recognition Named entity recognition can be performed through multiple approaches to accommodate different use cases and complexity levels. The simplest method requires only the input text and a list of target entity types. Moreover, users can provide entity types with natural language descriptions using a dictionary format,

where keys represent entity types and values contain descriptive text that helps the model better understand the extraction target. The process and various usage patterns are illustrated in Figure 2.

```
text = "Apple Inc. CEO Tim Cook announced new products in Cupertino."

entities = ["company", "person", "location", "product"]
results = extractor.extract_entities(text, entities)
# {'entities': {'company': ['Apple Inc.'],
#               'person': ['Tim Cook'],
#               'location': ['Cupertino']}}

entity_descriptions = {
    "company":  "Business organizations and corporations",
    "person":   "Names of individuals including executives",
    "location": "Geographical places including cities"
}
results = extractor.extract_entities(text, entity_descriptions)
```

Figure 2: GLiNER2 for Named Entity Recognition with simple and enhanced approaches

Hierarchical Structure Extraction Hierarchical structure extraction is performed by defining a schema as shown in Figure 3. The schema defines a parent entity (termed a *structure*) containing multiple child fields using GLiNER2's field specification syntax. Each field follows the pattern `field_name::type::description`, where `type` specifies either `str` for single values or `list` for multiple values. Fields may incorporate choice constraints through the format `field_name::[option1|option2]::type`, exemplified by the category field restricted to *electronics*, *software*, or *hardware*.

```
text = "The new MacBook Pro costs $1999..."

product_schema = {
    "product": [
        "name::str::Product name and model",
        "price::str::Product cost",
        "features::list::Key product features",
        "category::[electronics|software|hardware]::str"
    ]
}
results = extractor.extract_json(text, product_schema)
```

Figure 3: Hierarchical structure extraction with field constraints and descriptions

The framework supports multiple structures within a single schema for complex extraction scenarios. For instance, Figure 4 shows how users can define two structures, *product* and *company*, in a single query. This enables simultaneous extraction of product details (*name* and *price*) alongside company information (*name* and *headquarters*), all processed efficiently in a single forward pass.

Text Classification Like NER, text classification functionality provides both streamlined and highly customizable interfaces to accommodate various application requirements. For quick deployment, users need only provide the input text and a dictionary mapping task names (e.g., "sentiment") to

```
text = "Apple Inc., based in Cupertino..."

multi_schema = {
    "product": [
        "name::str",
        "price::str"
    ],
    "company": [
        "name::str",
        "headquarters::list"
    ]
}
results = extractor.extract_json(text, multi_schema)
```

Figure 4: Composing multiple hierarchical structures in a single schema

lists of classification labels, as shown in the first example of Figure 5. For more sophisticated applications, the library supports extensive customization options including label descriptions and multi-label classification capabilities. When multi-label classification is enabled, the model applies sigmoid activation to allow multiple simultaneous label assignments, while single-label tasks use softmax normalization for mutually exclusive predictions.

```
text = "This movie was absolutely fantastic! Great acting and plot."

labels = ["positive", "negative", "neutral"]
results = extractor.classify_text(text, {"sentiment": labels})
# {'sentiment': 'positive'}

tasks = {
    "aspects": {
        "labels": ["acting", "plot", "visuals", "music"],
        "multi_label": True,
        "descriptions": {
            "acting":  "Quality of character performances",
            "plot":    "Story structure and narrative",
            "visuals": "Cinematography and visual effects",
            "music":   "Soundtrack and audio design"
        }
    }
}
results = extractor.classify_text(text, tasks)
# {'aspects': ['acting', 'plot']}
```

Figure 5: Text classification with simple and advanced configuration options

The library supports multiple classification tasks within a single call, as demonstrated in Figure 6. Each classification task can be independently customized with features such as label descriptions and multi-label settings.

```
results = extractor.classify_text(text, {
    "sentiment": ["positive", "negative", "neutral"],
    "genre":     ["comedy", "drama", "action", "thriller"]
})
```

Figure 6: Simultaneous multi-task classification.

Task Composition A key feature of the library is its ability to efficiently compose multiple extraction tasks within a single unified framework. Figure 8 demonstrates how to construct a comprehensive schema that combines entity recognition, text classification, and structured extraction seamlessly in one inference call.

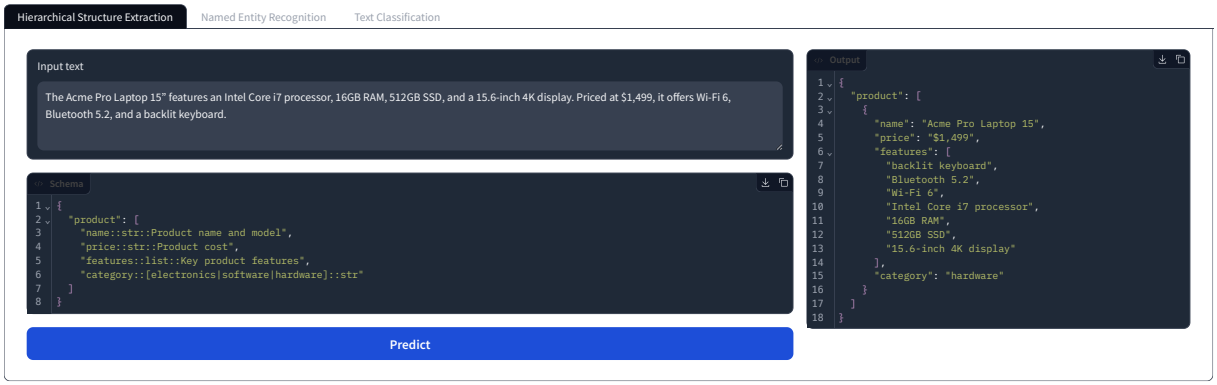


Figure 7: GLiNER2 Gradio demo interface showing hierarchical structure extraction.

```python
# Multi-task extraction in a single forward pass
schema = (extractor.create_schema()
    # Named Entity Recognition
    .entities(["person", "company", "product", "location", "price"])

    # Text Classification
    .classification("sentiment", ["positive", "negative", "neutral"])
    .classification("urgency", ["low", "medium", "high"])

    # Hierarchical Structure Extraction
    .structure("product_info")
        .field("name", dtype="str", description="Product name")
        .field("price", dtype="str", description="Product cost")
        .field("features", dtype="list", description="Key features")
        .field("company", dtype="str", description="Manufacturer")
)

# Extract all information simultaneously
results = extractor.extract(text, schema)
```

Figure 8: Comprehensive task composition combining all extraction types

4.2 Interactive Gradio Demo

We provide a web-based demonstration interface that allows users to interact with GLiNER2 without writing code. The demo enables real-time experimentation with entity types, classification labels, descriptions and other parameters. The interface consists of three tabs corresponding to GLiNER2's core capabilities. Figure 7 shows the hierarchical structure extraction tab, where users can define schemas with multiple fields and data types to extract structured information from text.

5 Related Work

Several frameworks have addressed information extraction tasks across different domains and approaches.

Traditional NLP Libraries: spaCy (Honnibal et al., 2020), Stanford CoreNLP (Manning et al., 2014), Stanza (Qi et al., 2020) provide comprehensive toolkits for named entity recognition, part-of-speech tagging, and dependency parsing. However, these frameworks require separate models for each task and lack unified architectures, and often does not generalize to unseen labels.

LLM-based Extraction: XNLP (Fei et al., 2024) demonstrated using large language models for diverse IE tasks through prompting strategies, while NuExtract (NuMind, 2024) focused on fine-tuning for JSON extraction. These approaches achieve strong performance but require significant computational resources and GPU inference.

Encoder-based Approaches: GLiNER (Zaratiana et al., 2024) introduced an efficient paradigm leveraging pretrained encoders fine-tuned on synthetic data for zero-shot named entity recognition, achieving fast CPU inference with competitive accuracy. This approach inspired subsequent work including GLiClass (Knowledgator, 2025) for text classification and GLiREL (Boylan et al., 2025) for zero-shot relation extraction. GLiNER2 extends this line of work by integrating multiple tasks within a single efficient framework, enabling multi-task composition while maintaining the computational advantages of compact encoder-based models.

6 Conclusion

We presented GLiNER2, which unifies entity recognition, text classification, and hierarchical extraction in a single CPU-efficient model. Unlike existing approaches requiring separate models per task, GLiNER2 enables multi-task extraction through declarative schemas while maintaining under 500M parameters for practical deployment. We release GLiNER2 as an open-source Python library under Apache 2.0 license, with pre-trained weights on Hugging Face. By combining efficiency with versatility, we hope our library makes advanced information extraction accessible for both research and production use.

References

Robin Armingaud. 2025. Glidre: Modèle généraliste pour l'extraction de relations à l'échelle de documents. In *EGC-Atelier TextMine*.

Sergei Bogdanov, Alexandre Constantin, Timothée Bernard, Benoit Crabbé, and Etienne P Bernard. 2024. NuNER: Entity recognition encoder pre-training via LLM-annotated data. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11829–11841, Miami, Florida, USA. Association for Computational Linguistics.

Jack Boylan, Chris Hokamp, and Demian Gholipour Ghalandari. 2025. GLiREL - generalist model for zero-shot relation extraction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8230–8245, Albuquerque, New Mexico. Association for Computational Linguistics.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.

CNIL. 2024. AI and GDPR: the CNIL publishes new recommendations to support responsible innovation.

Alessio Cocchieri, Giacomo Frisoni, Marcos Martínez Galindo, Gianluca Moro, Giuseppe Tagliavini, and Francesco Candoli. 2025. Open-BioNER: Lightweight Open-Domain Biomedical Named Entity Recognition Through Entity Type Description. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 818–837, Albuquerque, New Mexico. Association for Computational Linguistics.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Preprint*, arXiv:1805.10190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hao Fei, Meishan Zhang, Min Zhang, and Tat-Seng Chua. 2024. XNLP: An Interactive Demonstration System for Universal Structured NLP. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 19–30, Bangkok, Thailand. Association for Computational Linguistics.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *Preprint*, arXiv:2204.08582.

Ridong Han, Chaohao Yang, Tao Peng, Prayag Tiwari, Xiang Wan, Lu Liu, and Benyou Wang. 2024. An empirical study on information extraction using large language models. *Preprint*, arXiv:2305.14450.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Knowledgator. 2025. gliclass-base-v1.0-lw. `https://huggingface.co/knowledgator/gliclass-base-v1.0-lw`.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*.

Xabier Lareo. 2023. Large language models (llm). `https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm_en`. European Data Protection Supervisor, TechSonar.

Moritz Laurer, Wouter van Atteveldt, Andreu Casas, and Kasper Welbers. 2024. Building efficient universal classifiers with natural language inference. *Preprint*, arXiv:2312.17543.

Zihan Liu, Yan Xu, Tiezheng Yu, Wenliang Dai, Ziwei Ji, Samuel Cahyawijaya, Andrea Madotto, and Pascale Fung. 2020. Crossner: Evaluating cross-domain named entity recognition. *Preprint*, arXiv:2012.04373.

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.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

NuMind. 2024. Nuextract: A framework for structured data extraction. https://numind.ai/blog/nuextract-a-foundation-model-for-structured-extraction.

Mary Ellen Okurowski. 1993. Information extraction overview. In *TIPSTER TEXT PROGRAM: PHASE I: Proceedings of a Workshop held at Fredricksburg, Virginia, September 19-23, 1993*, pages 117–121, Fredericksburg, Virginia, USA. Association for Computational Linguistics.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Presidio. 2024. Using gliner as an external pii model.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Maarten Van Segbroeck. 2024. Gliner models for pii detection through fine-tuning on gretel-generated synthetic documents.

Yashothara Shanmugarasa, Ming Ding, Mahawaga Arachchige Pathum Chamikara, and Thierry Rakotoarivelo. 2025. Sok: The privacy paradox of large language models: Advancements, privacy risks, and mitigation.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, Guoyin Wang, and Chen Guo. 2025. GPT-NER: Named entity recognition via large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4257–4275, Albuquerque, New Mexico. Association for Computational Linguistics.

Ralph Weischedel, Sean Boisen, Daniel Bikel, Robert Bobrow, Michael Crystal, William Ferguson, Allan Wechsler, and The PLUM Research Group. 1996. Progress in information extraction. In *TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996*, pages 127–138, Vienna, Virginia, USA. Association for Computational Linguistics.

Yuhao Wu, Evin Jaff, Ke Yang, Ning Zhang, and Umar Iqbal. 2025. An in-depth investigation of data collection in llm app ecosystems. *Preprint*, arXiv:2408.13247.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Anthony Yazdani, Ihor Stepanov, and Douglas Teodoro. 2025. Gliner-biomed: A suite of efficient models for open biomedical named entity recognition. *Preprint*, arXiv:2504.00676.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2024. GLiNER: Generalist model for named entity recognition using bidirectional transformer. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5364–5376, Mexico City, Mexico. Association for Computational Linguistics.

Dawen Zhang, Pamela Finckenberg-Broman, Thong Hoang, Shidong Pan, Zhenchang Xing, Mark Staples, and Xiwei Xu. 2024. Right to be forgotten in the era of large language models: Implications, challenges, and solutions. *Preprint*, arXiv:2307.03941.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification. *Preprint*, arXiv:1509.01626.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2024. UniversalNER: Targeted distillation from large language models for open named entity recognition. In *The Twelfth International Conference on Learning Representations*.

A Architecture Details

Special Token Vocabulary Our architecture employs a set of learned special tokens, each serving a specific semantic role:

- [P] (Prompt): Marks the beginning of task specifications, signaling the model to interpret subsequent tokens as task metadata
- [E] (Entity): Precedes each entity type in NER tasks to create distinct embeddings for entity categories
- [C] (Child/Component): Indicates attribute fields in hierarchical structures and establishes parent-child relationships
- [L] (Label): Denotes classification options, with each label receiving a unique embedding for scoring
- [SEP] (Separator): Delimits different input segments to prevent information leakage between task specifications and content.

These tokens are randomly initialized and learned during training, allowing the model to develop task-specific representations.

Named Entity Recognition NER tasks follow the input format:

[P] entities ([E] e_1 [E] e_2 ... [E] e_n) [SEP] $x_1, x_2, \ldots, x_N$

During extraction, each [E] token generates an embedding representing its entity type. The model creates representations for all possible text spans up to a maximum width, then computes matching scores between span-entity pairs using:

$$\text{score}(s_i, e_j) = \text{sim}(\mathbf{h}_{s_i}, \mathbf{h}_{e_j}) \tag{1}$$

where $\mathbf{h}_{s_i}$ is the span representation, $\mathbf{h}_{e_j}$ is the entity type embedding, and $\text{sim}(\cdot, \cdot)$ is the dot product with sigmoid activation.

For example, given [P] entities ([E] person [E] location) and text *"John works in Paris"*, all span candidates (e.g., *"John"*, *"works"*, *"Paris"*, *"works in"*) are scored against the entity type embeddings (i.e., representations of each [E] token). Spans with a predicted probability above 0.5 for any entity type are selected as entities.

Hierarchical Structure Extraction Hierarchical extraction uses the format:

[P] *parent* ([C] a_1 [C] a_2 ... [C] a_m) [SEP] $x_1, x_2, \ldots, x_N$

The process operates in two stages. First, an MLP processes the [P] token embedding to predict the number K of parent entity instances in the text. This MLP performs 20-class classification (for counts 0-19), trained using the ground-truth instance counts during training. Then, the model generates K distinct representations for each attribute by conditioning the [C] token embeddings on learned occurrence ID embeddings. Specifically, for each instance $k \in \{1, ..., K\}$, the model combines the base [C] embeddings with occurrence-specific embeddings learned during training, producing unique representations for each instance-attribute pair. These $K \times m$ representations are matched against text spans using the same scoring mechanism as NER, ensuring each instance maintains separate attribute values. Consider the structured extraction task:

[P] product ([C] name [C] price)

Given input text: *"iPhone costs $999. Galaxy is $899."* the model processes this in three steps:

1. **Count Prediction**: The MLP count predictor processes the [P] token embedding and outputs $K = 2$, indicating two product instances are present in the text.
2. **Representation Generation**: The count embedding layer generates K sets of conditioned representations for each attribute field. This produces two distinct embeddings for [C] name and two for [C] price, with each pair corresponding to one product instance.
3. **Span Extraction**: Each conditioned representation computes similarity scores with all possible text spans as for NER. The model selects the highest-scoring spans for each field while maintaining instance coherence:
   - Instance 1: {name: "iPhone", price: "$999"}
   - Instance 2: {name: "Galaxy", price: "$899"}

This parallel processing enables efficient extraction of multiple structured entities while preserving the semantic relationships between fields within each instance.

Text Classification Classification tasks use the format:

`[P] task ([L]` ℓ_1 `[L]` ℓ_2 `... [L]` ℓ_k`) [SEP]` $x_1, x_2, \ldots, x_N$

Each `[L]` token produces a label-specific embedding that is refined through a classification head. Specifically, for each label ℓ_i, the model computes:

$$\text{logit}_i = \text{MLP}(\mathbf{h}_{\ell_i}) \tag{2}$$

where $\mathbf{h}_{\ell_i}$ is the contextualized embedding from the `[L]` token for label ℓ_i, and MLP is a multi-layer perceptron that projects these embeddings to scalar logits representing label-text compatibility. Single-label tasks apply softmax over all logits to select the highest-probability label, while multi-label scenarios use sigmoid activation on each logit independently. Consider the text classification task:

`[P] sentiment ([L] positive [L] negative [L] neutral)`

Given input text: *"This movie is amazing!"*. The model processes this in three steps:

1. **Label Embedding Generation**: Each `[L]` token creates a distinct embedding for its corresponding label (*positive*, *negative*, *neutral*).

2. **Classification Head**: The label embeddings are projected through an MLP to produce classification logits, which are then normalized using softmax activation for single-label prediction.

3. **Label Selection**: The model selects the highest-scoring label, predicting *"positive"* for the given input text.

For multi-label scenarios, sigmoid activation replaces softmax, allowing multiple labels to be selected simultaneously.

Task Composition Multiple tasks can be composed for efficient multi-task inference using:

$[\texttt{Task}_1] \oplus \texttt{[SEP]} \oplus [\texttt{Task}_2] \oplus \ldots \oplus \texttt{[SEP]} \oplus [x_1, x_2, \ldots, x_N]$

This enables simultaneous execution of multiple extraction tasks in a single forward pass. For instance, combining NER and sentiment classification on "Steve Jobs loved the iPhone" extracts entities {person: ["Steve Jobs"], product: ["iPhone"]} and sentiment "positive" in one computation, improving efficiency over separate model runs.

B Experimental setup

Baselines We evaluate our approach against several strong baselines. As an upper bound, we use GPT-4o across all tasks. For classification tasks, we compare against two state-of-the-art open-source models with comparable parameter counts: (1) GLiClass (*knowledgator/gliclass-base-v1.0*) (Knowledgator, 2025), a classification-specific adaptation of GLiNER, and (2) DeBERTa-v3-base-zeroshot (*MoritzLaurer/deberta-v3-large-zeroshot-v2.0*) (Laurer et al., 2024), the de facto standard for zero-shot classification on Hugging Face. For NER tasks, we use GLiNER-M performance as reported in Zaratiana et al. (2024), which represents the current state-of-the-art for generalist entity recognition.

Hyperparameters We train our model for 5 epochs using the AdamW optimizer with differential learning rates: 2×10^{-5} for task-specific layers and 1×10^{-5} for the encoder backbone. This differential approach allows fine-tuning of pretrained representations while enabling faster adaptation of task-specific components. We apply weight decay of 0.01 for regularization and gradient clipping at 1.0 to ensure training stability. The learning rate schedule includes 1,000 warmup steps with linear scaling. Table 5 summarizes the training configuration.

Hyperparameter	Value
Epochs	5
Optimizer	AdamW
Learning rate (backbone)	1×10^{-5}
Learning rate (task layers)	2×10^{-5}
Weight decay	0.01
Warmup steps	1,000
Gradient clipping	1.0

Table 5: Training hyperparameters used across all experiments.

B.1 Training Data

Our training dataset comprises 254,334 examples combining real-world texts and synthetic data. Table 6 shows the distribution across domains.

Real-world data (135,698 examples) was collected from news articles, Wikipedia, legal documents, ArXiv papers, and PubMed abstracts. All texts were annotated using GPT-4o for entity recognition, hierarchical extraction, and classification

Domain	Count
Real-world Data	
Law	19,798
PubMed	16,400
Wikipedia	17,909
ArXiv	7,135
News	74,456
Synthetic Data	
Mixed Domains	118,636
Total	254,334

Table 6: Training data distribution across domains.

tasks. Synthetic data (118,636 examples) was generated using GPT-4o to cover practical use cases including emails, text messages, resumes, social media posts, e-commerce orders, banking records, and sports commentary. Each example includes annotations for all applicable task types.

SciClaims: An End-to-End Generative System for Biomedical Claim Analysis

Raúl Ortega
Language Technology Research Lab
Expert.ai / Madrid, Spain
rortega@expert.ai

José Manuel Gómez-Pérez
Language Technology Research Lab
Expert.ai / Madrid, Spain
jmgomez@expert.ai

Abstract

We present SciClaims, an interactive web-based system for end-to-end scientific claim analysis in the biomedical domain. Designed for high-stakes use cases such as systematic literature reviews and patent validation, SciClaims extracts claims from text, retrieves relevant evidence from PubMed, and verifies their veracity. The system features a user-friendly interface where users can input scientific text and view extracted claims, predictions, supporting or refuting evidence, and justifications in natural language. Unlike prior approaches, SciClaims seamlessly integrates the entire scientific claim analysis process using a single large language model, without requiring additional fine-tuning. SciClaims is optimized to run efficiently on a single GPU and is publicly available for live interaction[1].

1 Introduction

Systematic Literature Review (SLR) plays a critical role in biomedical research and the pharmaceutical industry, supporting clinical decisions, regulatory submissions, and R&D pipelines. A central task in SLR is the validation of scientific claims, ensuring that assertions made in scientific texts are supported by prior peer-reviewed research. However, this task is labor-intensive, prone to errors, and increasingly difficult to scale as the number of scientific publications grows.

We present SciClaims, a fully automated system that addresses this challenge by providing an end-to-end scientific claim analysis pipeline in an interactive, user-friendly interface. The system extracts factual claims from scientific texts, retrieves evidence from a curated biomedical corpus, and verifies the validity of each claim using large language models (LLMs). To improve transparency, SciClaims also offers natural language rationales and highlights key supporting or refuting evidence.

SciClaims is optimized for real-world deployment and operates efficiently on a single GPU, supports high-throughput processing, and handles documents of up to 10,000 characters. It is accessible through a web-based interface, allowing users to analyze both preloaded and custom input texts.

In this demonstration, we showcase SciClaims as a useful tool to enable users to validate scientific claims in real time, facilitating trustworthy knowledge discovery in high-stakes domains like biomedicine and pharmaceuticals. A how-to video[2] and all the code[3] used in this project have been published and made publicly available.

2 Related Work

The task of analyzing scientific claims from real-world texts based on background knowledge consists of three primary components: claim extraction, evidence retrieval from a document corpus, and verifying or fact-checking the claims against the evidence (Eldifrawi et al., 2024; Vladika and Matthes, 2023). However, solving this pipeline end-to-end across all three stages remains an open challenge.

Several studies have addressed the challenge of extracting claims from scientific texts. Some frameworks based on zero-shot models (Pan et al., 2021; Wright et al., 2022) have achieved promising results, primarily focusing on generating claim datasets from raw texts to train fact-checking models in specific domains. However, these methods rely on multi-stage NLP pipelines that are prone to failure and tend to produce a large number of claims per document, making their integration into an end-to-end system difficult. In contrast, recent

[1] https://labdemos.expertcustomers.ai/health_claims (user: guest / password: acldemos2025)

[2] www.youtube.com/watch?v=jyms_Ey0YSQ

[3] www.github.com/expertailab/sciclaims-backend and www.github.com/expertailab/sciclaims-frontend

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 141–154
November 4-9, 2025

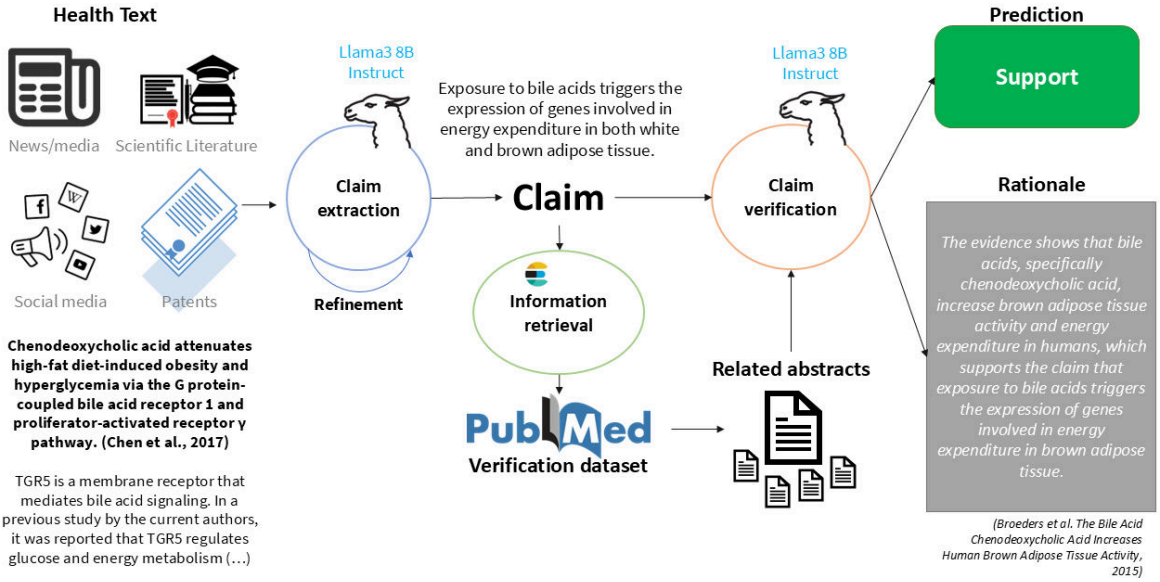


Figure 1: System Architecture.

approaches based on LLMs extract atomic factual units (Min et al., 2023; Chern et al., 2023), which serve as concise, interpretable summaries of the source texts.

For the evidence retrieval phase, dense passage retrieval methods, such as ColBERT (Khattab and Zaharia, 2020), have emerged due to their ability to retrieve highly relevant documents from large corpora with great precision. However, these methods are computationally expensive and therefore less practical for real-time, lightweight applications. Thus, simpler approaches such as BM-25 (Robertson and Zaragoza, 2009) and tools like Elasticsearch, known for their balance between retrieval quality and computational efficiency, are still preferred for deployment in production environments.

The release of several claim verification datasets has spurred the development of numerous models aimed at addressing this challenge. VERT5ERINI (Pradeep et al., 2021), PARAGRAPHJOINT (Li et al., 2021), and MultiVerS (Wadden et al., 2022) have achieved promising results on scientific benchmarks like SciFact (Wadden et al., 2020) and CLIMATE-FEVER (Diggelmann et al., 2021). However, these approaches still leave room for improvement and typically rely on large, labeled datasets, which limits their scalability across domains. In parallel, recent advances in LLMs have led to increased interest in verifying automatically generated content. Frameworks such as FactScore (Min et al., 2023), FacTool (Chern et al., 2023), and LLM Oasis (Scirè et al., 2024) focus on assessing the factuality of LLM-generated text rather than existing, human-written passages. Nonetheless, their techniques, such as extracting atomic knowledge units and using zero-shot LLMs for claim extraction and verification, can be extended to real-world scientific texts, offering a promising direction for scalable, data-efficient claim analysis.

Although individual components have seen substantial progress, an integrated system capable of seamlessly connecting these steps remains an open problem. Many existing systems, such as FactDetect (Jafari and Allan, 2024), CliVER (Liu et al., 2024), and more recently (Liang and Sontag, 2025; Vladika et al., 2025), focus on claims that are already identified, neglecting the crucial first step of extracting relevant claims from raw, real-world texts. As a result, these systems are limited to pre-identified claims rather than addressing the full pipeline of claim extraction, retrieval, and verification.

Given the complexity of this problem, our approach aims to optimize each stage of the pipeline to ensure both efficiency and accuracy in real-world biomedical claim analysis. In this work, we tap on the strengths of modern LLMs to address previous limitations, specifically in the claim extraction and verification stages, and integrate our results as a robust online system.

3 System Description

SciClaims is an interactive, end-to-end system for scientific claim analysis, comprising three main

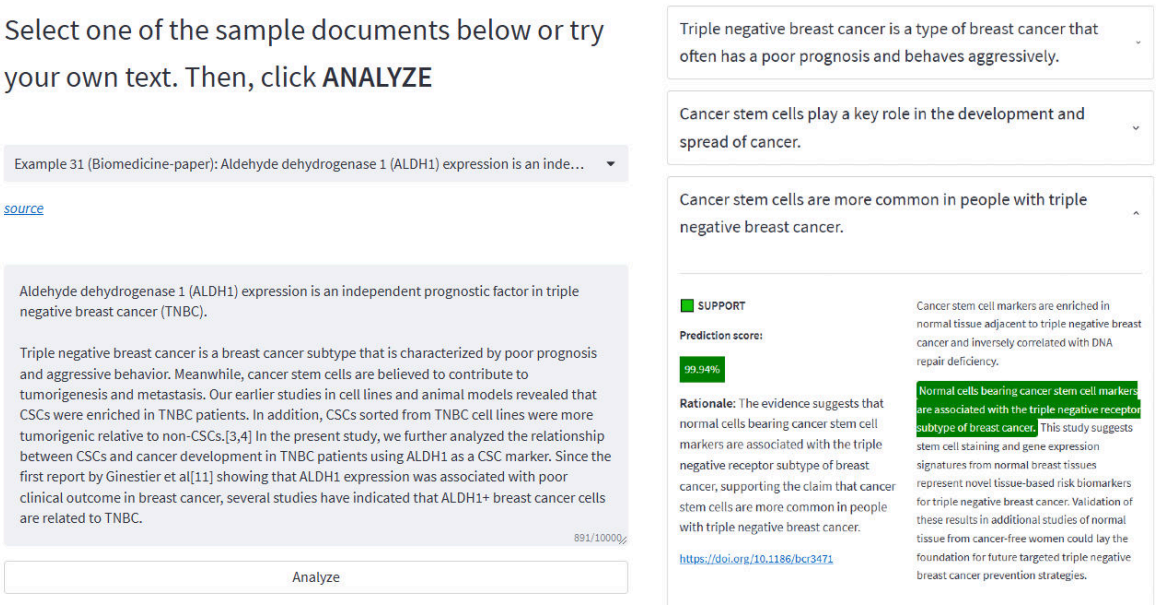


Figure 2: Screenshot of the SciClaims Demo.

components: claim extraction, evidence retrieval, and claim verification. It is optimized to run efficiently on a single 24GB VRAM GPU, enabling real-time performance via a web-based interface.

The system architecture (Figure 1) is built around a Llama3 8B Instruct model (Grattafiori et al., 2024) and an Elasticsearch-based retrieval engine. It processes biomedical or scientific text input and outputs a list of extracted claims, their verification status, relevant evidence, and a natural language rationale. The model is set up using vLLM (Kwon et al., 2023), which enables high-throughput processing in inference. Next, we present the main building blocks of SciClaims.

Claim extraction: This first module extracts potential claims from the source text. A claim is characterized by a specific set of properties detailed in section 5. The Claim Extraction module calls the Llama3 8B Instruct model twice: first, to generate an initial list of claims, and second, to refine and filter them, improving the quality of the resulting claims. All the prompts used in our pipeline are provided in Appendix A. Based on our evaluation, we made adjustments to the initial prompts to improve claim extraction performance. Further details on this refinement can be found in section 5.

Document retrieval: The second module retrieves potentially relevant documents from the verification dataset, using the claim as a query to the Elasticsearch index. The verification dataset contains 4.7 million abstracts from PubMed (2000–2022) that were curated using the Semantic Scholar's *Highly Influential Citations* metric (Valenzuela et al., 2015), ensuring that each article is backed by at least three highly influential citations. This selection criterion helps prioritize documents that have been extensively referenced in the academic community, enhancing the quality and relevance of retrieved information for verification. We chose not to filter documents based on Elasticsearch's scoring mechanism to maximize recall. Instead, the subsequent verification module is responsible for discarding irrelevant documents.

Claim verification: The final module performs fact-checking by making another call to the LLM, providing the claim along with the retrieved related documents. The model assigns one of the following three labels: 1) *SUPPORT* if the claim is verified by the document, 2) *REFUTE* if the claim is refuted by the document, or 3) *NEI* (not enough information) if the document lacks sufficient evidence or is not relevant to the claim. To improve transparency and interpretability, we also request the model to provide a rationale for its decision, including identifying the most relevant sentence(s) that support its conclusion.

4 User interface

In this section, we demonstrate the functionality of our approach through a web-based claim analysis tool that allows users to easily interact with the system and analyze claims within relevant texts,

leveraging the backend architecture presented in the previous section. The demo showcases the complete workflow, from entering a scientific passage to obtaining fact-checking results with supporting evidence and explanations.

The user interface provides a drop-down menu featuring over 30 pre-selected examples from various domains, including biomedicine papers, COVID-related news, social media, and patents (see Figure 2). These examples were chosen to represent a broad spectrum of platforms and relevant disciplines, allowing users to explore diverse contexts. By selecting any example from the list, the corresponding text will be displayed in the text box below, with a hyperlink to the original source. The text box is fully editable, enabling users to modify the content and analyze their own text. While the application can process texts up to 10,000 characters, we recommend keeping the input under 2,000 characters for faster results.

As shown in Figure 2, once the user has selected or entered a text, they can click the *Analyze* button to initiate the claim analysis process. Results are presented as a list of identified claims, each of which can be expanded to reveal detailed information. For each claim, the interface shows:

Prediction label: Whether the claim is supported (green) or refuted (red). Results labeled as *Not Enough Information* by the LLM are not presented to the user.

Prediction score: A normalized probability score that represents the confidence level of the model in its prediction. This score is derived from the statistical outputs of the model, particularly the tokens representing the label string, and it is expressed as a percentage.

Evidence: The related document selected by the retrieval module, along with its DOI. The specific sentence(s) in the abstract of the paper that were most influential for the prediction according to the LLM are highlighted.

Rationale: A justification of the reasoning behind the classification, providing additional insight into the decision-making process carried out by the model.

This interactive interface provides an intuitive and user-friendly environment for testing and exploring claim analysis in various health-related texts. The analysis goes through the entire pipeline, offering a label for the claims extracted along with the retrieved evidence. When a claim receives conflicting labels from different pieces of evidence, our system returns all relevant pairs, allowing users to assess which is more accurate. Since scientific knowledge evolves, evidence considered true at one time may later be contradicted. We recognize the value of providing all supporting and refuting evidence along with their publication dates.

5 Experimentation and Results

We evaluate our system following a three-stage approach. First, we evaluate SciClaims' modules against some baselines using SciFact as benchmark. Then, we evaluate the performance of the system at every step of the pipeline using an LLM as a judge. Finally, we perform a human evaluation for the whole system.

5.1 Evaluation in SciFact

We use SciFact, a benchmark dataset focused on biomedical literature, to evaluate SciClaims. It includes 1,400 expert-authored scientific claims, each linked to evidence-containing abstracts and annotated with a verification label.

SciFact supports two complementary evaluations: (1) claim extraction, by comparing system-generated claims to human-written ones using shared source documents, and (2) claim verification, by assessing how accurately the system labels claim-evidence pairs.

5.1.1 Claim extraction

To evaluate the claim extraction module, we consider the expert-written claims from SciFact as the ground truth for each source paragraph.

Method	Rouge1	Rouge2	RougeL	Similarity Score
Sentence tokenizer	0.3313	**0.1980**	0.3030	0.7211
(Pan et al., 2021)	0.2780	0.1334	0.2555	0.6561
(Wright et al., 2022)	0.2525	0.0762	0.2220	0.6475
Noun Phrases Gen	0.2567	0.0953	0.2293	0.6693
SciClaims	**0.3387**	0.1896	**0.3084**	**0.7250**

Table 1: Claim quality scores in SciFact

As a first baseline, we select a sentence tokenizer, treating each sentence in the source paragraph as a claim. Next, we select two baselines (Pan et al., 2021; Wright et al., 2022) that use a pipeline with two transformer models to generate claims from the source paragraph through entity extraction. The first model generates a question-answer pair, where the answer is the entity, while the second reformulates it as a claim. We also introduce a method where we propose to build the claims around noun phrases rather than named entities.

We evaluate the quality of claim generation by matching each system-generated claim with its most similar gold claim from the same paragraph. We discard pairs with a Levenshtein similarity score below 0.3, a threshold empirically tuned to balance recall and noise. For valid matches, we compute ROUGE-1, ROUGE-2, ROUGE-L, and a semantic similarity score using a DeBERTa model fine-tuned on the STS-B dataset. As shown in Table 1, SciClaims achieves the highest scores in ROUGE-1, ROUGEL, and semantic similarity, indicating more faithful and informative claim generation.

5.1.2 Claim verification

We evaluate SciClaims on SciFact's test set, comparing its label accuracy for claim-evidence pairs with existing approaches.

Architecture	Precision	Recall	F1-Score
Roberta-base*	0.4662	0.5963	0.5220
MultiVerS*	**0.7286**	**0.7321**	**0.7303**
SciClaims (LLaMA3 8B Instruct)	0.7034	0.6863	0.6788
SciClaims (Qwen2 7B Instruct)	0.6649	0.6677	0.6439
SciClaims (Phi3 Small 8K)	0.7100	0.6894	0.6525
SciClaims (OLMO2 7B Instruct)	0.6379	0.6118	0.5886

Table 2: Precision, recall and F1-Scores in Claim Verification of SciFact test set. RoBERTa-base and MultiVerS are fine-tuned models, while SciClaims is zero-shot. In parenthesis, the backbone used in each SciClaims configuration

We compare SciClaims with two strong baselines: a RoBERTa-base classifier (Liu et al., 2019) and MultiVerS (Wadden et al., 2022), a longformer-based model for claim verification. Both baselines are fine-tuned on SciFact. While MultiVerS achieves the highest F1-score, SciClaims performs competitively despite operating in a zero-shot setting. As shown in Table 2, the relatively narrow performance gap demonstrates SciClaims' strong generalization ability without task specific training. Furthermore, as shown in Table 2, we evaluate SciClaims with different LLMs with similar sizes as backbone, being LLaMA3 8B Instruct the one which offered the best performance.

5.2 Evaluation with a judge model

The second stage of our evaluation provides a comprehensive assessment of the entire system using a LLM as judge. Based on the Judge Arena leaderboard[4], we select Qwen 2.5 72B turbo as judge, since it is the first ranked model with open weights and a higher parameter count than our system's LLM (Llama3 8B Instruct). We chose its 4-bit quantization version (Qwen2.5 72B AWQ[5]) due to hardware limitations. For the evaluation sample, we randomly select 120 documents from the PubMed dataset presented in section 3. The evaluation is conducted in three phases: Claim quality, document retrieval, and claim verification and full-system evaluation.

5.2.1 Claim quality evaluation

In this phase, we evaluate the quality of the claims generated by SciClaims and compare it to other methods, using the same baselines mentioned in Section 5.1.

We devised a questionnaire consisting of eight yes/no questions (see Appendix B) to capture the desired properties in a correct claim and ask the judge model to answer it. The first question requires context from the source paragraph, while the remaining questions focus solely on the claim. Correct claims need to receive a *Yes* to all questions. Table 3 shows that our LLM-based system outperforms all other methods, scoring 15 points higher than the second-best, the sentence tokenizer.

To further enhance the results, we refined our claim extraction prompts with two optimizations:

Claim Definition Properties (CDP). Here we enhance the prompt by incorporating the characteristics of a claim, such as precision, conciseness, or check-worthiness. These characteristics are derived from the questionnaire used to evaluate the quality of the claims. The goal is to guide the LLM to adhere to these criteria when generating the list of claims.

Claim Refinement (CR): This upgrade involves a follow-up call to the LLM in order to refine the initial list of claims. For each candidate claim, we pair it with the source paragraph and ask the LLM to refine the claim based on the same criteria (precision, conciseness, check-worthiness, etc.). This step aims to eliminate poorly formed claims and reinforce the application of the specified criteria.

These two upgrades result in an additional 29-point increase in the percentage of correct claims generated by SciClaims while reducing the number of candidate claims generated by our approach. Notably, QA-oriented baselines, such as (Wright et al., 2022) and our noun phrase-based generation

[4]https://huggingface.co/spaces/AtlaAI/judge-arena

[5]huggingface.co/Qwen/Qwen2.5-72B-Instruct-AWQ

Method	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Correct Claims (%)	Candidate claims
Sentence tokenizer	**0.9987**	0.9100	0.8436	0.8709	0.9322	0.9009	0.3703	0.9596	32.46	767
(Pan et al., 2021)	0.2449	0.1303	0.1618	0.1371	0.2112	0.1169	0.0337	0.1641	3.15	445
(Wright et al., 2022)	0.8043	0.4714	0.5040	0.3842	0.5593	0.4829	0.1641	0.6249	11.28	4175
Noun Phrases Gen	0.3001	0.1562	0.1857	0.1209	0.2388	0.1320	0.0316	0.2203	2.42	**4335**
SciClaims	0.9949	0.9474	0.9718	0.9089	0.9320	0.9345	0.5238	0.9756	47.37	779
SciClaims+CDP	0.9942	0.9690	0.9845	0.9380	0.9574	0.9593	0.5930	**0.9903**	55.43	516
SciClaims+CDP+CR	0.9923	**0.9787**	**0.9884**	**0.9845**	**0.9806**	**0.9748**	**0.7810**	**0.9903**	**76.36**	516

Table 3: Claim extraction (phase 1) evaluation results with a judge model (Qwen2.5 72B AWQ).

Claims	All Claims			Correct Claims		
	R@1	R@3	R@5	R@1	R@3	R@5
Sentence tokenizer	0.5591	0.7205	0.7795	0.6265	0.7390	0.7871
(Pan et al., 2021)	0.2970	0.4653	0.5248	0.6429	0.7143	0.7143
(Wright et al., 2022)	**0.5860**	**0.7474**	**0.8038**	**0.7113**	**0.8365**	**0.8726**
Noun Phrases Gen	0.4634	0.6456	0.7013	0.6286	0.7619	0.8095
SciClaims	0.5045	0.6645	0.7135	0.5257	0.6938	0.7344
SciClaims+CDP	0.5146	0.6940	0.7466	0.5944	0.7448	0.7867
SciClaims+CDP+CR	0.4727	0.6250	0.6777	0.5102	0.6675	0.7107

Table 4: Document retrieval (phase 2) evaluation results with a judge model (Qwen2.5 72B AWQ).

method, generate significantly larger quantities of claims compared to the other methods in Table 3.

5.2.2 Document retrieval evaluation

We assess document relevance by asking the judge model whether each retrieved paragraph aids claim verification. Recall is evaluated at k = 1,3,5 retrieved documents per claim. As shown in Table 4, claims generated by (Wright et al., 2022) retrieve more potentially relevant documents than SciClaims. However, SciClaims performs well, retrieving at least one relevant document in 75% of cases when fetching five documents per claim, a common real-world scenario. This highlights its balance between claim accuracy and document relevance. QA-oriented methods generate more compact, less specific claims than LLMs, increasing the likelihood of retrieving related documents.

5.2.3 Claim verification and full-system

The verification task involves predicting the veracity of claim-evidence pairs, classifying each as *SUPPORT*, *REFUTE*, or *NOT ENOUGH INFORMATION (NEI)*. Thus we evaluate our system's label accuracy by asking to the judge model whether the assigned label is correct. We compare SciClaims results with MultiVerS (Wadden et al., 2022).

SciClaims demonstrates superior claim verification accuracy, outperforming the fine-tuned MultiVerS (Wadden et al., 2022) model despite operating in a zero-shot setting. As shown in Table 5, SciClaims consistently produces more accurate labels across all claim generation methods, except (Pan et al., 2021). Notably, SciClaims returns significantly fewer *NEI* labels than MultiVerS, providing greater value to users by delivering more definitive *SUPPORT* or *REFUTE* labels. The unusually high accuracy of (Pan et al., 2021). likely stems from its excessive *NEI* labels, which simplify label selection. Furthermore, the SciClaims+CDP+CR system achieves the highest overall performance, correctly labeling 50% of generated claims, outperforming the next-best MultiVerS-based system by over five points.

Table 5 also reports the average processing time per document for each system configuration. Systems using MultiVerS as the verification module tend to be the fastest overall, with the exception of those paired with claim extraction modules like (Wright et al., 2022) and Noun Phrases Generation, which produce a high number of claims and thus increase runtime. Among all configurations, those using SciClaims+CDP+CR for claim generation—paired with either SciClaims or MultiVerS for verification—offer the best trade-off between predictive accuracy and time efficiency. Importantly, only the configuration that uses SciClaims as both the generation and verification module can run on a single 24GB GPU, making it uniquely suitable for real-world deployment.

5.3 Human evaluation

To complement our automated evaluation with LLMs, we conducted a human evaluation to assess the quality, relevance, and usability of SciClaims outputs. Five NLP-experts independently reviewed a sample of 154 PubMed documents processed by the system. The annotation tasks were divided into three phases, following the same structure as our LLM-based evaluation (see Section 5.2).

Claim quality evaluation. Annotators were shown an input paragraph and one randomly selected claim extracted by SciClaims. They answered the same eight binary questions used in the

System		All Claims		Correct Claims			
Extraction Module	Verification Module	Label Accuracy	Not NEI (%)	Label Accuracy	Not NEI (%)	System Score	Time/doc (secs.)
Sentence tokenizer	MultiVerS	0.5494	4.77	0.5127	4.82	0.1664	2.36
(Pan et al., 2021)	MultiVerS	**0.7591**	2.31	0.5714	11.90	0.0180	2.17
(Wright et al., 2022)	MultiVerS	0.5015	5.53	0.4268	8.35	0.0482	14.25
Noun Phrases Gen	MultiVerS	0.5965	3.08	0.4825	6.67	0.0117	15.01
SciClaims	MultiVerS	0.5944	6.02	0.5709	6.50	0.2704	2.86
SciClaims+CDP	MultiVerS	0.5867	5.71	0.5361	5.60	0.2971	1.82
SciClaims+CDP+CR	MultiVerS	0.6204	3.72	0.5922	7.44	0.4522	2.93
Sentence tokenizer	SciClaims	0.6041	61.20	0.6693	59.17	0.2173	11.72
(Pan et al., 2021)	SciClaims	0.6568	39.60	0.619	71.43	0.0195	7.61
(Wright et al., 2022)	SciClaims	0.6397	**69.44**	**0.7098**	**74.66**	0.0801	65.28
Noun Phrases Gen	SciClaims	0.6608	53.02	0.6762	70.48	0.0164	68.01
SciClaims	SciClaims	0.6361	58.49	0.6567	59.71	0.3111	12.38
SciClaims+CDP	SciClaims	0.6296	58.87	0.6364	62.47	0.3527	8.13
SciClaims+CDP+CR	SciClaims	0.6589	53.06	0.6574	54.23	**0.5020**	9.24

Table 5: Verification evaluation (phase 3) results with a judge model (Qwen2.5 72B AWQ). The table shows the label accuracy of each combination of extraction and verification module. The Correct Claims column counts only those considered correct in the Phase 1 questionnaire. Not NEI represents the proportion of claim–evidence pairs labeled as SUPPORT or REFUTE, rather than NEI (Not Enough Information). The last two columns summarize system-level performance: Processing Time per Document is the average time (in seconds) each system takes to process a single document, and System Score reflects the ratio of correctly labeled and well-formed claims across the system.

LLM-based evaluation (see Appendix B), assessing conciseness, precision, and other key dimensions. Table 6 shows that human judgments closely align with the judge model, with 70.5% of the claims meeting all eight criteria.

Claim Quality	Score	Rationale Quality	Score
Q1 (grounding)	0.9048		
Q2 (grammar)	0.9810		
Q3 (completeness)	0.9810	RQ1 (justification)	0.8571
Q4 (precision)	0.8667	RQ2 (relevance)	0.9206
Q5 (relevance)	0.8952	RQ3 (completeness)	0.7937
Q6 (conciseness)	0.8571		
Q7 (self-contained)	0.9238		
Q8 (contribution)	0.8857		
Correct Claims (%)	**70.47**	Correct Rationales (%)	**74.61**

Table 6: Claim and rationale evaluation of SciClaims by human annotators. Score indicates the overall ratio of 'yes' responses to the question along the annotators.

Document retrieval evaluation. Annotators were presented with a claim and its corresponding retrieved paragraph from SciClaims. They were asked whether the retrieved evidence provided sufficient information to verify the claim. In 60% of cases, annotators judged the paragraph as informative enough for claim verification, indicating moderate effectiveness of the retrieval module in real-world scenarios.

Claim verification evaluation. In this phase, annotators were shown a claim, its evidence, the system's predicted label, and the corresponding rationale. They rated the label's accuracy on a five-point scale, from 1 (completely inaccurate) to 5 (highly accurate). SciClaims achieved a strong average score of 4.40, reflecting high alignment between system predictions and human judgment. Additionally, annotators evaluate the quality of the generated rationale using three yes/no questions: (RQ1) Is the label justified by the rationale? (RQ2) Does the rationale focus on relevant information? (RQ3) Does the rationale provide enough context to understand the label?. As shown in Table 6, the responses from annotators indicate high satisfaction with the relevance and justification of rationales, though they also noted that completeness could be improved.

6 Conclusion

We presented SciClaims, a practical, end-to-end system for scientific claim analysis in the biomedical domain. Through an interactive web-based interface, users can extract, verify, and explore scientific claims with evidence-backed rationales, all powered by LLMs and a curated biomedical corpus. SciClaims is designed for usability and speed, and it is already being explored in real-world settings such as pharmaceutical patent analysis and systematic literature reviews. By combining explainable outputs with efficient infrastructure, it provides a robust tool for researchers, clinicians, and analysts who need to validate scientific information quickly and reliably. The system is openly accessible and ready for demonstration, offering an engaging experience to showcase the capabilities of modern LLM-based scientific reasoning.

Acknowledgments

The authors gratefully acknowledge the EU Large Language Models for EU (LLMs4EU) project (DIGITAL-20234-AI-06-LANGUAGE-01) and the HORIZON FAIR to Adapt to Climate Change (FAIR2Adapt) grant (agreement 101188256) for their support during the development of this work.

References

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. 2023. Factool: Factuality detection in generative ai – a tool augmented framework for multi-task and multi-domain scenarios. *Preprint*, arXiv:2307.13528.

Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. 2021. Climate-fever: A dataset for verification of real-world climate claims. *Preprint*, arXiv:2012.00614.

Islam Eldifrawi, Shengrui Wang, and Amine Trabelsi. 2024. Automated justification production for claim veracity in fact checking: A survey on architectures and approaches. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6679–6692, Bangkok, Thailand. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Fe-

ichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Nazanin Jafari and James Allan. 2024. Robust claim verification through fact detection. *Preprint*, arXiv:2407.18367.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Preprint*, arXiv:2004.12832.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Xiangci Li, Gully Burns, and Nanyun Peng. 2021. A paragraph-level multi-task learning model for scientific fact-verification. *Preprint*, arXiv:2012.14500.

Siting Liang and Daniel Sonntag. 2025. Explainable biomedical claim verification with large language models.

Hao Liu, Ali Soroush, Jordan G Nestor, Elizabeth Park, Betina Idnay, Yilu Fang, Jane Pan, Stan Liao, Marguerite Bernard, Yifan Peng, and Chunhua Weng. 2024. Retrieval augmented scientific claim verification. *JAMIA Open*, 7(1):ooae021.

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. *Preprint*, arXiv:1907.11692.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan, and William Yang Wang. 2021. Zero-shot fact verification by claim 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 2: Short Papers), pages 476–483, Online. Association for Computational Linguistics.

Ronak Pradeep, Xueguang Ma, Rodrigo Nogueira, and Jimmy Lin. 2021. Scientific claim verification with VerT5erini. In *Proceedings of the 12th International Workshop on Health Text Mining and Information Analysis*, pages 94–103, online. Association for Computational Linguistics.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

Alessandro Scirè, Andrei Stefan Bejgu, Simone Tedeschi, Karim Ghonim, Federico Martelli, and Roberto Navigli. 2024. Truth or mirage? towards end-to-end factuality evaluation with llm-oasis. *Preprint*, arXiv:2411.19655.

Marco Valenzuela, Vu Ha, and Oren Etzioni. 2015. Identifying meaningful citations. In *AAAI Workshops*.

Juraj Vladika, Ivana Hacajova, and Florian Matthes. 2025. Step-by-step fact verification system for medical claims with explainable reasoning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 805–816, Albuquerque, New Mexico. Association for Computational Linguistics.

Juraj Vladika and Florian Matthes. 2023. Scientific fact-checking: A survey of resources and approaches. *Preprint*, arXiv:2305.16859.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online. Association for Computational Linguistics.

David Wadden, Kyle Lo, Lucy Lu Wang, Arman Cohan, Iz Beltagy, and Hannaneh Hajishirzi. 2022. Multivers: Improving scientific claim verification with weak supervision and full-document context. *Preprint*, arXiv:2112.01640.

Dustin Wright, David Wadden, Kyle Lo, Bailey Kuehl, Arman Cohan, Isabelle Augenstein, and Lucy Lu Wang. 2022. Generating scientific claims for zero-shot scientific fact checking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2448–2460, Dublin, Ireland. Association for Computational Linguistics.

A Prompts

A.1 SciClaims claim extraction first step

<| begin_of_text |><| start_header_id |>system<| end_header_id|>
Your task is to generate a list with the main factual claims stated in a text. A factual claim makes an assertion about something regarding the subject matter that can be proved or contradicted with factual evidence. Factual claims must be expressed as meaningful, self-contained sentences. Do not include narrative context and disregard absolutely ALL self-referential parts.

Arrange your output using the format:
-- claim
-- claim
-- claim.

<| begin_of_text |><| start_header_id |>user<| end_header_id|>
TEXT: {text}

A.2 SciClaims+CDP

<| begin_of_text |><| start_header_id |>system<| end_header_id|>
Identify and list the main scientific claims stated in a passage. Each claim must satisfy the following criteria:
- **Convey an insight, interpretation, or conclusion drawn from the passage that is testable and generalizable**: The claim should assert an outcome, capability, or effect rather than merely describing a method, aim, or process.
  - Example (Good): "Neural networks outperform decision trees in image classification tasks." (Testable outcome)
  - Example (Bad): "The proposed method aims to use neural networks for better image classification." (Descriptive, not assertive)
- **Be expressed as a meaningful, self-contained statement**: Each claim should be fully understandable on its own, without needing context from the passage or other claims. It must convey a complete, independent idea. If referencing a study, survey, result, or process, phrase it as a general, verifiable claim.
  - Example (Good): "The Amazon rainforest is home to over 10 million species."
  - Example (Bad): "As mentioned earlier, the Amazon is one of the most biodiverse places in the world." (Requires prior context and doesn't stand alone.)
- **Emphasize generalization and scientific assertion**: Avoid descriptive or narrative conclusions.
  - Example (Good): "Exposure to blue light before sleep can reduce melatonin production." (Generalized, testable assertion)
  - Example (Bad): "The study investigates how exposure to blue light before sleep affects melatonin production." (Descriptive of method, not a general claim)
- **Be clear and concise**: Use straightforward language without unnecessary words.
  - Example (Good): "The Eiffel Tower is in Paris."
  - Example (Bad): "The Eiffel Tower, which is one of the most iconic landmarks in Europe and attracts millions of tourists every year, is located in Paris, France."
- **Exclude narrative context**: Focus on the factual assertion itself, not the surrounding story or background information.
  - Example (Good): "Water boils at 100C under normal atmospheric pressure."
  - Example (Bad): "In many cultures, people have believed for centuries that water boils at 100C, as scientists confirmed in the 18th century." (Includes unnecessary background information.)
- **Disregard all self-referential content**: Ignore any statements referring to the passage itself or the author intentions.
  - Example (Good): "The Earth orbits the Sun."
  - Example (Bad): "The study explains how the Earth orbits the Sun." (This is self-referential and refers to the passage itself.)
- **Be precise and objective**: Avoid ambiguity, subjective interpretation, or vague statements. Present claims as clear, verifiable facts.
  - Example (Good): "The Great Wall of China stretches approximately 13,000 miles."
  - Example (Bad): "The Great Wall of China is pretty long." (Vague and subjective.)
- **Be relevant to the broader debate or public discourse**: Focus on verification-worthy claims that introduce new information rather than merely restating common knowledge.
  - Example (Good): "The global temperature has increased by about 1C since the late 19th century."
  - Example (Bad): "The Earth is a planet." (Common knowledge and not contributing new, verifiable information.)

Present the output using the following format:
-- claim
-- claim
-- claim

<| begin_of_text |><| start_header_id |>user<| end_header_id|>
PASSAGE: {text}

A.3 SciClaims claim extraction second step (refinement)

Now, using the information given by the passage, reformulate each individual claim to be fully understandable by itself, even without having the context from the passage or the rest of the claims from the list. Change the terminology or add context information to each claim if necessary.

A.4 SciClaims+CDP+CR (refinement)

<| begin_of_text |><| start_header_id |>system<| end_header_id|>
Given a claim and the passage it was extracted from, reformulate the claim to fully adhere to **ALL** the following criteria.
- **Convey an insight, interpretation, or conclusion drawn from the passage that is testable and generalizable**: The claim should assert an outcome, capability, or effect rather than merely describing a method, aim, or process.

- Example (Good): "Neural networks outperform decision trees in image classification tasks." (Testable outcome)
- Example (Bad): "The proposed method aims to use neural networks for better image classification." (Descriptive, not assertive)
- **Be expressed as a meaningful, self-contained statement**: Each claim should be fully understandable on its own, without needing context from the passage or other claims. It must convey a complete, independent idea. If referencing a study, survey, result, or process, phrase it as a general, verifiable claim.
  - Example (Good): "The Amazon rainforest is home to over 10 million species."
  - Example (Bad): "As mentioned earlier, the Amazon is one of the most biodiverse places in the world." (Requires prior context and doesn't stand alone.)
- **Emphasize generalization and scientific assertion**: Avoid descriptive or narrative conclusions.
  - Example (Good): "Exposure to blue light before sleep can reduce melatonin production." (Generalized, testable assertion)
  - Example (Bad): "The study investigates how exposure to blue light before sleep affects melatonin production." (Descriptive of method, not a general claim)
- **Be clear and concise**: Use straightforward language without unnecessary words.
  - Example (Good): "The Eiffel Tower is in Paris."
  - Example (Bad): "The Eiffel Tower, which is one of the most iconic landmarks in Europe and attracts millions of tourists every year, is located in Paris, France."
- **Exclude narrative context**: Focus on the factual assertion itself, not the surrounding story or background information.
  - Example (Good): "Water boils at 100C under normal atmospheric pressure."
  - Example (Bad): "In many cultures, people have believed for centuries that water boils at 100C, as scientists confirmed in the 18th century." (Includes unnecessary background information.)
- **Disregard all self-referential content**: Ignore any statements referring to the passage itself or the author intentions.
  - Example (Good): "The Earth orbits the Sun."
  - Example (Bad): "The study explains how the Earth orbits the Sun." (This is self-referential and refers to the passage itself.)
- **Be precise and objective**: Avoid ambiguity, subjective interpretation, or vague statements. Present claims as clear, verifiable facts.
  - Example (Good): "The Great Wall of China stretches approximately 13,000 miles."
  - Example (Bad): "The Great Wall of China is pretty long." (Vague and subjective.)
- **Be relevant to the broader debate or public discourse**: Focus on verification-worthy claims that introduce new information rather than merely restating common knowledge.
  - Example (Good): "The global temperature has increased by about 1C since the late 19th century."
  - Example (Bad): "The Earth is a planet." (Common knowledge and not contributing new, verifiable information.)

Present the output using the following format:
{"original_claim": <str>, "refined_claim": <str>, "rationale": <str>}

<|begin_of_text|><|start_header_id|>user<|end_header_id|>
CLAIM: {claim}
PASSAGE: {text}

A.5 SciClaims claim verification

<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a claim analyst. Upon receiving a claim and an evidence, your task is to figure out if the claim is either supported, contradicted or unrelated based exclusively on the evidence.

- If you are confident that the claim is supported by the evidence your answer will be "SUPPORT".
- If you are certain that the evidence directly contradicts the claim, your answer will be "CONTRADICT". Please note that if the claim is just not mentioned in the evidence, or if it is unrelated to the evidence, it does not mean it is contradicted. For that cases, the answer will be "NEI".
- If the evidence does not contain enough information or if it is not related to the claim, your answer will be "NEI", which stands for Not Enough Information.

Arrange the output as a JSON dictionary with the keys "response" and "evidence" and ensure your output is JSON-valid.
- The "response" values can only be "SUPPORT", "CONTRADICT" or "NEI".
- The "evidence" value must be a list of sentences from the evidence which are more related to your decision. If the decision is "NEI", this field will be empty.

<|begin_of_text|><|start_header_id|>user<|end_header_id|>
CLAIM: {claim}
EVIDENCE: {evidence}

A.6 Phase 1 evaluation with judge model (Q1)

<|begin_of_text|><|start_header_id|>system<|end_header_id|>
Given a sentence and a paragraph, answer the following question. Use exclusively the content of the paragraph to answer the question.

Is the sentence supported by the paragraph?

Return your response as a json dictionary, following this structure: {"answer":<Yes/No>, "rationale": <str>}

<|begin_of_text|><|start_header_id|>user<|end_header_id|>
SENTENCE: {claim}
PARAGRAPH: {text}

A.7 Phase 1 evaluation with judge model (Q2-8)

Given a claim, answer the following question.

{QUESTION}

Return your response as a json dictionary, following this structure: {"answer":<Yes/No>, "rationale": <str>}

<|begin_of_text|><|start_header_id|>user<|end_header_id|>
CLAIM: {claim}

A.8 Phase 2 evaluation with judge model

<|begin_of_text|><|start_header_id|>system<|end_header_id|>
Given a claim and a paragraph, answer the following question.

Is the information contained in the paragraph useful to verify the claim?

Return your response as a json dictionary, following this structure: {"answer":<Yes/No>, "rationale": <str>}

<|begin_of_text|><|start_header_id|>user<|end_header_id|>
CLAIM: {claim}
PARAGRAPH: {text}

A.9 Phase 3 evaluation with judge model

<|begin_of_text|><|start_header_id|>system<|end_header_id|>
Given a claim and a paragraph, answer the following question.

Is the claim {SUPPORTED/REFUTED} by the paragraph?

Return your response as a json dictionary, following this structure: {"answer":<Yes/No>, "rationale": <str>}

<|begin_of_text|><|start_header_id|>user<|end_header_id|>
CLAIM: {claim}
PARAGRAPH: {text}

B Claim Quality Questionnaire

Id	Question
Q1	Is the claim grounded by the original text?
Q2	Is the claim grammatically correct?
Q3	Does the claim have all the necessary components (subject, predicate, and relevant qualifiers) to form a complete thought?
Q4	Is the claim precise and specific rather than vague?
Q5	Does the claim introduce new information rather than just restating common knowledge?
Q6	Is the claim concise without losing essential information?
Q7	Does the claim provide enough information to be understood independently?
Q8	Would verifying the claim add value to public knowledge?

Table 7: Questions asked to the LLM judge and human annotators to evaluate the quality of the generated claims.

AgentCPM-GUI: Building Mobile-Use Agents with Reinforcement Fine-Tuning

Zhong Zhang[1*], **Yaxi Lu**[1*], **Yikun Fu**[1†], **Yupeng Huo**[2], **Shenzhi Yang**[2], **Yesai Wu**[1], **Han Si**[1†]
Xin Cong[1], **Haotian Chen**[1], **Yankai Lin**[2‡], **Jie Xie**[1], **Wei Zhou**[1], **Wang Xu**[1], **Yuanheng Zhang**[1†]
Zhou Su[3], **Zhongwu Zhai**[3], **Xiaoming Liu**[3], **Yudong Mei**[3], **Jianming Xu**[3], **Hongyan Tian**[3]
Chongyi Wang[3], **Chi Chen**[1], **Yuan Yao**[1,4], **Zhiyuan Liu**[1‡], **Maosong Sun**[1‡]
[1]Tsinghua University [2]Renmin University of China [3]ModelBest Inc. [4]Shanghai Qi Zhi Institute
zhongzhang@tsinghua.edu.cn lyx23@mails.tsinghua.edu.cn

Abstract

Large language model agents have enabled GUI-based automation, particularly for mobile devices. However, deployment remains limited by noisy data, poor generalization, and lack of support for non-English GUIs. In this work, we present AgentCPM-GUI, an 8B-parameter GUI agent built for robust and efficient on-device GUI interaction. Our training pipeline includes grounding-aware pre-training to enhance perception, supervised fine-tuning on high-quality Chinese and English trajectories to imitate human-like actions, and reinforcement fine-tuning with GRPO to improve reasoning capability. AgentCPM-GUI achieves promising performance on five public benchmarks and our proposed Chinese benchmark CAGUI. To facilitate reproducibility and further research, we publicly release all code, model checkpoint, and evaluation data at: https://github.com/OpenBMB/AgentCPM-GUI

1 Introduction

The rapid advancements in Large Language Models (LLMs) and Multimodal Large Models (MLLMs) have catalyzed a new era of autonomous AI agents (Zhao et al., 2023; Wang et al., 2024b). These agents are increasingly capable of understanding complex instructions (Ouyang et al., 2022; Qian et al., 2024), performing multi-step planning (Huang et al., 2024), and interacting with external tools or environments (Qin et al., 2024, 2025a). A critical frontier for deploying these intelligent agents in practical, human-centric applications is enabling them to proficiently operate Graphical User Interfaces (GUIs) (Wang et al., 2024c; Nguyen et al., 2025; Zhang et al., 2025a), particularly within the ubiquitous Android ecosystem, where they serve as the primary interaction layer for a vast array of daily digital tasks. Empowering LLM agents to seamlessly navigate and manipulate these mobile GUIs is essential for transforming them into truly versatile digital assistants capable of automating a wide spectrum of tasks on smartphones, thereby enhancing user productivity and accessibility.

Early GUI agents emerged when Vision-Language Models (VLMs) had limited ability in reliably control GUI widgets. To compensate, researchers augmented model inputs with structured metadata, such as Android view hierarchies and system APIs, and even off-loaded perception and planning to more capable external VLMs (e.g., GPT-4o (Hurst et al., 2024)), thereby improving widget grounding and action execution (Zhang et al., 2025b; Chen et al., 2025a; Chen and Li, 2024; Zheng et al., 2024; Kim et al., 2023; Wang et al., 2024a). Although effective, these hybrid pipelines propagated errors from cross-modal mismatches, incurred round-trip latency, and depended on metadata that many apps do not expose, creating significant challenges for generality and scalability. Recent GUI agents have advanced to resolving interface elements directly from raw pixels, enabling a single end-to-end model to match or even surpass earlier hybrid approaches (Hong et al., 2024; Cheng et al., 2024; Qin et al., 2025b; Xu et al., 2024; Wu et al., 2025; Lin et al., 2025; Zhang and Zhang, 2024). This shift positions purely visual, end-to-end modeling as the most scalable paradigm.

Despite significant progress, current visual GUI agents still face several challenges: **(1) Data quality and scale**. High-quality, fine-grained interaction trajectories that capture realistic user behavior in diverse mobile apps are notoriously difficult to collect at scale. Most publicly available datasets either rely on synthetic generation or emulator-based recordings, both of which can introduce noise and lack semantic diversity. Such imperfect supervision limits the agent's ability to learn precise wid-

[*] Equal contribution.
[†] Internship at Tsinghua University.
[‡] Corresponding authors.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 155–180
November 4-9, 2025

get grounding, compositional reasoning, and long-horizon action planning. **(2) Reasoning generalization**. GUI agents that are trained solely via imitation learning tend to overfit to interface patterns, resulting in brittle planning and poor generalization when task instructions deviate from seen templates or when UI layouts exhibit minor variations. **(3) Language and regional coverage**. Current research concentrates almost exclusively on English GUIs, paying limited attention to the rapidly growing and diverse Chinese mobile ecosystem, whose interface design conventions and linguistic cues differ substantially. These differences limit the generalizability of current agents in multilingual and culturally diverse settings.

To address these challenges, we propose AgentCPM-GUI, a VLM-based agent for mobile GUI understanding and interaction. The key features of this work are as follows.

- **High-quality training data.** We curate a large-scale corpus of 55K trajectories with 470K steps, encompassing a wide variety of Chinese Android apps via targeted collection and meticulous annotation. To enhance generalization and mitigate overfitting, we further incorporate and rigorously de-duplicate multiple public English Android datasets. The resulting unified dataset supports effective training, enabling robust cross-lingual and cross-app behavior modeling.

- **Progressive training for perception, imitation, and reasoning.** We adopt a three-stage progressive training pipeline to equip the agent with strong GUI understanding and reasoning capabilities, consisting of grounding-aware pre-training to enhance visual perception; supervised fine-tuning (SFT) to establish a reliable behavioral prior; and reinforcement fine-tuning (RFT) (OpenAI, 2024; Shao et al., 2024; Trung et al., 2024) to further strengthen reasoning ability, enabling robust performance on long-horizon and compositional tasks. In addition, we optimize the training framework with asynchronous rollout and load balancing to support scalable reinforcement learning.

- **Edge device oriented design.** To reduce decoding overhead, we carefully select action tokens to avoid unnecessary token fragmentation and adopt a compact JSON-based action format, resulting in an average output length of just 9.7 tokens per action. While prior works largely overlook redundancy in action space design, our concise representation significantly improves runtime efficiency, enabling smooth and responsive on-device execution.

- **Comprehensive benchmarking.** We evaluate AgentCPM-GUI on the widely used English GUI agent benchmarks: AndroidControl (Li et al., 2024), GUI-Odyssey (Lu et al., 2024a), and AITZ (Zhang et al., 2024). In addition, we introduce **CAGUI**, the first large-scale Chinese Android GUI benchmark. CAGUI is a representative subset of our corpus designed for public evaluation. AgentCPM-GUI achieves new state-of-the-art performance across all datasets, demonstrating robust multilingual and cross-app generalization.

2 Method

2.1 Architecture Overview

As shown in Figure 1, we adopt a three-stage training framework to transform MiniCPM-V (Yao et al., 2024), a lightweight 8B vision-language model, into a GUI-capable agent. The first stage focuses on visual perception and grounding, using tasks like OCR and widget localization to enhance the model's ability to align GUI elements with language. In the second stage, the model is fine-tuned on supervised GUI trajectories paired with natural language instructions, enabling it to imitate human-like actions. Finally, reinforcement fine-tuning is applied using Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to further improve planning and decision-making.

2.2 Action Space Design

We design a unified and compositional action space that is compact and friendly for language model generation. It consists of six atomic actions, enabling expressive yet efficient GUI control:

- `POINT`: Specifies a normalized coordinate `(x,y)` in `[0,1000]` to perform a tap. Combined with `to` or `duration`, it supports swipes and long presses.

- `to`: Indicates swipe direction or complements `POINT` to define gesture endpoints.

- `TYPE`: Inputs a specified text string into the

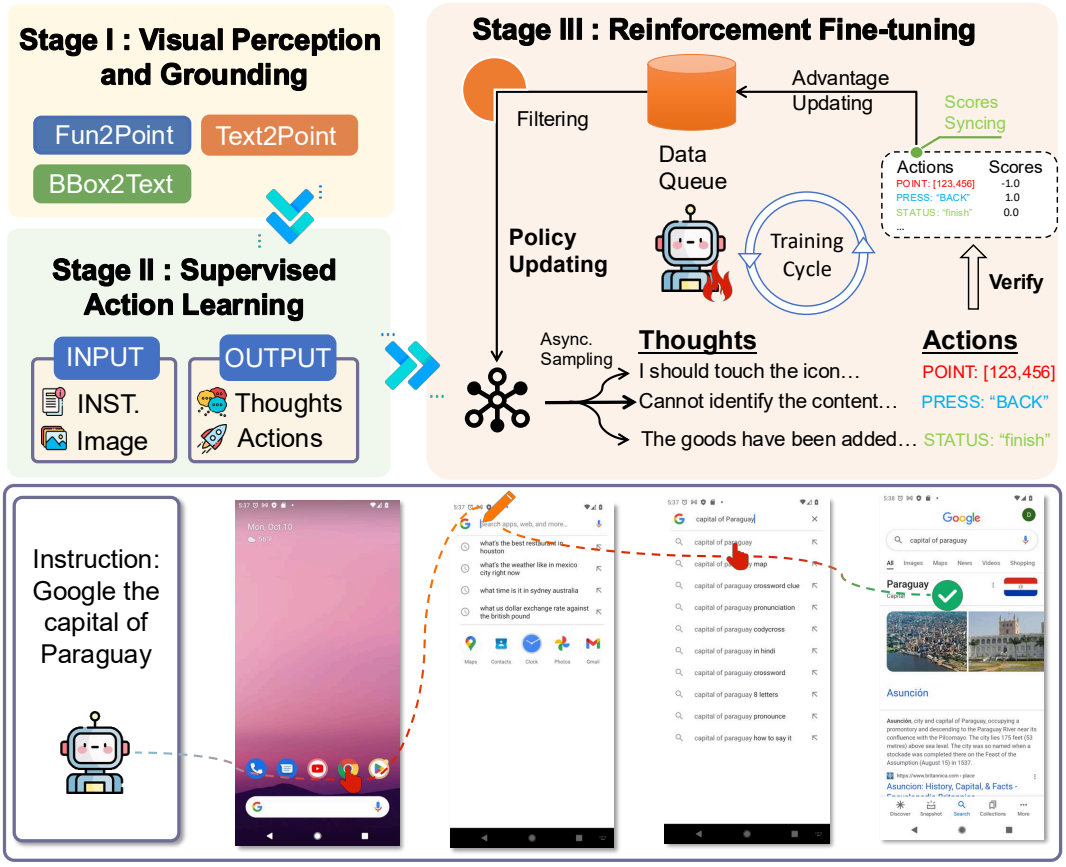


Figure 1: Overview of our training framework.

currently focused input field.

- `PRESS`: Simulates device keys like `"HOME"`, `"BACK"`, or `"ENTER"` for common operations.
- `STATUS`: Communicates task state (e.g., `"continue"`, `"finish"`, `"impossible"`), allowing dynamic control flow.
- `duration`: Time an action lasts. Used alone for delays or with `POINT` for long actions.

To reduce token overhead, we adopt a compact JSON format with no extra whitespace, resulting in a low average token cost of 9.7 per action, enabling fast and efficient execution on edge devices.

2.3 Stage I: Visual Perception and Grounding

For grounding pre-training, we collect Android GUI data by sampling examples from several open-source corpora (AITZ (Zhang et al., 2024), GUICourse (Chen et al., 2025b), OS-Atlas (Wu et al., 2025), UGround (Gou et al., 2025), ScreenSpot (Cheng et al., 2024)) and additional screenshots from our collected Chinese app data. Each image is formulated as either an OCR task that asks the model to write the text in a marked region, or a widget-localization task that asks it to output the bounding box coordinate of a referenced UI element. Grounding batches mix in 50% general multimodal SFT data (e.g., Chat, VQA, Multimodal Reasoning) (Yao et al., 2024), which regularizes the vision module while letting it absorb GUI-specific cues. In total, the grounding pre-training dataset comprises 12M samples.

This pre-training stage plays a crucial role in establishing the model's low-level perceptual and grounding abilities. We observe that, after this stage, the model demonstrates strong proficiency in identifying and locating GUI widgets, especially in accurately predicting coordinates based on visual cues. However, the model at this point still struggles to generate well-formed function calls or to reason over action types, indicating limited understanding of higher-level task semantics and planning. These capabilities are further enhanced in the subsequent SFT and RFT stages.

2.4 Stage II: Supervised Imitation Learning

Due to the scarcity of high-quality open-source datasets for Chinese Android apps, we constructed a large-scale, high-fidelity dataset of GUI inter-

action trajectories to support supervised imitation learning. The corpus covers over 30 mainstream Chinese apps, spanning eight functional domains: life services, e-commerce, navigation, social, video, music/audio, reading/learning, and productivity. This ensures that the agent is exposed to a wide spectrum of UI layouts, widget types, and task intents. In total, we obtained 55K complete task trajectories comprising 470K atomic steps, approximately 8.5 steps per trajectory.

In order to enhance cross-lingual generalization and reduce over-fitting, we augmented our Chinese corpus with publicly available English-language datasets: AITW (Li et al., 2024), AITZ (Zhang et al., 2024), AMEX (Chai et al., 2025), AndroidControl (Li et al., 2024), and GUI-Odyssey (Lu et al., 2024a). Since AITW is internally redundant, we performed intra-query de-duplication. For each trajectory, we extracted ResNet-50 features from its screenshots and averaged them to produce a trajectory embedding. We then grouped trajectories by shared query and, within each group, removed those whose cosine similarity to any previously retained sample exceeded a fixed threshold. This retained approximately 40% of the original data.

Empirically, training solely on GUI-interaction data led to a pronounced mode collapse during the subsequent RFT stage, manifesting as impoverished and repetitive reasoning thoughts. To mitigate this, we mixed 50% general multimodal SFT data into training batches, which helped stabilize policy optimization. The SFT data comprises a mix of single-turn (system-user-assistant) and multi-turn dialogues. For multi-turn examples, we retained only the last three turns of user-assistant interaction to provide sufficient conversational context while keeping input sequences within tractable length limits. In total, 6.9M instances were used for the SFT stage.

2.5 Stage III: Reinforcement Fine-tuning

We introduce an RFT stage to improve the agent's reasoning ability. To make RFT practical at scale, we further develop a training framework which supports asynchronous rollout and two levels of load balancing to improve efficiency and scalability across distributed environments.

2.5.1 Algorithmic Design

We conduct RFT based on the GRPO (Shao et al., 2024) algorithm. GRPO replaces the value critic of PPO (Schulman et al., 2017) with a group-wise comparison of candidate completions. For reward design and validation, we apply a two-stage validation scheme to evaluate model outputs: (1) format checking and (2) semantic correctness. The reward is mapped to the range $[-1, 1]$. If an output fails the format check (e.g., malformed structure or missing fields), a reward of -1 is assigned. If the format is correct but the answer is semantically incorrect, the reward is 0. If both format and answer are correct, the reward is 1. For action spaces involving continuous goals, such as predicting a `POINT` target, we further define correctness by spatial accuracy: if the predicted point falls within the ground-truth bounding box, a reward of 1 is assigned; otherwise, 0. This fine-grained reward design encourages both syntactic correctness and task-specific accuracy.

2.5.2 System Optimization

Our training system adopts an asynchronous architecture that decouples rollout execution from policy updates. Once a task ID is dispatched from the global task queue, it is sampled n times according to the GRPO algorithm to generate multiple candidate responses per policy. After inference and reward computation for each sample are complete, the main process computes the advantage for the samples using GRPO's variance-reduced estimator. These advantage values are then sent to the node-level main process for policy updating. The global main process collects all necessary statistics and, when synchronization conditions are met, coordinates a unified policy update across nodes. This design ensures tight integration of GRPO's optimization logic within our distributed, asynchronous training framework.

Asynchronous Rollout. In our design, each GPU group performs inference independently and asynchronously. The inference results are first synchronized to the local node's main process. Then, each local main process communicates its inference status with a global main process, which tracks global rollout progress and coordinates training updates. During inference, each GPU group also asynchronously requests the next batch of data required for computing policy gradients. The global main process monitors the overall rollout status and, once a pre-defined synchronization condition is met, broadcasts a signal to all GPU groups to pause rollout and perform a synchronized model update. This asynchronous rollout scheme ensures that GPU groups operate efficiently without wait-

Table 1: GUI grounding accuracy on the CAGUI benchmark over the Fun2Point, Text2Point, and Bbox2Text sub-tasks. **Bold** and underline indicate the best and second-best results.

Models	Fun2Point	Text2Point	Bbox2Text	Average
Closed-source Models				
GPT-4o (Hurst et al., 2024)	22.1	19.9	14.3	18.8
GPT-4o with grounding (Lu et al., 2024b)	44.3	44.0	14.3	34.2
Open-source Models				
Qwen2.5-VL-7B (Bai et al., 2023)	59.8	59.3	50.0	56.4
InternVL2.5-8B (Dong et al., 2024)	17.2	24.2	45.9	29.1
InternVL2.5-26B (Dong et al., 2024)	14.8	16.6	36.3	22.6
OS-Genesis-7B (Sun et al., 2025)	8.3	5.8	4.0	6.0
UI-TARS-7B (Qin et al., 2025b)	56.8	66.7	1.4	41.6
OS-Altas-7B (Wu et al., 2025)	53.6	60.7	0.4	38.2
Aguvis-7B (Xu et al., 2024)	60.8	**76.5**	0.2	45.8
AgentCPM-GUI	**79.1**	**76.5**	**58.2**	**71.3**

ing for each other, thus fully utilizing resources.

Hierarchical Load Balancing. The asynchronous design introduces challenges related to load imbalance, particularly at two levels: intra-node (between GPU groups) and inter-node (between different compute nodes). Intra-node imbalance is addressed by constructing a global task queue from which inference tasks are dynamically dispatched to GPU groups. This design make each GPU group consistently have access to available tasks, thereby minimizing idle time. However, nodes with differing hardware configurations or system loads can result in inter-node imbalance: some nodes may accumulate more rollout results than others. To address this, we implement a work stealing mechanism: underutilized nodes can request inference results from overburdened peers. This approach is particularly suited for large-scale, multi-modal inference outputs, which are often expensive to transmit and manage. Work stealing provides a flexible and scalable solution that avoids the drawbacks of forced synchronization across machines.

3 Experiments

3.1 GUI Grounding Capability

We evaluate GUI grounding on CAGUI through three tasks designed to assess different aspects of visual-language alignment and understanding: **1) Fun2Point**. Given a description of a component's function in the GUI (e.g., "this button opens the website"), the model must locate the correct coordinates of the mentioned component; **2) Text2Point**. The model is required to locate a given textual string appearing within the GUI; **3) Bbox2Text**. The model receives a bounding box location on the GUI and must accurately output the corresponding textual content. Representative examples of these tasks are included in Appendix C.1.

All three grounding tasks are evaluated on the CAGUI benchmark, which was specifically curated for assessing GUI grounding capability in Chinese Android apps. The raw dataset consists of screenshots paired with corresponding XML metadata collected from real-world apps. Each XML file provides fine-grained annotations for GUI widgets, including bounding box coordinates, textual content, and component types. For the Text2Point and Bbox2Text tasks, annotations were directly extracted from the XML metadata by aligning textual content with their corresponding bounding boxes. For Fun2Point, additional function-level labels were constructed to reflect the semantic roles of GUI widgets. To generate these labels, we first overlaid bounding boxes onto the screenshots to explicitly highlight the spatial boundaries of each widget. Then, we prompted a strong VLM Qwen2.5-VL-72B to produce concise functional descriptions, yielding high-quality semantic labels for widgets.

Evaluation procedures were tailored to the input-output formats of each model. InternVL models output bounding boxes, which are evaluated against the ground-truth using the Intersection-over-Union

Table 2: Step-level action prediction performance on five GUI Agent benchmarks, in terms of Type Match (TM) and Exact Match (EM). **Bold** and underline indicate the best and second-best results. *OS-Atlas uses different train/test splits on GUI-Odyssey benchmark and is not directly comparable.

Models	AC-Low		AC-High		Odyssey		AITZ		CAGUI	
	TM	EM	TM	EM	TM	EM	TM	EM	TM	EM
Closed-source Models										
GPT-4o (Hurst et al., 2024)	-	19.5	-	20.8	-	20.4	70.0	35.3	3.67	3.67
Gemini 2.0 (Deepmind, 2024)	-	28.5	-	60.2	-	3.27	-	-	-	-
Claude (Anthropic, 2024)	-	19.4	-	12.5	60.9	-	-	-	-	-
Open-source Models										
Qwen2.5-VL-7B (Bai et al., 2023)	94.1	85.0	75.1	62.9	59.5	46.3	78.4	54.6	74.2	55.2
UI-TARS-7B (Qin et al., 2025b)	**95.2**	**91.8**	**81.6**	**74.4**	86.1	67.9	80.4	65.8	88.6	70.3
OS-Genesis-7B (Sun et al., 2025)	90.7	74.2	65.9	44.4	11.7	3.63	20.0	8.45	38.1	14.5
OS-Atlas-7B (Wu et al., 2025)	73.0	67.3	70.4	56.5	91.8*	76.8*	74.1	58.5	81.5	55.9
Aguvis-7B (Xu et al., 2024)	93.9	89.4	65.6	54.2	26.7	13.5	35.7	19.0	67.4	38.2
OdysseyAgent (Lu et al., 2024a)	65.1	39.2	58.8	32.7	90.8	73.7	59.2	31.6	67.6	25.4
AgentCPM-GUI	94.4	90.2	77.7	69.2	**90.9**	**75.0**	**85.7**	**76.4**	**96.9**	**91.3**

(IoU) metric, with a threshold of 0.5 indicating a successful match. GPT-4o is augmented with OmniParser (Lu et al., 2024b), which extracts layout structures and text/icon segments before the model predicts a target box index. Models including ours generate point coordinates and are assessed by comparing them with ground-truth locations under a predefined spatial tolerance.

The results are summarized in Table 1. AgentCPM-GUI significantly outperforms all baselines across all three tasks. In particular, it achieves a large performance margin in the Bbox2Text task, where most baseline models struggle-largely due to the need for precise alignment between visual regions and text content. Despite the task's difficulty, AgentCPM-GUI attains a 58.2% accuracy, while nearly all competing models score below 5%. This highlights our model's superior grounding ability, especially in mobile interface contexts where visual complexity, small text, and overlapping elements pose unique challenges.

3.2 Action Prediction Capability

We conduct a comprehensive evaluation of AgentCPM-GUI on representative benchmarks: AndroidControl (Li et al., 2024), GUI-Odyssey (Lu et al., 2024a), AITZ (Zhang et al., 2024), and CAGUI, covering diverse GUI interaction patterns across both English and Chinese environments. Each benchmark adopts two standard evaluation metrics: Type Match (TM), which checks if the predicted action type matches the ground truth, and Exact Match (EM), which additionally requires all parameters to be correctly predicted. As shown in Table 2, AgentCPM-GUI achieves state-of-the-art performance across all benchmarks. Notably, it demonstrates strong generalization in complex multi-step scenarios, such as those in GUI-Odyssey and AITZ, significantly outperforming existing models. On the CAGUI benchmark, our model achieves 96.9% TM and 91.3% EM, substantially ahead of other models, highlighting its effectiveness in Chinese-language GUI settings.

All baseline results are from our own re-implementations to ensure fair and reproducible comparisons. We closely followed each model's official instructions and prompts where available, and applied consistent input and evaluation protocols throughout. Notably, OS-Atlas uses a different train/test split on GUI-Odyssey benchmark, so its results are not directly comparable. Our evaluation code and benchmarks are publicly released to support reproducibility and future research.

3.3 Effects of Reinforcement Fine-tuning

To assess the contribution of RFT, we compare our model's performance before and after RFT across all benchmarks, as shown in Table 3. On challenging datasets such as AndroidControl-Low, GUI-Odyssey, and AITZ, RFT brought significant

Table 3: Ablation study comparing AgentCPM-GUI before and after RFT.

Models	AC-Low		AC-High		Odyssey		AITZ		CAGUI	
	TM	EM	TM	EM	TM	EM	TM	EM	TM	EM
AgentCPM-GUI-SFT	87.6	83.1	**78.6**	**69.5**	86.1	66.7	79.0	61.1	**96.9**	**91.5**
AgentCPM-GUI-RFT	**94.4**	**90.2**	77.7	69.2	**90.9**	**75.0**	**85.7**	**76.4**	**96.9**	91.3

improvements, especially in exact match accuracy. This demonstrates its effectiveness in enhancing the model's ability to handle long-horizon reasoning and complex decision-making. However, on datasets like AndroidControl-High and CAGUI, the SFT-only model already performed competitively or even slightly better. This can be attributed to the benchmarks' large and diverse training sets, which expose the model to similar patterns during SFT. As a result, imitation learning alone suffices for effective generalization, with additional reinforcement offering minimal incremental benefits.

4 Related Work

Recent advances in GUI agents have been supported by the development of various datasets and benchmarks, covering both grounding tasks and interaction modeling (Deng et al., 2023; Cheng et al., 2024; Wu et al., 2025; Chen et al., 2025b; Gou et al., 2025; Rawles et al., 2023; Zhang et al., 2024; Li et al., 2024; Lu et al., 2024a; Chai et al., 2025; Rawles et al., 2025). However, most of these focus on English GUIs, limiting cross-lingual generalization. Concurrently, the field has witnessed a transition from modular to end-to-end vision-language agents, with large VLMs trained on millions of screenshots increasingly used for grounding and planning (Wang et al., 2024a; Zheng et al., 2024; Hong et al., 2024; Xu et al., 2024; Qin et al., 2025b; Lin et al., 2025; Yang et al., 2025; Sun et al., 2025). To improve reasoning and adaptability, reinforcement learning techniques have been incorporated, ranging from offline policy training to reward-based fine-tuning and reasoning-centric paradigms (Bai et al., 2024; Wang et al., 2025; Bai et al., 2025; Zhai et al., 2024; Liu et al., 2025b; Tan et al., 2025; Huang et al., 2025; Zhou et al., 2025; Lu et al., 2025; Xia and Luo, 2025; Liu et al., 2025a; Papoudakis et al., 2025).

5 Conclusion

We present AgentCPM-GUI, a VLM-based agent for mobile GUI interaction, trained via a three-stage pipeline that builds grounding, action, and reasoning skills. To support this, we construct a high-quality Chinese Android dataset and incorporate selected English data for cross-lingual generalization. Reinforcement fine-tuning further enhances planning for long-horizon tasks. Experiments on public and CAGUI benchmarks show strong performance, particularly in Chinese settings. All code, data, and models will be released to support future research.

Limitations

While AgentCPM-GUI demonstrates strong performance across both English and Chinese GUI tasks, several limitations remain. First, the model's ability to handle long-horizon interactions is still constrained by limited historical context. Although reinforcement fine-tuning enhances planning and reasoning, the agent only conditions on short, recent trajectories, which can hinder its ability to manage complex, multi-turn tasks requiring memory of earlier states or user preferences. Second, error recovery remains a challenge. The current agent lacks a robust mechanism for detecting failures and autonomously retrying or rolling back actions. While reinforcement training improves overall task success, it does not explicitly teach the model to recover from suboptimal decisions or ambiguous states. Third, our action space, though efficient, assumes deterministic execution and does not yet account for real-time feedback or unexpected UI changes during interaction, which may reduce robustness in deployment. Future work may incorporate memory modules, error-aware execution loops, or uncertainty modeling to further strengthen the agent's autonomy and adaptability in dynamic mobile environments.

Acknowledgments

This work was supported by the Postdoctoral Fellowship Program of CPSF (Grant No. GZC20240831) and the China Postdoctoral Science Foundation (Grant No. 2025M771586).

References

Anthropic. 2024. Introducing computer use, a new claude 3.5 sonnet, and claude 3.5 haiku. https://www.anthropic.com/news/3-5-models-and-computer-use.

Hao Bai, Yifei Zhou, Li Erran Li, Sergey Levine, and Aviral Kumar. 2025. Digi-Q: Learning VLM q-value functions for training device-control agents. In *International Conference on Learning Representations*.

Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. In *Advances in Neural Information Processing Systems 38*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint*.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. 2025. AMEX: android multi-annotation expo dataset for mobile GUI agents. In *Findings of the Association for Computational Linguistics*, pages 2138–2156.

Wei Chen and Zhiyuan Li. 2024. Octopus v2: On-device language model for super agent. *arXiv preprint*.

Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2025a. Octopus: On-device language model for function calling of software apis. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 329–339.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2025b. GUICourse: From general vision language models to versatile GUI agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 21936–21959.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 9313–9332.

Google Deepmind. 2024. Introducing gemini 2.0: our new ai model for the agentic era. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samual Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems 36*.

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Songyang Zhang, Haodong Duan, Wenwei Zhang, Yining Li, Hang Yan, Yang Gao, Zhe Chen, Xinyue Zhang, Wei Li, Jingwen Li, Wenhai Wang, Kai Chen, Conghui He, and 5 others. 2024. InternLM-XComposer2-4KHD: A pioneering large vision-language model handling resolutions from 336 pixels to 4k HD. In *Advances in Neural Information Processing Systems 38*.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2024. CogAgent: A visual language model for GUI agents. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024*, pages 14281–14290.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. 2025. Vision-R1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of LLM agents: A survey. *arXiv preprint*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. GPT-4o system card. *arXiv preprint*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. In *Advances in Neural Information Processing Systems 36*.

Wei Li, William W. Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on computer control agents. *arXiv preprint*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2025. ShowUI: One vision-language-action model for GUI visual agent. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19498–19508.

Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. 2025a. InfiGUI-R1: Advancing multimodal GUI agents from reactive actors to deliberative reasoners. *arXiv preprint*.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. 2025b. Visual-RFT: Visual reinforcement fine-tuning. *arXiv preprint*.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. 2024a. GUIOdyssey: A comprehensive dataset for cross-app GUI navigation on mobile devices. *arXiv preprint*.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024b. Omniparser for pure vision based GUI agent. *arXiv preprint*.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025. UIR1: Enhancing action prediction of gui agents by reinforcement learning. *arXiv preprint*.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, Xintong Li, Jing Shi, Hongjie Chen, Viet Dac Lai, Zhouhang Xie, Sungchul Kim, Ruiyi Zhang, Tong Yu, Md. Mehrab Tanjim, and 11 others. 2025. GUI agents: A survey. In *Findings of the Association for Computational Linguistics*, pages 22522–22538.

OpenAI. 2024. Reinforcement fine-tuning. `https://platform.openai.com/docs/guides/reinforcement-fine-tuning`.

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 *Advances in Neural Information Processing Systems 35*.

Georgios Papoudakis, Thomas Coste, Zhihao Wu, Jianye Hao, Jun Wang, and Kun Shao. 2025. Appvlm: A lightweight vision language model for online app control. *arXiv preprint*.

Cheng Qian, Bingxiang He, Zhong Zhuang, Jia Deng, Yujia Qin, Xin Cong, Zhong Zhang, Jie Zhou, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Tell me more! towards implicit user intention understanding of language model driven agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 1088–1113.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, Chi Han, Yi R. Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, and 24 others. 2025a. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):101:1–101:40.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025b. UI-TARS: pioneering automated GUI interaction with native agents. *arXiv preprint*.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. 2025. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy P Lillicrap. 2023. Android in the wild: A large-scale dataset for android device control. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint*.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. 2025. Os-Genesis: Automating GUI agent trajectory construction via reverse task synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 5555–5579.

Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. 2025. Reason-RFT: Reinforcement fine-tuning for visual reasoning. *arXiv preprint*.

Luong Quoc Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. ReFT: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 7601–7614.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-Agent: Autonomous multi-modal

mobile device agent with visual perception. *arXiv preprint*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Weinan Gan, Xingshan Zeng, Shuai Yu, Xinlong Hao, Kun Shao, Yasheng Wang, and Ruiming Tang. 2024c. GUI agents with foundation models: A comprehensive survey. *arXiv preprint*.

Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. 2025. DistRL: An asynchronous distributed reinforcement learning framework for on-device control agent. In *The Thirteenth International Conference on Learning Representations*.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2025. OS-ATLAS: foundation action model for generalist GUI agents. In *The Thirteenth International Conference on Learning Representations*.

Xiaobo Xia and Run Luo. 2025. GUI-R1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. 2024. Aguvis: Unified pure vision agents for autonomous GUI interaction. *arXiv preprint*.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2025. Aria-UI: Visual grounding for GUI instructions. In *Findings of the Association for Computational Linguistics*, pages 22418–22433.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, and 4 others. 2024. Minicpm-v: A GPT-4V level MLLM on your phone. *arXiv preprint*.

Simon Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Peter Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. 2024. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. In *Advances in Neural Information Processing Systems 38*.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2025a. Large language model-brained GUI agents: A survey. *Transactions on Machine Learning Research*, 2025.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025b. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*.

Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024. Android in the zoo: Chain-of-action-thought for GUI agents. In *Findings of the Association for Computational Linguistics*, pages 12016–12031.

Zhuosheng Zhang and Aston Zhang. 2024. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics*, pages 3132–3149.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. A survey of large language models. *arXiv preprint*.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. 2024. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning*.

Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. 2025. R1-zero's "aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint*.

A Training Details

We list the main hyperparameters for the SFT and RFT stages in Table 4 and Table 5, respectively.

Table 4: Training parameters for Stage II: Supervised Fine-tuning.

Parameter	Default Value	Description
model_max_length	2304	Maximum sequence length
max_line_res	1120	Maximum image resolution for the longest axis
per_device_train_batch_size	1	Training batch size per device
gradient_accumulation_steps	1	Gradient accumulation steps
num_train_epochs	3	Number of training epochs
learning_rate	1e-5	Learning rate
weight_decay	0.1	Weight decay coefficient
adam_beta1	0.9	Adam optimizer beta1 parameter
adam_beta2	0.999	Adam optimizer beta2 parameter
max_grad_norm	N/A	Gradient clipping disabled
lr_scheduler_type	cosine	Learning rate scheduler type
warmup_ratio	0.05	Linear warmup ratio
bf16	True	Use bfloat16 precision
gradient_checkpointing	False	Whether using gradient checkpointing
deepspeed	ZeRO-2	Deepspeed optimization stage

Table 5: Training parameters for Stage III: Reinforcement Fine-tuning.

Parameter	Default Value	Description
max_prompt_length	16384	Maximum prompt length
max_completion_length	512	Maximum completion length
max_line_res	1120	Maximum image resolution for the longest axis
num_generations	8	Number of generations
per_device_train_batch_size	1	Training batch size per device
gradient_accumulation_steps	32	Gradient accumulation steps
learning_rate	1e-6	Learning rate
num_train_epochs	3	Number of training epochs
weight_decay	0.1	Weight decay coefficient
adam_beta2	0.99	Adam optimizer beta2 parameter
max_grad_norm	1.0	Maximum gradient norm for clipping
lr_scheduler_type	cosine	Learning rate scheduler type
beta	0.04	KL divergence coefficient
bf16	True	Use bfloat16 precision

B Evaluation Details

To ensure fair and consistent evaluation across all models, we adopt a unified evaluation framework. Since different models may define their own action formats and conventions, their outputs are first converted into a shared action representation defined by AgentCPM-GUI. This normalization allows us to compare models under the same evaluation criteria and metrics. In the following, we provide representative input prompts for each model, detail the evaluation settings and hyperparameters, and describe how action space conversion is performed when applicable.

B.1 Qwen2.5-VL-7B

B.1.1 Data example

Qwen2.5-VL-7B Data Example

System Message

You are a helpful assistant.

Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name_for_human": "mobile_use", "name": "mobile_use", "
description": "Use a touchscreen to interact with a mobile device, and take screenshots.
* This is an interface to a mobile device with touchscreen. You can perform actions like clicking,
typing, swiping, etc.
* Some applications may take time to start or process actions, so you may need to wait and take
successive screenshots to see the results of your actions.
* The screen's resolution is 1092x2408.
* Make sure to click any buttons, links, icons, etc with the cursor tip in the center of the element.
Don't click boxes on their edges unless asked.", "parameters": {"properties": {"action": {"
description": "The action to perform. The available actions are:
* `key`: Perform a key event on the mobile device.
  - This supports adb's `keyevent` syntax.
  - Examples: \"volume_up\", \"volume_down\", \"power\", \"camera\", \"clear\".
* `click`: Click the point on the screen with coordinate (x, y).
* `long\_press`: Press the point on the screen with coordinate (x, y) for specified seconds.
* `swipe`: Swipe from the starting point with coordinate (x, y) to the end point with
coordinates2 (x2, y2).
* `type`: Input the specified text into the activated input box.
* `system\_button`: Press the system button.
* `open`: Open an app on the device.
* `wait`: Wait specified seconds for the change to happen.
* `terminate`: Terminate the current task and report its completion status.", "enum": ["key", "
click", "long_press", "swipe", "type", "system_button", "open", "wait", "terminate"], "type
": "string"}, "coordinate": {"description": "(x, y): The x (pixels from the left edge) and y (
pixels from the top edge) coordinates to move the mouse to. Required only by `action=click
`, `action=long_press`, and `action=swipe`.", "type": "array"}, "coordinate2": {"
description": "(x, y): The x (pixels from the left edge) and y (pixels from the top edge)
coordinates to move the mouse to. Required only by `action=swipe`.", "type": "array"}, "

text": {"description": "Required only by `action=key`, `action=type`, and `action=open`.", "type": "string"}, "time": {"description": "The seconds to wait. Required only by `action=long\_press` and `action=wait`.", "type": "number"}, "button": {"description": "Back means returning to the previous interface, Home means returning to the desktop, Menu means opening the application background menu, and Enter means pressing the enter. Required only by `action=system\_button`", "enum": ["Back", "Home", "Menu", "Enter"], "type": "string"}, "status": {"description": "The status of the task. Required only by `action=terminate`.", "type": "string", "enum": ["success", "failure"]}}, "required": ["action"], "type": "object"}, "args_format": "Format the arguments as a JSON object."}}
</tools>

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function−name>, "arguments": <args−json−object>}
</tool_call>

User

The user query: [user_request]
Current step query: low_lew_instruction (included only when low_lew_instruction is defined)
Task progress (You have done the following operation on the current device): [history_actions]
[current_screenshot]

Assistant

[thought_and_action]

B.1.2 Action Space Mapping

Table 6 shows the action space mapping from Qwen2.5-VL-7B to the standardized representation. Two key differences must be addressed during conversion. First, Qwen2.5-VL-7B expresses duration in seconds for actions such as `long_press` and `wait`, whereas AgentCPM-GUI expects time in milliseconds. Second, Qwen2.5-VL-7B produces absolute screen coordinates (in pixels) for spatial actions like `click`, `long_press`, and `swipe`, while AgentCPM-GUI uses normalized coordinates in the range $[0, 1000]$ relative to screen size.

Table 6: Action space mapping from Qwen2.5-VL-7B to AgentCPM-GUI.

Qwen2.5-VL-7B	**Input Parameters**	**AgentCPM-GUI**
`click`	coordinate = (x, y)	`{"POINT":[int(x/width*1000), int(y/height*1000)]}`
`long_press`	coordinate = (x, y), time	`{"POINT":[x,y],"duration": time*1000}`
`swipe`	coordinate = (x1, y1), coordinate2 = (x2, y2)	`{"POINT":[x1,y1],"to": direction}` *Note: direction is computed from two points*
`type`	text	`{"TYPE":text}`
`system_button`	button = Back / Home / Enter	`{"PRESS":BACK/HOME/ENTER}`
`terminate`	None	`{"STATUS":"finish"}`
`wait`	time	`{"duration":time*1000}`

B.1.3 Hyperparameters

We adopt the same hyperparameter settings as used in Qwen2.5-VL-7B for fair comparison, as summarized in Table 7.

Table 7: Inference hyperparameters for Qwen2.5-VL-7B.

Parameter	Default Value	Description
do_sample	True	Whether to use sampling (replaces greedy)
top_p	0.01	Nucleus sampling threshold
top_k	1	Top-k sampling limit
temperature	0.01	Controls sampling randomness
repetition_penalty	1.0	Penalty factor for repetition
max_new_tokens	2048	Maximum number of new tokens to generate

B.2 UI-TARS

B.2.1 Data example

UI-TARS Data Example

System Message

You are a helpful assistant.

User

You are a GUI agent. You are given a task and your action history, with screenshots. You need to perform the next action to complete the task.

Output Format

```
Thought: . . .
Action: . . .
```

Action Space

```
click(start_box='<|box_start|>(x1,y1)<|box_end|>')
long_press(start_box='<|box_start|>(x1,y1)<|box_end|>', time='')
type(content='')
scroll(direction='down or up or right or left')
press_back()
press_home()
wait()
finished() # Submit the task regardless of whether it succeeds or fails.
```

Note

- Use English in Thought part.
- Summarize your next action (with its target element) in one sentence in Thought part.

User Instruction

```
[user_request]
```

User

[history_screenshot]

Assistant

[history_thought_and_action]

User

[current_screenshot]

Assistant(included only when low_lew_instruction is defined)

Thought: [low_lew_instruction]
Action:

Assistant

[thought_and_action]

B.2.2 Action Space Mapping

Table 8 shows the action space mapping from UI-TARS to the standardized representation. Since UI-TARS and AgentCPM-GUI define scroll directions oppositely, the direction must be reversed during conversion.

Table 8: Action space mapping from UI-TARS to AgentCPM-GUI.

UI-TARS	**Input Format**	**AgentCPM-GUI**
`click(...)`	start_box with (x, y)	`{"POINT":[x,y]}`
`long_press(...)`	start_box with (x, y), time='ms' (optional)	`{"POINT":[x,y],"duration": time (default 1000)}`
`type(...)`	content='text'	`{"TYPE":text}`
`scroll(...)`	direction='up/down/left/right'	`{"POINT":[500,500], "to":reversed direction}` *Note: direction is reversed (e.g., up → down)*
`press_back()`	-	`{"PRESS":BACK}`
`press_home()`	-	`{"PRESS":HOME}`
`wait()`	-	`{"duration":200}`
`finished()`	-	`{"STATUS":"finish"}`

B.3 OS-ATLAS

B.3.1 Data example

OS-ATLAS Data Example

System Message

You are a helpful assistant.

User

You are a foundational action model capable of automating tasks across various digital environments, including desktop systems like Windows, macOS, and Linux, as well as mobile platforms such as Android and iOS. You also excel in web browser environments. You will interact with digital devices in a human-like manner: by reading screenshots, analyzing them, and taking

appropriate actions.

Your expertise covers two types of digital tasks:
- Grounding: Given a screenshot and a description, you assist users in locating elements mentioned. Sometimes, you must infer which elements best fit the description when they aren't explicitly stated.
- Executable Language Grounding: With a screenshot and task instruction, your goal is to determine the executable actions needed to complete the task.

You are now operating in Executable Language Grounding mode. Your goal is to help users accomplish tasks by suggesting executable actions that best fit their needs. Your skill set includes both basic and custom actions:

1. Basic Actions
Basic actions are standardized and available across all platforms. They provide essential functionality and are defined with a specific format, ensuring consistency and reliability.
Basic Action 1: CLICK
- purpose: Click at the specified position.
- format: `CLICK <point>[[x-axis, y-axis]]</point>`
- example usage: `CLICK <point>[[101, 872]]</point>`

Basic Action 2: TYPE
- purpose: Enter specified text at the designated location.
- format: `TYPE [input text]`
- example usage: `TYPE [Shanghai shopping mall]`

Basic Action 3: SCROLL
- purpose: Scroll in the specified direction.
- format: `SCROLL [direction (UP/DOWN/LEFT/RIGHT)]`
- example usage: `SCROLL [UP]`

2. Custom Actions
Custom actions are unique to each user's platform and environment. They allow for flexibility and adaptability, enabling the model to support new and unseen actions defined by users. These actions extend the functionality of the basic set, making the model more versatile and capable of handling specific tasks.

Custom Action 1: LONG_PRESS
- purpose: Long press at the specified position.
- format: `LONG_PRESS <point>[[x-axis, y-axis]]</point>`
- example usage: `LONG_PRESS <point>[[101, 872]]</point>`

Custom Action 2: PRESS_BACK
- purpose: Press a back button to navigate to the previous screen.
- format: `PRESS_BACK`
- example usage: `PRESS_BACK`

Custom Action 3: PRESS_HOME
- purpose: Press a home button to navigate to the home page.
- format: `PRESS_HOME`

- example usage: `PRESS_HOME`

Custom Action 4: PRESS_RECENT
- purpose: Press the recent button to view or switch between recently used applications.
- format: `PRESS_RECENT`
- example usage: `PRESS_RECENT`

Custom Action 5: WAIT
- purpose: Wait for the screen to load.
- format: `WAIT`
- example usage: `WAIT`

Custom Action 6: COMPLETE
- purpose: Indicate the task is finished.
- format: `COMPLETE`
- example usage: `COMPLETE`

In most cases, task instructions are high-level and abstract. Carefully read the instruction and action history, then perform reasoning to determine the most appropriate next action. Ensure you strictly generate two sections: Thoughts and Actions.
Thoughts: Clearly outline your reasoning process for current step.
Actions: Specify the actual actions you will take based on your reasoning.

Your current task instruction, action history, and associated screenshot are as follows:
Screenshot:[current_screenshot]
Task: [user_request] You need to: [low_lew_instruction](included only when low_lew_instruction is defined)
History:
[history_low_lew_instruction](included only when low_lew_instruction is defined)

Assistant

[thought_and_action]

B.3.2 Action Space Mapping

Table 9 shows the action space mapping from OS-ATLAS to the standardized representation. When evaluating the AndroidControl-Low setting, we found that the model's predicted scroll direction is often opposite to that indicated in the low-level instruction. Therefore, the scroll direction is reversed during evaluation.

B.4 OS-Genesis

B.4.1 Data Example

For the GUI-Odyssey, AITZ, and CAGUI benchmarks, we construct evaluation prompts following the format described in Data Example. For AndroidControl, we adopt the official evaluation code provided in the benchmark's GitHub repository.

OS-Genesis Data Example

System Message

You are a helpful assistant.

Table 9: Action space mapping from OS-Atlas to AgentCPM-GUI.

OS-Atlas	Input Format	AgentCPM-GUI
CLICK	[[x, y]]	{"POINT":[x,y]}
LONG_PRESS	[[x, y]]	{"POINT":[x,y],"duration":1000}
TYPE	[text]	{"TYPE":text}
SCROLL	[direction]	{"POINT":[500,500],"to":direction} *Note: if use_low_instruction is True, direction is reversed: up↔down, left↔right*
PRESS_BACK	-	{"PRESS":BACK}
PRESS_HOME	-	{"PRESS":HOME}
PRESS_RECENT	-	{"PRESS":RECENT}
WAIT	-	{"duration":200}
COMPLETE	-	{"STATUS":"finish"}

User

You are a GUI task expert, I will provide you with a high-level instruction, an action history, a screenshot with its corresponding accessibility tree.

High-level instruction: [user_request]
Action history:
Accessibility tree:
Please generate the low-level thought and action for the next step.

Assistant

[thought_and_action]

B.4.2 Action Space Mapping

Table 10 shows the action space mapping from OS-Genesis to the standardized representation. Similar to OS-ATLAS, the predicted scroll direction on AndroidControl-Low is often opposite to the instruction, and is therefore reversed during evaluation.

Table 10: Action space mapping from OS-Genesis to AgentCPM-GUI.

OS-Genesis	Input Fields	AgentCPM-GUI
type	text	{"TYPE":text}
click	x, y	{"POINT":[x,y]}
long_press	x, y	{"POINT":[x,y],"duration":1000}
dismiss	x, y	{"POINT":[x,y]}
get_text	x, y	{"POINT":[x,y]}
navigate_home	-	{"PRESS":HOME}
navigate_back	-	{"PRESS":BACK}
scroll	direction	{"POINT":[500,500],"to":direction} *Note: If use_low_instruction is True, direction is reversed: up↔down, left↔right*
wait	-	{"duration":200}

B.5 OdysseyAgent

B.5.1 Data example

Following the official implementation, OdysseyAgent's input consists of the current instruction along with a history of images and their associated actions.

OdysseyAgent Data Example

System Message

You are a helpful assistant.

User

Picture 1: <img>image_path</img>
I'm looking for guidance on how to [instruction]
Previous screenshots: <img>image-history: image_path</img>
Previous Actions: 1. [Action 1]
2. [Action 2].
...

Assistant

[Action]

B.5.2 Action Space Mapping

Table 11 shows the action space mapping from OdysseyAgent to the standardized representation. The output format of OdysseyAgent is largely compatible with AgentCPM-GUI. The only exception is the `RECENT` action, which is not part of the AgentCPM-GUI action space and is therefore ignored during evaluation.

Table 11: Action space mapping from OdysseyAgent to AgentCPM-GUI.

OdysseyAgent	Input Fields	AgentCPM-GUI
`CLICK`	x, y	`{"POINT":[x,y]}`
`LONG_PRESS`	x, y	`{"POINT":[x,y],"duration":1000}`
`SCROLL`	direction	`{"POINT":[500,500],"to":direction}`
`TYPE`	text	`{"TYPE":text}`
`HOME`	-	`{"PRESS":HOME}`
`BACK`	-	`{"PRESS":BACK}`
`COMPLETE`	-	`{"STATUS":"finish"}`
`IMPOSSIBLE`	-	`{"STATUS":"impossible"}`

B.5.3 Hyperparameters

We follow the original implementation for inference, enabling the `image_history` option to incorporate temporal context. Specifically, we store the last 4 actions and their corresponding images. The inference is conducted with the torch seed set to 1234 and the random seed set to 2020 to ensure reproducibility.

B.6 Aguvis-7B

B.6.1 Data Example

Aguvis Data Example

System Message

You are a GUI agent. You are given a task and a screenshot of the screen. You need to perform a series of pyautogui actions to complete the task.
You have access to the following functions:
- {"name": "mobile.swipe", "description": "Swipe on the screen", "parameters": {"type": "object", "properties": {"from_coord": {"type": "array", "items": {"type": "number"}, "description": "The starting coordinates of the swipe"}, "to_coord": {"type": "array", "items": {"type": "number"}, "description": "The ending coordinates of the swipe"}}, "required": ["from_coord", "to_coord"]}}
- {"name": "mobile.home", "description": "Press the home button"}
- {"name": "mobile.back", "description": "Press the back button"}
- {"name": "mobile.wait", "description": "wait for the change to happen", "parameters": {"type": "object", "properties": {"seconds": {"type": "number", "description": "The seconds to wait"}}, "required": ["seconds"]}}
- {"name": "mobile.long_press", "description": "Long press on the screen", "parameters": {"type": "object", "properties": {"x": {"type": "number", "description": "The x coordinate of the long press"}, "y": {"type": "number", "description": "The y coordinate of the long press"}}, "required": ["x", "y"]}}
- {"name": "mobile.open_app", "description": "Open an app on the device", "parameters": {"type": "object", "properties": {"app_name": {"type": "string", "description": "The name of the app to open"}}, "required": ["app_name"]}}

User

Please generate the next move according to the ui screenshot, instruction and previous actions.
Instruction: [Instruction]
Previous actions: [previous_actions]

Assistant

[thought and Action]

Table 12: Action space mapping from Aguvis to AgentCPM-GUI.

Aguvis	**Input Fields**	**AgentCPM-GUI**
`pyautogui.click`	x, y	`{"POINT":[x*1000,y*1000]}`
`mobile.long_press`	x, y	`{"POINT":[x*1000,y*1000], "duration":1000}`
`pyautogui.scroll()/hscroll()`	direction	`{"POINT":[500,500],"to": direction}` *Note:* `scroll` *performs vertical, and* `hscroll` *performs horizontal swipes*
`pyautogui.write`	text	`{"TYPE":text}`
`mobile.home()/`	-	`{"PRESS":HOME}`
`mobile.back()`	-	`{"PRESS":BACK}`
`mobile.terminate()`	-	`{"STATUS":"finish"}`
`mobile.open_app`	app_name	-
`mobile.wait`	[time]	`{"duration":3000}`

B.6.2 Action Space Mapping

Table 12 shows the action space mapping from Aguvis to the standardized representation. All coordinates in Aguvis are in the range $[0, 1]$ and are scaled accordingly during conversion. Swipe actions are mapped following the definition in the `pyautogui` package. Since AgentCPM-GUI does not include an "open app" action, it is ignored during evaluation.

B.6.3 Hyperparameters

The hyper parameters are the same as the origin implementation. To be specific, we choose "self-plan" mode during inference, with temperature set as 0 and generate only 1024 new max tokens. Historical actions are not included during inference, as their inclusion leads to abnormal model behavior.

C CAGUI Benchmark

C.1 CAGUI_Grounding

We provide examples from the three tasks that constitute the grounding benchmark, each containing 1,500 samples. The Text2Bbox and Bbox2Text tasks are based on the same dataset. Each bounding box is defined by four absolute coordinates in the format <$x_{\min}$, $y_{\min}$, $x_{\max}$, $y_{\max}$>, with the origin located at the top-left corner of the screen.

Text2Point Data Examples
Text
QQ音乐
Bounding Box
<643, 462, 849, 744>
Prompt of AgentCPM-GUI
你是一个GUI组件定位的专家，擅长输出图片上文本对应的坐标。你的任务是根据给定的GUI截图和图中某个文本输出该文本的坐标。输入：屏幕截图，文本描述输出：文本的相对坐标的中心点,POINT:[...,...]为格式

Bbox2Text Data Examples
Bounding Box
<60, 120, 132, 192>
Bounding Box
返回
Prompt of AgentCPM-GUI
你是一个GUI组件文字识别的专家，擅长根据组件的边界框（bounding box）描述输出对应的文字。你的任务是根据给定的GUI截图和图中某个组件的边界框输出组件的中的文字。输入：屏幕截图，边界框的坐标输出：组件中的文本

Fun2Point Data Examples

Function

UI元素是一个菜单按钮。其主要功能是弹出一个菜单面板，允许用户选择不同的功能选项。通常可以通过点击该按钮触发，点击后会展示一个下拉或侧滑菜单，用户可以在其中进行进一步操作，例如切换功能页面或设置选项。

Bounding Box

<1061, 2424, 1159, 2522>

Prompt of AgentCPM-GUI

你是一个GUI组件定位的专家，擅长根据组件的功能描述输出对应的坐标。你的下一步操作是根据给定的GUI截图和图中某个组件的功能描述点击组件的中心位置。坐标为相对于屏幕左上角位原点的相对位置，并且按照宽高比例缩放到0～1000 输入：屏幕截图，功能描述输出：点击操作，以POINT:[...,...]为格式，其中不能存在任何非坐标字符

C.2 CAGUI_Agent

We present examples of our dataset tasks, each consisting of a query, a screenshot, and the corresponding answer operation. The system prompt used to evaluate AgentCPM-GUI is also included. In total, the benchmark comprises 600 tasks, which together contain 4,516 single-step images. During evaluation, inputs to AgentCPM-GUI follow the standard chat format. Each user message contains both the task query and the associated screenshot, structured as a list with two elements: a text string formatted as "<Question>{query}</Question>\n当前屏幕截图：" and the corresponding image.

Agent Data Examples

Query

请优酷视频根据我的历史记录播放7天内观看超过60%的短视频。

Operation

Action Type: Click
Action Detail: [0.13, 0.61]

System Prompt of AgentCPM-GUI

 # Role
你是一名熟悉安卓系统触屏GUI操作的智能体，将根据用户的问题，分析当前界面的GUI元素和布局，生成相应的操作。

 # Task
针对用户问题，根据输入的当前屏幕截图，输出下一步的操作。

 # Rule
- 以紧凑JSON格式输出
- 输出操作必须遵循Schema约束

 # Schema

```
{
  "type": "object",
  "description": "执行操作并决定当前任务状态",
```

```
  "additionalProperties": false,
  "properties": {
    "thought": {
      "type": "string",
      "description": "智能体的思维过程"
    },
    "POINT": {
      "$ref": "#/$defs/Location",
      "description": "点击屏幕上的指定位置"
    },
    "to": {
      "description": "移动，组合手势参数",
      "oneOf": [
        {
          "enum": [
            "up",
            "down",
            "left",
            "right"
          ],
          "description": "从当前点（POINT）出发，执行滑动手势操作，方向包括向
上、向下、向左、向右"
        },
        {
          "$ref": "#/$defs/Location",
          "description": "移动到某个位置"
        }
      ]
    },
    "duration": {
      "type": "integer",
      "description": "动作执行的时间或等待时间，毫秒",
      "minimum": 0,
      "default": 200
    },
    "PRESS": {
      "type": "string",
     "description": "触发特殊按键，HOME为回到主页按钮，BACK为返回按钮，ENTER为
回车按钮",
      "enum": [
        "HOME",
        "BACK",
        "ENTER"
      ]
    },
    "TYPE": {
      "type": "string",
      "description": "输入文本"
    },
    "STATUS": {
```

```
      "type": "string",
           "description": "当前任务的状态。特殊情况：satisfied，无需操作；impossible，任务无法完成；interrupt，任务中断；need_feedback，需要用户反馈；",
      "enum": [
        "continue",
        "finish",
        "satisfied",
        "impossible",
        "interrupt",
        "need_feedback"
      ],
      "default": "continue"
    }
  },
  "$defs": {
    "Location": {
      "type": "array",
       "description": "坐标为相对于屏幕左上角位原点的相对位置，并且按照宽高比例缩放到0～1000，数组第一个元素为横坐标x，第二个元素为纵坐标y",
      "items": {
        "type": "integer",
        "minimum": 0,
        "maximum": 1000
      },
      "minItems": 2,
      "maxItems": 2
    }
  }
}
```

D Case Study

We demonstrate GUI agent tasks on a real Xiaomi 12S device running MIUI 14.0.11. All interactions with the graphical interface are carried out via ADB control based on the AgentCPM-GUI's predicted actions. The original input and output were in Chinese and translated into English.

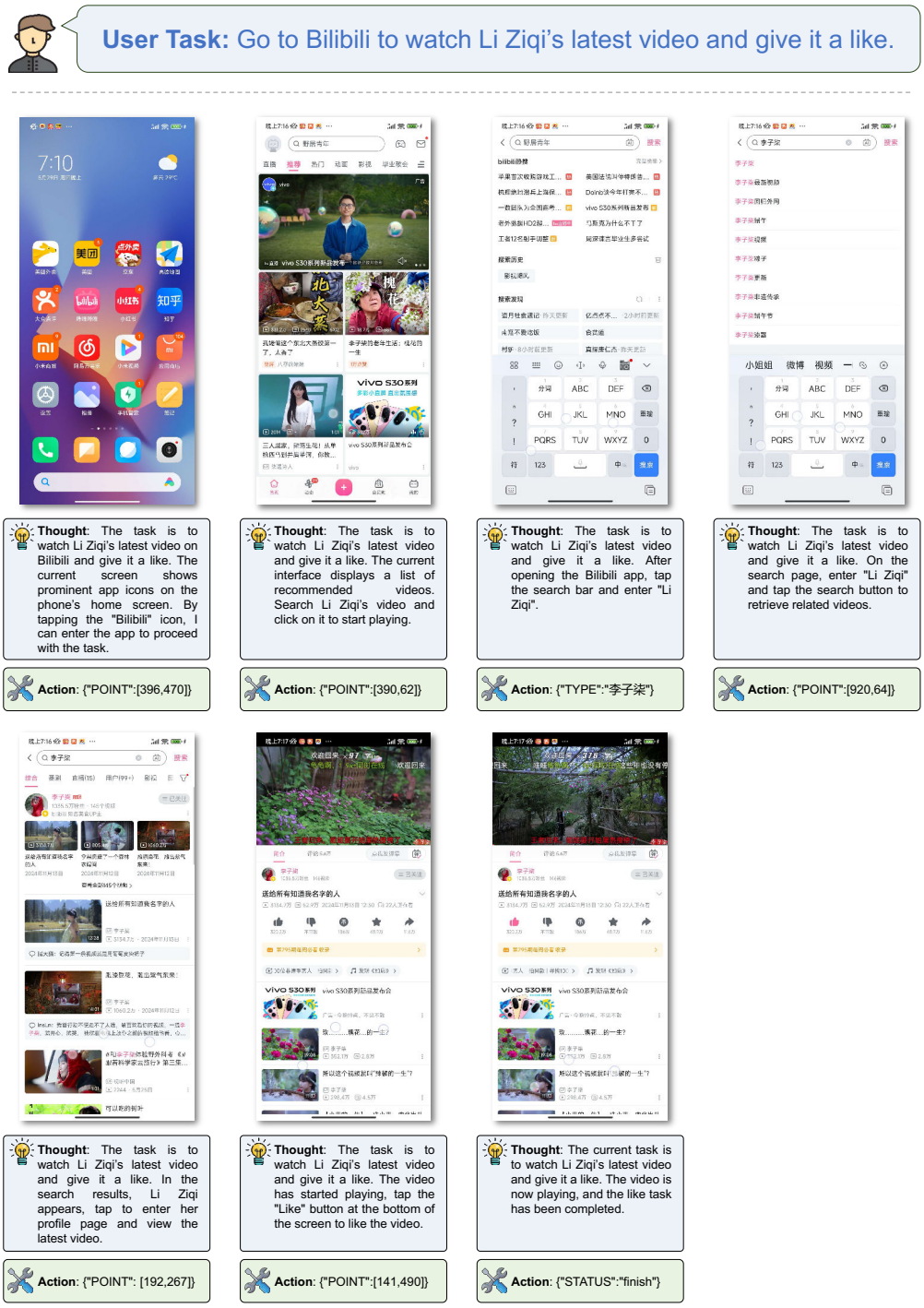


Figure 2: A demo case on the BiliBili.

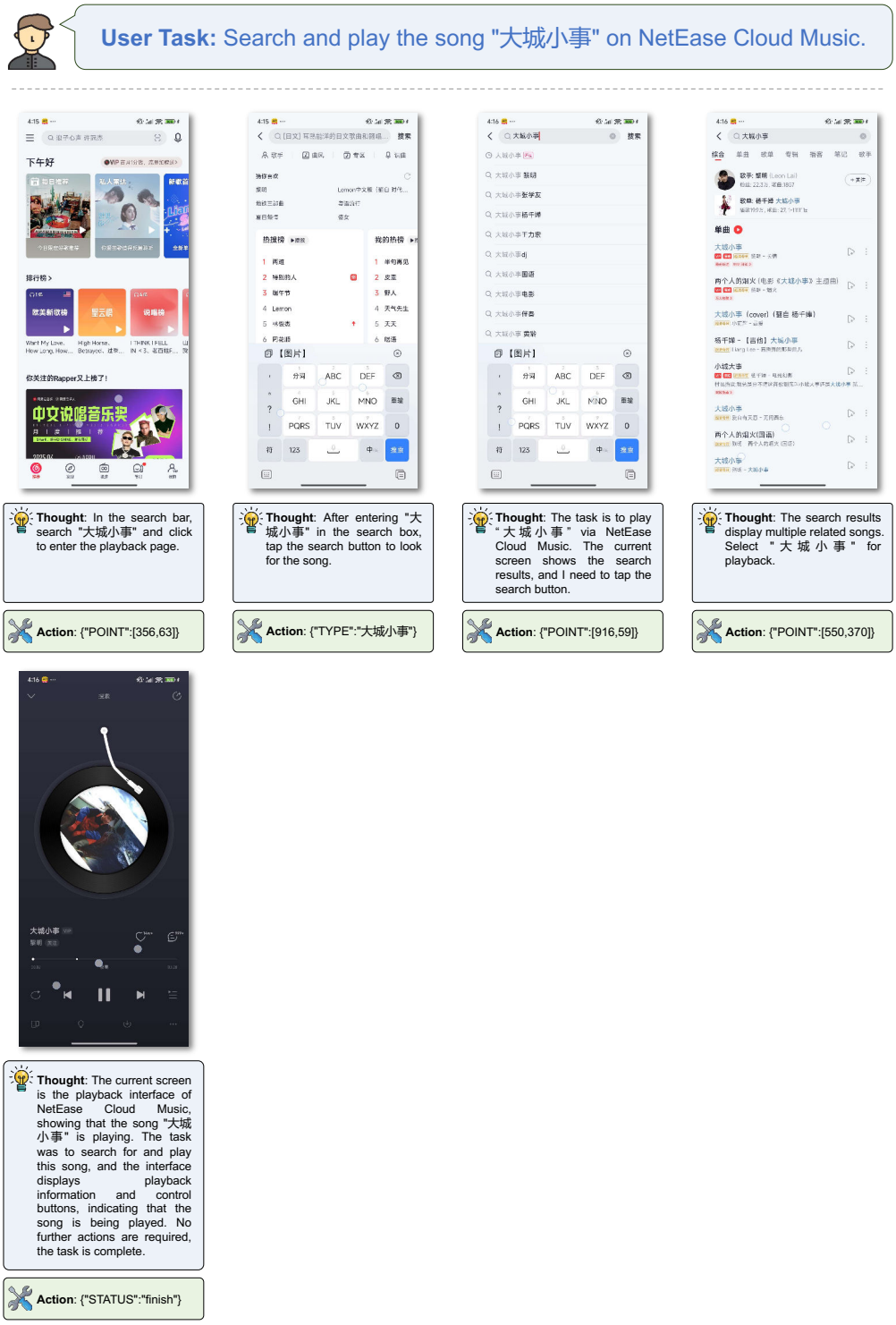


Figure 3: A demo case on the NetEase Cloud Music.

Marcel: A Lightweight and Open-Source Conversational Agent for University Student Support

Jan Trienes **Anastasiia Derzhanskaia** **Roland Schwarzkopf**
Markus Mühling **Jörg Schlötterer** **Christin Seifert**
Marburg University
{jan.trienes,christin.seifert}@uni-marburg.de

Abstract

We present *Marcel*, a lightweight and open-source conversational agent designed to support prospective students with admission-related inquiries. The system aims to provide fast and personalized responses, while reducing workload of university staff. We employ retrieval-augmented generation to ground answers in university resources and to provide users with verifiable, contextually relevant information. We introduce a Frequently Asked Question (FAQ) retriever that maps user questions to knowledge-base entries, which allows administrators to steer retrieval, and improves over standard dense/hybrid retrieval strategies. The system is engineered for easy deployment in resource-constrained academic settings. We detail the system architecture, provide a technical evaluation of its components, and report insights from a real-world deployment.

github.com/aix-group/marcel-chat
youtu.be/uLCB2R6szz4

1 Introduction

Prospective university students often face challenges navigating the admission requirements. While many universities are investing in student support services, these typically do not grow at the same rate as many study programs. To contextualize this problem in a real-world example, we consider the M.Sc Data Science program at our institution. The growing interest in this subject and a recent internationalization of the program lead to a stark increase in applications. However, many applications had to be rejected because formal criteria or prerequisites were not met. This substantially increased the workload of university staff and led to missed opportunities for students. Expanding support services with alternative information channels appears necessary to better inform applicants and to reduce the workload of university staff.

A potential solution to this problem are conversational search systems like chatbots (Zamani et al., 2023; Mo et al., 2025). We posit that these systems make it easier for students to find and understand relevant information due to their interactivity. Recent advances in Retrieval Augmented Generation (RAG, Lewis et al., 2020) allow language models to provide grounded and verifiable answers based on university resources. However, operating such a RAG-based system in a university setting is challenging. Many universities require on-premise deployments to comply with privacy laws, have limited computational resources, and need cost-effective systems that are easy to customize, maintain and monitor. Several universities are actively developing or using such chatbots (Nguyen and Quan, 2025; Odede and Frommholz, 2024; Cherumanal et al., 2024; Brereton, 2021) which indicates high demand. However, they are typically closed-source which limits others to build on those efforts.

To bridge this gap, we introduce *Marcel* a student chatbot aimed at answering enrollment-related questions.[1] This demo makes three contributions:

(1) FAQ Retriever (Section 2). We introduce a novel retriever which maps user queries to a set of curated frequently asked questions with associated answer-relevant documents. This component allows administrators to steer retrieval, and improves over dense/hybrid retrieval strategies.

(2) Efficient Deployment (Section 3). We outline our design decisions to facilitate deployments with minimal infrastructure (i.e., a single node with a GPU). We empirically evaluate several retrieval and generation strategies to better understand quality-efficiency tradeoffs in this task.

(3) Practical Insights (Section 4). We share qualitative insights from deploying the chatbot in practice. To support adoption, we release our system as containerized software along with support-

[1] For this demo, we focus on enrollment-related questions for the M.Sc Data Science program at our institution. However, all design decisions are made to generalize to other information-seeking tasks in the university setting.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 181–195
November 4-9, 2025

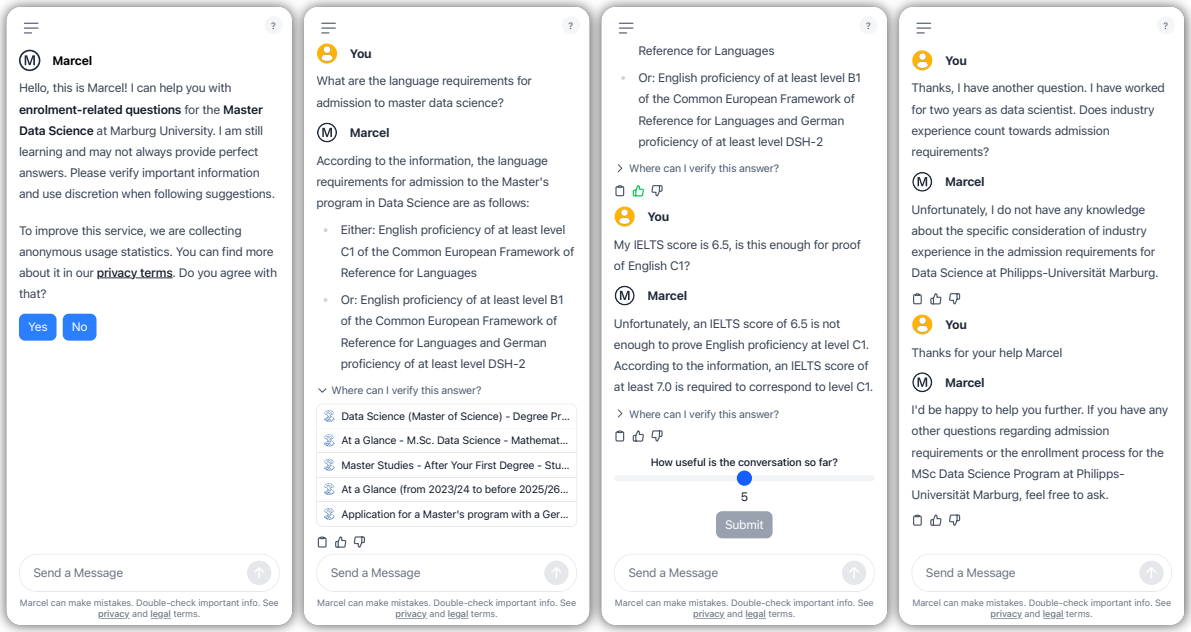


Figure 1: Overview of student-facing UI. ① Welcome and consent are shown on first use. ② For each generated answer, Marcel provides the list of source documents which are linked to the respective websites. ③ Users can rate individual responses using a thumbs up/down button. The overall conversation can be rated after three interactions on a 10-point Likert scale. ④ Marcel abstains from answering questions that are not reflected in the underlying knowledge base. In those cases, no list of source documents is shown. An admin-facing UI is given in Figure 6.

ing artifacts such as content scraper and deployment scripts to enable customization.[2]

2 System Description

2.1 Question Answering

Document retrieval. *Marcel* uses RAG to provide answers grounded in university resources. To improve retrieval precision and add a layer of administrative control, we introduce an FAQ retriever that maps queries to a set of curated questions, each linked to one or more relevant documents. The closest FAQ is selected through semantic similarity of question embeddings. Intuitively, FAQ retrieval acts as a lightweight intent classifier. We combine it with standard retrievers (e.g., BM25) to provide fallbacks for question intents falling outside the FAQ. Although FAQ curation requires manual effort, many universities maintain FAQ-like resources (e.g., for student support services) that can be bootstrapped. For our deployment, we worked with university staff to establish 36 FAQs, reflecting most enrollment related information-needs. We provide example FAQs in Appendix, Table 5.

Answer generation. An instruction-tuned Large Language Model (LLM) generates answers conditioned on retrieved documents. To improve answer grounding and dialogue handling, answering is done in two stages. First, an LLM-based classifier determines whether a query requires retrieval or is chit-chat or a meta-question (e.g., "What topics can you help with?"). If so, the system responds without retrieval. Otherwise, the LLM is instructed to answer only if the retrieved documents provide sufficient information, or to abstain and return a fallback response (e.g., "Unfortunately, I cannot answer based on..."). See Appendix B for the prompts of query classifier and answer generator.

Knowledge base. We scrape enrollment-related documents from university websites (e.g., study descriptions, admission requirements and procedures, course catalog, and accommodation services). Curating the knowledge base through scraping is efficient, since latest information is published on the websites. Periodically re-running the scraper captures changes automatically (e.g., updated admission requirements). Pre-processing of scraped content is kept minimal: to reduce token count in downstream generation, we convert HTML to Markdown and replace links with numeric IDs. We avoid splitting documents to preserve context. The

[2]All software is released under the MIT license.

corpus consists of 249 documents, with an average length of 712 tokens using the NLTK tokenizer.

2.2 User Interaction

We support two user groups: (i) students seeking information, and (ii) administrators wanting to monitor and evaluate the system. The student-facing UI adopts familiar conventions of chat interfaces and is fully responsive (Figure 1). On first visit, a consent dialog informs users about data collection and chatbot limitations. The disclaimer states that generated answers may be inaccurate and must be verified. Users need to confirm that they have read this message before continuing. We discuss broader ethical implications of the chatbot at the end of the paper.

Responses are shown alongside links to retrieved sources to support verification. Users can rate each response with a thumbs-up/down, and after several turns, provide conversation-level feedback on a Likert scale. If a query is not answerable with retrieved context, the system abstains and hides sources. The admin UI shows usage statistics over a user-defined period and allows to review full conversations and ratings (Appendix, Figure 6).

2.3 Software Architecture and Deployment

Marcel is designed for fast deployment in resource-constrained academic settings. The chatbot is implemented as a containerized architecture with four components: a JavaScript frontend (Vue.JS), a Python backend (FastAPI), a relational database (MariaDB), and an LLM inference engine (vLLM, Kwon et al., 2023). A reverse proxy (nginx) routes traffic and handles HTTPs encryption. We run Marcel on a node with 2 A100 GPUs for vLLM[3] and a VM (4 cores, 4GB RAM) for the remaining containers. We implement best practices for container security by OWASP (2025), including (i) container isolation, (ii) minimal rights, (iii) not exposing the Docker socket. Automated, reproducible deployments and updates are done using Ansible. To support adoption by other institutions, we share both the Ansible scripts and regular docker-compose files as a blueprint.

3 Technical Evaluation

We conduct a technical evaluation of *Marcel's* core components to better understand the quality and efficiency trade-offs in this resource-constrained university chatbot setting. Our experiments are guided by the following research questions:

RQ1. How does the FAQ-augmented retriever compare to standard retrieval strategies?
RQ2. To what extent can a second-stage reranker improve retrieval quality?
RQ3. How well do small-scale, open-weights generators perform in this domain?
RQ4. What is the impact of quantization on generation quality and efficiency?

3.1 Experimental Systems

Retrievers. To address RQ1 and RQ2, we evaluate diverse retrieval strategies commonly used in production systems, spanning lexical, dense, and hybrid methods, with and without reranking.

- **BM25** (Robertson and Zaragoza, 2009). Standard lexical search, which is a strong baseline in practical settings.
- **Dense**. Document relevance is determined by the similarity between query and document embeddings. Document embeddings are precomputed during indexing using models optimized for semantic search in sentence-transformers (Reimers and Gurevych, 2019). Specifically: `MiniLM` (22.7M params) and `MS MARCO Bert` (109M params).
- **HyDE** (Gao et al., 2023). An LLM is prompted to generate n artificial documents that could plausibly answer a given query. The average embedding of these documents is used as the query, similar to dense retrieval. To understand how generator scale affects HyDE performance, we test the Gemma 3 model family (Kamath et al., 2025). We set $n = 3$ and use the same embedding model as the dense retriever (MS MARCO).
- **FAQ**. Retrieving relevant documents via FAQ matching (see Section 2.1).

We evaluate combinations of above retrievers by merging document rankings through reciprocal rank fusion (Cormack et al., 2009). To complement these retrievers, we evaluate second-stage reranking using cross-encoders (**Rerank**).[4] For retrieval evaluation, we take the top $k = 50$ documents. For

[3]The newer Gemma 27b model runs on a single A100. With 4bit quantization a 3090 is supposed to be sufficient.

[4]We use `mxbai-rerank-base-v1`, which performs well on general benchmarks (Shakir et al., 2024); See Appendix A for an evaluation of other representative models in the 33M to 435M parameter range.

generation, we restrict to the top $k = 5$ documents to meet latency and context length constraints.

Generators. We seek a generator that is (i) open-weight, allowing for a local and privacy-compliant operation, and (ii) that is deployable in hardware constrained settings (e.g., 1–2 GPUs). The current deployment of *Marcel* uses Llama 3.1 (Grattafiori et al., 2024), in the 70B variant with int8 quantization. For comparison, we benchmark a second candidate, Gemma 3 (Kamath et al., 2025), because of its strong performance in the Chatbot Arena (Chiang et al., 2024).[5] To address RQ3 and RQ4, we evaluate all available model sizes for both Llama and Gemma families, and assess the impact of int8 quantization for Llama models. As retriever we use BM25+FAQ. Additionally, we use an Oracle to estimate generation quality under optimal retrieval. For each query, we generate three outputs with temperature sampling ($\tau = 0.7$), and report evaluation metrics averaged over these generations. The system prompt is provided in Appendix, Listing 1.

3.2 Evaluation Protocol

Data. In the initial development phase of *Marcel*, we lacked real user queries directed at the chatbot. Therefore, we bootstrapped an evaluation dataset from student emails sent to university support offices. This provides a realistic approximation of the information needs that end-users have. To simplify the analysis, we focus on single-turn interactions.

Emails were collected and anonymized during a time frame of 6 months. We performed light deduplication of overrepresented information needs. Each question is associated with a manually written ground-truth response and the relevant sources in the knowledge base (i.e., document IDs). We distinguish between questions that are *answerable* and *unanswerable* with the knowledge base, to assess the generator's ability to abstain when the retrieved context does not provide relevant information (e.g., questions about visa issues, late arrivals, conditional accepts, etc.). See Table 1 for an overview of the dataset and Appendix, Table 6 for examples.

Metrics. We evaluate retrieval by Mean Reciprocal Rank (**MRR**) and Recall at Cutoff (**R@K**). To evaluate generation, we employ both reference-based and reference-free metrics, using LLM-as-a-Judge. For reference-based metrics, we report **ROUGE-1** (Lin, 2004) and **BERTScore** (Zhang et al., 2020). For reference-free evaluation, we report **Answer Faithfulness** and **Answer Relevance** (Es et al., 2024). Answer faithfulness estimates the fraction of answer claims supported by the retrieved context. Answer relevance generates n hypothetical questions from the answer, and measures their similarity to the original query.

Statistic	All	Answerable	Unanswerable
Questions	95	76	19
Question length	43.3	40.5	54.3
Answer length	45.5	46.0	43.6
Answer sources	1.0	1.3	0.0
FAQs	36	25	11
FAQ length	9.7	8.9	12.8

Table 1: Summary statistics of the evaluation dataset.

Additionally, we evaluate the generator's ability to abstain when a query cannot be answered with the knowledge base. Abstaining is important, as any answer generated from the parametric knowledge of the LLM is not grounded in the retrieved context and may therefore be incorrect, irrelevant or outdated. To this end, we define **Selectivity**. Let $a_{i,j} \in \{0, 1\}$ indicate whether generation j is an attempt at answering query i, and let $y_i \in \{0, 1\}$ denote the ground-truth obtained from manual annotation. Given N queries and M generations,

$$\text{Selectivity} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{M} \sum_{j=1}^{M} \mathbb{1}[a_{i,j} = y_i] \right),$$

where $\mathbb{1}$ is the indicator function. Selectivity ranges from 0 to 1, where higher is better. To obtain $a_{i,j}$, we instruct an LLM to detect hedging phrases (e.g., *I don't know*, *The documents do not provide any information*). See prompt in Appendix, Listing 4. We report all generator metrics separately for *answerable* and *unanswerable* queries. This allows us to understand if the system abstains correctly when questions cannot be answered.

Finally, we report computational efficiency as **seconds per query**. Experiments were conducted on a Slurm node with four CPU cores and two A100 GPUs (80GB), with timings collected after pipeline warmup. All LLM-as-a-Judge metrics are computed using Llama 3.1 70B (int8 quantized).[6]

[5] At the time of study (June 2025), Gemma 3 was the highest ranked model that can be deployed on a single GPU (ranked 28/254). The next better open-weights model was Qwen 3 with 235B parameters (ranked 23/254).

[6] While Llama could be favoring its own generations, Panickssery et al. (2024) showed that this mainly applies to pairwise evaluations, not direct ratings used in our experiments.

Retriever	MRR	R@1	R@5	R@50	sec/q	sec/q†
Baseline						
BM25	0.26	0.09	0.37	0.74	**0.02**	—
+ Dense (MINILM)	0.31	0.15	0.44	0.82	0.06	0.04
+ Dense (MARCO)	0.40	0.21	0.50	0.78	0.23	0.06
Hyde (Gao et al., 2023)						
+ HyDE (G-1B)	0.28	0.13	0.34	0.73	—	4.81
+ HyDE (G-4B)	0.38	0.25	0.41	0.82	—	5.22
+ HyDE (G-27B)	0.44	0.29	0.49	**0.84**	—	20.92
FAQ Retriever (Ours)						
+ FAQ (MINILM)	**0.70***	**0.57***	**0.77***	**0.84**	0.05	**0.03**
+ FAQ (MARCO)	0.61*	0.47*	0.63	0.80	0.19	0.03
Reranker (`mxbai-rerank-base-v1`)						
+ Dense + Rerank	0.46	0.33	0.48	0.78	—	0.59
+ HyDE + Rerank	0.53	0.38	0.55	0.83	—	21.41
+ FAQ + Rerank	0.48	0.34	0.49	**0.84**	—	0.56

Table 2: Evaluation of retrieval strategies. Statistically significant differences relative to baseline (BM25 + DENSE MARCO) are denoted by * at a significance level of $\alpha = 0.01/40$ (Bonferroni correction). Seconds per query are reported on CPU (sec/q) and GPU (sec/q†), while HyDE and Rerank are only tractable on GPU.

3.3 Results

We present the evaluation of retrievers in Table 2 and the evaluation of generators in Table 3.

Effectiveness of FAQ retriever (RQ1). The FAQ retriever significantly outperforms standard retrieval methods, achieving a 75% increase in MRR over the BM25+Dense baseline. This gain in early-rank retrieval quality is particularly valuable for downstream generation, which typically benefits from smaller context sizes both in output quality and throughput. Importantly, strong performance is achieved even with lightweight embeddings (MiniLM, 22.7M parameters), enabling fast, CPU-friendly retrieval with low latency (0.05 seconds). Overall, these findings suggest that the FAQ retriever is an effective mechanism for administrators to steer retrieval, with the trade-off of manual FAQ curation and maintenance.

Effectiveness of reranking (RQ2). We next examine the impact of second-stage reranking on retrieval quality. While both BM25+Dense and BM25+HyDE retrieval show modest improvements at early ranks, the quality of FAQ retrieval is substantially degraded ($\approx 31\%$ drop in MRR). This suggests that current rerankers may not generalize to this domain. One potential mitigation strategy is to exclude FAQ-retrieved documents from reranking. However, this approach is only feasible when the FAQ retriever reliably returns relevant documents. Future work could investigate techniques to dynamically select the most appropriate retriever per query, or to fine-tune rerankers.

Generator evaluation (RQ3). We find that Gemma (12B, 27B) and Llama (70B) perform comparably, especially on the *answerable* subset of queries (Table 3). Gemma (12B) strikes a good balance between quality and resource efficiency, showing that it may not be necessary to resort to the largest available models. We note an interesting model-dependent trend on response length: larger Llama models tend to produce longer responses, while Gemma models show the opposite pattern. However, we cannot conclude that either behavior leads to higher response quality.

Substituting the FAQ retriever with an oracle retriever results in minimal changes in performance on answerable queries. This highlights the quality of the FAQ retriever, but also indicates that the current evaluation protocol may be insensitive to improvements beyond a certain threshold. On unanswerable queries, the use of an oracle retriever significantly increases the rate of correct abstentions (0.68→0.93), suggesting that more conservative retrieval can further improve faithfulness. Additionally, it may be worthwhile to investigate alternative strategies for classifying whether or not a question is answerable given the retrieved context (Peng et al., 2025). Currently, this task is left to the answer generation prompt (cf. Section 2.1) which may leave room for improvement.

Impact of quantization (RQ4). We find that int8 quantization has no noticeable effect on generation quality for both Llama 8B and 70B. At the same time, it significantly reduces inference latency (speedup of $\approx 20\%$–42%). This makes quantization a suitable technique to reduce hardware requirements for practical deployments of *Marcel*.

4 Insights from End-User Deployment

We deployed Marcel during the final weeks of the current admissions cycle (winter term 2025/26). Despite limited user engagement due to the timing, this deployment demonstrates production-readiness and allowed us to collect qualitative insights from end-user interactions. Over a nine-week period, Marcel served 92 users, exchanging 926 messages in 126 conversations, without any system errors or downtime. Below, we discuss several qualita-

Generator	ROUGE	BERTScore	Faithfulness	Relevance	Selectivity	Average	$\lvert\hat{y}\rvert$	sec/q	GPUs
Answerable with knowledge base									
Gemma 3 (1B)	0.17	-0.03	0.33	0.73	0.38	0.32	142	7.2†	1
Gemma 3 (4B)	0.24	0.04	0.56	0.72	**0.88**	0.49	206	3.3	1
Gemma 3 (12B)	0.30	0.15	**0.76**	0.71	0.79	0.54	113	4.1	1
Gemma 3 (27B)	**0.31**	**0.16**	0.71	0.70	**0.88**	**0.55**	114	8.0	1
→ Oracle retriever	0.35	0.18	0.63	0.68	0.91	0.55	126	6.8	1
Llama 3.1 (8B)	0.22	0.10	0.54	0.74	0.44	0.41	89	2.4	1
→ int8 quantized	0.20	0.07	0.54	**0.75**	0.43	0.40	93	1.9	1
Llama 3.1 (70B)	0.29	0.16	0.70	0.74	0.76	0.53	132	12.5	2
→ int8 quantized	0.29	0.16	0.73	0.73	0.80	0.54	130	8.3	2
Not Answerable with knowledge base									
Gemma 3 (1B)	0.16	0.02	0.31	0.64	0.65	0.32	181	7.1†	1
Gemma 3 (4B)	0.16	0.04	0.54	0.61	0.56	0.49	161	2.9	1
Gemma 3 (12B)	0.21	**0.14**	0.70	0.56	0.75	0.54	59	2.7	1
Gemma 3 (27B)	**0.21**	0.12	**0.75**	0.60	0.68	**0.55**	80	6.5	1
→ Oracle retriever	0.18	0.13	—	0.67	0.93	0.48	43	2.1	1
Llama 3.1 (8B)	0.15	0.13	0.47	0.55	**0.91**	0.41	53	1.6	1
→ int8 quantized	0.17	0.14	0.52	0.55	0.89	0.40	41	1.0	1
Llama 3.1 (70B)	0.16	0.11	0.55	0.61	0.67	0.53	64	9.5	2
→ int8 quantized	0.18	0.13	0.55	**0.66**	0.63	0.54	67	5.5	2

Table 3: Evaluation of generators stratified by whether a query is answerable with the knowledge base. Additionally to quality metrics, we report the length of response in tokens ($|\hat{y}|$) and processing time per query (sec/q). † = We were unable to obtain reliable timings for Gemma 3 1B due to a vLLM bug (v0.9.1, vllm/#19575).

tive observations and examples that reflect current limitations of the chatbot.

(1) Answers are tailored, but not proactive (Appendix, Figure 2). Some information-needs could be addressed more efficiently through proactivity. For example, instead of answering "Since only 50% your BSc courses are related to Data Science, you may not be admitted," the chatbot could prompt the user to provide specific course contents, to subsequently match them against admission requirements. However, these interactions require strong model reasoning and carry a high risk of ungrounded inference.

(2) Knowledge gaps (Appendix, Figure 3). We encountered several questions that the chatbot abstained from answering. This is caused either by an incomplete knowledge base, or because the website lacks this information. While frequent abstention limits the chatbot usefulness, it improves grounding, and has the beneficial side-effect of identifying ways to improve clarity and completeness of university websites.

(3) Contextualized queries (Appendix, Figure 4). The chatbot struggles with queries contextualized in multi-turn dialog. For example, follow-up questions often lack information of previous queries, causing the retriever to surface irrelevant documents. An adjacent problem are underspecified queries. For example, many user questions do not explicitly call out that the query is about the data science program (e.g., "When is the deadline for application?"), which is a consequence of users entering the chatbot through program-specific pages. Practically, this degrades retrieval quality and can lead to unfocused generations when the retrieved context provides multiple valid or even conflicting answers. A potential solution could be query expansion via decontextualization (Choi et al., 2021).

(4) Link-seeking queries (Appendix, Figure 5). While source links are shown alongside answers, the chatbot cannot embed them within responses as they are usually absent from source documents. To enable link-seeking interactions, it may be necessary to embed links in the documents and to instruct the generator to cite them.

(5) Little uptake of feedback mechanisms. We observed little user engagement with the feedback mechanisms. The message-level rating (thumbs-up/down) was used only once, and only 11 conversations received a Likert-scale rating. This highlights a general challenge in obtaining direct user feedback. Downstream evaluation may therefore need to focus on answer quality. In addition, latent usage signals such whether users ask followup questions or whether they return to the chatbot could be taken into account.

5 Related Work

RAG (Lewis et al., 2020) is becoming a widely adopted strategy across many domains in conversational search (Zamani et al., 2023). In the educational domain, Nguyen et al. (2021) deployed a chatbot to support admissions using the dialog management framework Rasa (Bocklisch et al., 2017). Significant engagement demonstrates high student demand. Odede and Frommholz (2024) conducted a user study comparing live-chat with university staff to a RAG-chatbot. Participants valued fast responses and low-entry barriers of the chatbot.

Cherumanal et al. (2024) find that intent-based retrieval can be an effective alternative to document retrieval, which supports the overall idea of our FAQ retriever. Closely related, Nguyen and Quan (2025) proposed a two-tiered strategy: if a query is sufficiently similar to a pre-defined FAQ, the system returns a pre-defined answer; otherwise, it falls back to document retrieval. Our system differs in two ways. First, instead of pre-defining answers, we link FAQs to knowledge base documents to ensure grounded and up-to-date answers. Second, instead of imposing a minimum similarity between queries and FAQ entries, we use the similarity as a soft ranking signal combined with other retriever scores which avoids an additional hyperparameter.

In the general domain, document assistants such as TruthReader (Li et al., 2024) and Verba (Weaviate, 2025) follow a similar RAG architecture as our system, but assume user-provided documents. The RAG-framework Onyx (2025) provides rich utilities to connect to heterogeneous data sources, but allows little customization of the retrieval strategy.

Despite these advances, there remains a gap in open-source chatbots designed for deployment in resource-constrained university settings, which our paper aims to address.

6 Conclusion

We present *Marcel*, an open-source RAG-based chatbot designed for handling enrollment-related student questions. The design and architecture targets resource-constrained university settings, taking into account privacy, controllability and observability. We analyze quality-efficiency trade-offs of common RAG-strategies, and provide insights from a real-world deployment at our institution. We plan to improve answer quality through query expansion and to integrate mechanisms for online evaluation of different answering-strategies.

Limitations

We note two limitations of our study. First, we considered only one prompt for the answer generator. It is conceivable that different prompting strategies could be used to improve performance, in particular for deciding when to abstain from answering a query. Second, we have only reported an intrinsic evaluation of the chatbot components. A longer-term evaluation is necessary to see if the system improves information access for students. As mentioned in the introduction, we hope that improved information access leads to a reduction in the number of unsuccessful applications due to formal criteria, and subsequently reduces the workload of academic staff. Understanding whether the chatbot will help to reach those goals requires a long-term evaluation and additional user uptake.

Ethical Considerations

It is important to note that university admissions is a high-risk domain, where wrong chatbot responses may adversarially affect users. At the current stage of the technology, it cannot be guaranteed that the chatbot provides correct answers due to the inherently stochastic nature of LLMs. Therefore, the chatbot only serves as an *additional* information source next to existing student support structures in form of the website, FAQs, email, and phone. To mitigate the risks associated with wrong responses, we (i) insert prominent disclaimers in the user interface, (ii) encourage users to verify important information through the display of relevant sources, and (iii) provide an admin interface to allow periodic review of conversations. A useful extension of this would be automatic flagging of potentially inaccurate responses so that they can be reviewed more efficiently. Lastly, all users in our study gave their explicit consent for the data collection and can retroactively opt-out.

References

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *Preprint*, arXiv:1712.05181.

Erin Brereton. 2021. Higher education's increasingly nuanced, AI-powered chatbots. Accessed: 2025-07-04.

Sachin Pathiyan Cherumanal, Lin Tian, Futoon M. Abushaqra, Angel Felipe Magnossão de Paula,

Kaixin Ji, Halil Ali, Danula Hettiachchi, Johanne R. Trippas, Falk Scholer, and Damiano Spina. 2024. Walert: Putting conversational information seeking knowledge into action by building and evaluating a large language model-powered chatbot. In *Proceedings of the 2024 ACM SIGIR Conference on Human Information Interaction and Retrieval, CHIIR 2024*, pages 401–405.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Forty-first International Conference on Machine Learning*.

Eunsol Choi, Jennimaria Palomaki, Matthew Lamm, Tom Kwiatkowski, Dipanjan Das, and Michael Collins. 2021. Decontextualization: Making sentences stand-alone. *Transactions of the Association for Computational Linguistics*, 9:447–461.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158.

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, and 196 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Dongfang Li, Xinshuo Hu, Zetian Sun, Baotian Hu, Shaolin Ye, Zifei Shan, Qian Chen, and Min Zhang. 2024. TruthReader: Towards trustworthy document assistant chatbot with reliable attribution. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 89–100.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Fengran Mo, Kelong Mao, Ziliang Zhao, Hongjin Qian, Haonan Chen, Yiruo Cheng, Xiaoxi Li, Yutao Zhu, Zhicheng Dou, and Jian-Yun Nie. 2025. A survey of conversational search. *Preprint*, arXiv:2410.15576.

Long S. T. Nguyen and Tho T. Quan. 2025. URAG: Implementing a unified hybrid RAG for precise answers in university admission chatbots – a case study at HCMUT. In *Information and Communication Technology*, pages 82–93.

Trung Thanh Nguyen, Anh Duc Le, Ha Thanh Hoang, and Tuan Nguyen. 2021. NEU-chatbot: Chatbot for admission of national economics university. *Computers and Education: Artificial Intelligence*, 2:100036.

Julius Odede and Ingo Frommholz. 2024. Jaybot - aiding university students and admission with an LLM-based chatbot. In *Proceedings of the 2024 ACM SIGIR Conference on Human Information Interaction and Retrieval, CHIIR 2024*, pages 391–395.

Onyx. 2025. Onyx: Open source Gen-AI + enterprise search. Accessed: 2025-07-04.

Cheat Sheet Series Team OWASP. 2025. Docker security cheat sheet. Accessed: 2025-07-04.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. 2024. LLM evaluators recognize and favor their own generations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xiangyu Peng, Prafulla Kumar Choubey, Caiming Xiong, and Chien-Sheng Wu. 2025. Unanswerability evaluation for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8452–8472.

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.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Aamir Shakir, Darius Koenig, Julius Lipp, and Sean Lee. 2024. Boost your search with the crispy mixedbread rerank models.

Weavitate. 2025. Verba: The golden RAGtriever. Accessed: 2025-07-04.

Hamed Zamani, Johanne R. Trippas, Jeff Dalton, and Filip Radlinski. 2023. Conversational information seeking. *Foundations and Trends® in Information Retrieval*, 17(3–4):244–456.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

A Benchmarking Rerankers

Retriever	MRR	ΔMRR	R@1	ΔR@1	R@5	ΔR@5	sec/query	Δsec/query	Params
BM25 + Dense (MARCO)	0.40	—	0.21	—	**0.50**	—	**0.06**	—	—
+ Rerank (`ms-marco-MiniLM-L12-v2`)	0.23	-42%	0.08	-62%	0.35	-30%	0.19	217%	33.4M
+ Rerank (`jina-reranker-v1-turbo-en`)	0.26	-35%	0.10	-52%	0.40	-20%	0.17	183%	37.8M
+ Rerank (`ms-marco-MiniLM-L6-v2`)	0.27	-32%	0.11	-48%	0.36	-28%	0.14	**133%**	22.7M
+ Rerank (`jina-reranker-v1-tiny-en`)	0.27	-32%	0.12	-43%	0.37	-26%	0.14	**133%**	33M
+ Rerank (`mxbai-rerank-large-v1`)	0.37	-8%	0.20	-5%	0.45	-10%	1.59	2550%	435M
+ Rerank (`mxbai-rerank-xsmall-v1`)	0.42	5%	0.30	43%	0.46	-8%	0.30	400%	70.8M
+ Rerank (`mxbai-rerank-base-v1`)	**0.46**	**15%**	**0.33**	**57%**	0.48	**-4%**	0.59	883%	184M

Table 4: Evaluation of second-stage rerankers. Each metric score is given with its relative delta (Δ) over the baseline. Inference time per query (sec/query) is measured on a GPU.

B LLM Prompts

Listing 1: Prompt for question answering based on retrieved documents.

```
// System prompt
You are a helpful and engaging chatbot called Marcel. If someone asks you, your name is Marcel
and you are employed at the Marburg University. You answer questions of students around their
studies. Please answer the questions based on the provided documents only. Ignore your own
knowledge. Don't say that you are looking at a set of documents. If you cannot find the answer
to a given question in the documents you must apologize and say that you don't have any
information about the topic (e.g., "Unfortunately, I do not have any knowledge about <rephrase
the question>").

// (Optional) Conversation history (turns of user/assistant messages)
...

// User prompt for question answering
Given these documents, answer the question.

## Documents
{% for doc in documents %}
### {{ doc.title }}
{{ doc.content | replace("\n", "\\\\n") }}

{% endfor %}

## Question
{{ question }}
```

Listing 2: Prompt for question answering without retrieval.

```
You are a helpful and engaging chatbot called Marcel. Your name is Marcel and you are employed
at the Marburg University. You help students with questions regarding admission and enrolment
to the MSc Data Science Program. Only engage in topics related to admission and enrolment. Do
not engage in other discussions.
```

Listing 3: Prompt to classify a query into retrieval or no-retrieval.

```
Please determine if the following user utterance is a question. Respond with 'YES' if it is a
genuine question. Respond with 'NO' if it is chit-chat or if the user asks the chatbot what
kind of information it could provide, unrelated to earlier conversation. Only respond with the
label YES or NO.

Message: {message}
```

Listing 4: Prompt to detect non-answers for evaluation of selectivity.

```
## Instruction
Classify the given response as either an answer or a non-answer.

Non-answers are responses that indicate a lack of information or inability to provide an
answer. They typically include phrases such as:
- "I don't know..."
- "I can't provide information about..."
- "It is not stated..."
- "There is no answer"
- "There is no information..."
- Or other synonymous expressions indicating uncertainty or lack of information

Any response that provides concrete information or a direct answer, even if brief, should be
classified as an answer. For each response, provide a classification in the following format:
Classification: {'rationale': 'Brief explanation of why it's an answer or non-answer',
'non_answer': 0 or 1}

Use 0 for answers and 1 for non-answers in the 'non_answer' field.

## Examples
Response: The information provided does not mention anything about compulsory elective modules
or the number of free modules a student can choose from in the M.Sc. Data Science program at
Marburg University.
Classification: {'rationale': 'The response indicates a lack of information, which is
characteristic of a non-answer.', 'non_answer': 1}

Response: The question is not clear because there is no specific module mentioned. The text
only talks about enrollment as a doctoral student and provides information about the required
documents and the enrollment process. It does not mention a specific module.
Classification: {'rationale': 'The response states that specific information is not provided,
making it a non-answer.', 'non_answer': 1}

Response: Master of Science (M.Sc.)
Classification: {'rationale': 'The response provides a specific piece of information, making
it a valid answer.', 'non_answer': 0}

Now, classify the following response:
Response: {{ response }}
Classification:
```

C Example FAQ Data

#	FAQ	Source
1	Which Bachelor degree is needed to be admitted to the Masters of Data Science program?	(D_1) Master Data Science Admission Requirements; (D_4) Master Data Science Eligibility Assessment Process
2	Which language certificates are accepted?	(D_1) Master Data Science Admission Requirements; (D_5) Demonstrating English proficiency (C1); (D_6) Demonstrating English proficiency (B2); (D_7) Demonstrating German proficiency
3	What are the application deadlines?	(D_8) Application Deadlines for Master Programs

Table 5: Example FAQ data. Each question is linked to answer-relevant documents in the knowledge base.

D Example Evaluation Data

#	Question	Reference Answer	Source
1	I am writing to ask whether me a Aerospace Engineering graduate can apply for the Master in Data Science? If yes will there be additional requirements?	Your bachelor degree needs to be in Data Science, Mathematics or Computer Science or comparable. Overall, at least 90 credit points (CP) must have been completed in in academic courses on Mathematics and Computer Science. Note that there is a list of specific courses that are required, and you need to cross-check whether you fulfil the requirements on those, too. Industry training or professional experience does not count as academic education.	(D_1) Master Data Science Admission Requirements
2	I have 2 questions: Is German language A2 required during online application? If I don't have proof of German language A2 level, Is it possible to be accepted?	You need to prove either English level C1, or German DSH-2 and English B1. Proof of language proficiency needs to be submitted with the application.	(D_1) Master Data Science Admission Requirements; (D_2) Application with a degree from abroad (first semester)
3	I aspire to take admission at your esteemed university for masters in data science for second semester in summer as I am pursuing masters in data science for first semester at a university in Germany currently for the winter semester. I wanted to know how to apply for second semester and the deadline for it.	In short, you need to obtain an advanced placement certificate from the departmental advisor of the Masters Data Science programme. You then and apply via uniassist, indicating an application for the second semester. The process is explained in detail on the university website.	(D_3) Application with a degree from abroad (higher semester)
4	Can my application be checked before upload to see whether I am eligible. I do not want to pay the application fee if I would not be admitted to the program.	The admission committee cannot do any preapplication checks. You can check your transcript of records yourself and see whether it matches the admission criteria.	n/a
5	I am now in the process of preparing my papers for submission, but due to some circumstances, it may be a little delayed. In the end, everything will be ready before Deadline on February 23. My question is: Does delaying the application affect the chance of acceptance? Does the acceptance stage take place during the application period or does it not take place until after the deadline has passed for all applicants?	Applications are checked independent of each other, so your chance of admission does not depend on the time you submitted the documents. It is only checked whether the admission requirements are fulfilled. Offer letters will be sent as soon as the application is checked.	n/a
6	If I am unable to obtain my visa before the semester begins, would it be possible to defer my admission to the next semester master data science?	You can transfer your admission to the next semester one time if the following conditions are met: (1.) You are not yet enrolled. (2) The study program is offered in the following semester (which will be for Master Data Science). (3) Admission Conditions in the Study Regulations have not changed (not planned for Master Data Science). You should then write an email to masters@uni-marburg.de in December / July and request the transfer. Then you will get a new letter of admission (after the check whether the above conditions are met).	n/a

Table 6: Sample questions from evaluation data. Each question has a manually written reference answer. Questions that are answerable with the knowledge base (#1–3) also have ground-truth sources to evaluate retrieval. Unanswerable questions (#4–6) are used to evaluate generator's ability to abstain.

E Example Generations Reflecting Current Limitations

User: I have an industrial engineering bachelor degree from iran butmy requirement courses are 50% rekated to requirement of data science program can I be elighable?

Marcel: According to the Data Science (Master of Science) document, it is stated that "We require proof that you have completed a relevant Bachelor's degree in the fields of Data Science, Mathematics or Informatics or proof of a comparable domestic or foreign university degree qualifying you for a profession."

Additionally, it is required that "at least 72 credit points (CP) must have been earned in modules from the scientific discipline of Computer Science and at least 18 CP in modules from the scientific discipline of Mathematics."

However, it is not explicitly stated that the entire undergraduate degree program must be directly related to Data Science. Nevertheless, it is stated that "It is strongly recommended that students take responsibility for acquiring the skills from the above modules that they have not already acquired before commencing their studies."

Since you have mentioned that only 50% of your requirement courses are related to the requirements of the Data Science program, it might be possible that you are eligible, but you would likely need to demonstrate that you have acquired the necessary skills in the areas where your undergraduate degree program was lacking.

You can contact Prof. Dr. Christin Seifert for content-related questions about the program, and Master's Coordination for formal questions about your application. Email: christin.seifert@uni-marburg.de Email: master@uni-marburg.de

User: about elighbility assesment dose operations research is related to 'Basic Real Analysis' or 'Analysis I'

Marcel: Unfortunately, I do not have any information about Operations Research being directly related to 'Basic Real Analysis' or 'Analysis I' in the context of the Data Science program at Marburg University in the provided documents.

User: so what is fundumentals of basic real analysis

Marcel: Based on the provided documents, the content of "Basic Real Analysis" is not explicitly stated. However, "Analysis I" and "Analysis II" are mentioned as alternatives to "Basic Real Analysis" in the prerequisites for some modules.

In the module "Stochastical Analysis", it is mentioned that either "Analysis I and Analysis II" or "Basic Real Analysis" is recommended. This suggests that "Basic Real Analysis" may cover similar topics as "Analysis I and Analysis II", but with a focus on real analysis.

In general, a course on real analysis typically covers topics such as:

- Real numbers and their properties
- Sequences and series of real numbers
- Continuity and differentiability of real-valued functions
- Integration of real-valued functions
- Metric spaces and topology

However, without more specific information, it is difficult to say exactly what topics are covered in the module "Basic Real Analysis" at Marburg University.

User: what about Efficient Algorithms

Marcel: Efficient Algorithms is a module that is part of the Data Science program at Marburg University. It is mentioned as one of the recommended prerequisites for the Parameterized Algorithms module, and it is also mentioned as an example of a module that students may take as part of their studies in the Perspectives document.

According to the Online-Modulhandbuch, the "Efficient Algorithms" module (CS 566) is an advanced module that covers topics such as algorithmic techniques, greedy algorithms, dynamic programming, and space and time complexity. The module has a workload of 270 hours and is worth 9 credit points.

The qualification goals of the "Efficient Algorithms" module include:

- Applying skills in designing algorithms and knowledge of key design and analysis paradigms
- Using efficient data structures in algorithm design
- Analyzing algorithms with respect to correctness and effort
- Practicing scientific working methods (recognizing, formulating, solving problems, abstraction)
- Training to speak freely about scientific content

There are no explicit prerequisites for the "Efficient Algorithms" module, but the following modules are recommended:

- Algorithms and Data Structures
- Practical Informatics II: Data Structures and Algorithms for Pre-Service-Teachers

The module is assigned to the Computer Science department and can be attended by students in various study programs, including B.Sc. Data Science, B.Sc. Computer Science, M.Sc. Data Science, M.Sc. Computer Science, and others.

Figure 2: Example where a user wants to determine if their academic background meets admission requirements.

User: For Data Science can I submit my IELTS after deadline?

Marcel: Unfortunately, I do not have any information about the possibility to submit IELTS scores after the deadline for the Data Science Master's program.

User: Can I submit my APS after deadline?

Marcel: Unfortunately, I do not have any information about the possibility to submit APS scores after the deadline for the Data Science Master's program

Figure 3: Examples where the chatbot abstains from answering due to knowledge gaps.

User: Is a Thesis required for admission to msc data science?

Marcel: No, a thesis is not a requirement for admission to the Data Science Master's program. However, if a final thesis worth at least 10 credit points has already been completed in the fundamental university degree program, proof of this final thesis and its scope must be submitted as part of the eligibility assessment process.

User: Is english required

Marcel: Yes, English proficiency of at least level B1 of the "Common European Framework of Reference for Languages" or English proficiency of at least level C1 is required, depending on the specific circumstances.

User: What are the circumstances

Marcel: It appears that the circumstances refer to the situation in which initial applicants can move into an apartment straight away. According to the text, only under individual circumstances can initial applicants move into an apartment straight away, and in such cases, an early application is strongly advised.

Figure 4: Example of a contextualization issue. The third query is a follow-up on the second query. Omission of the subject ("circumstances when B1 or C1 is required") lead to retrieval/generation of unrelated information.

User: Hi Marcel, what to study at uni marburg?

Marcel: You can find explanations of our degrees in the "Studying in Marburg" section, which may help you get a first impression of life in Marburg and the variety of subjects we offer.

User: Can you show me a link to that?

Marcel: As a chatbot, I don't have the capability to share links. However, you can visit the University of Marburg's official website, and from there, you can navigate to the "Studying in Marburg" or the "Degree programmes" section. This will give you more information about the various courses they offer, including the MSc Data Science program that I can help you with.

Would you like to know more about the MSc Data Science program specifically?

User: Why should I study Msc. DataScience?

Marcel: Unfortunately, I do not have any information about why you should study M.Sc. Data Science specifically, but you can find an overview of the program on the M.Sc. Data Science web page, which includes the required study skills and interests, as well as the content and structure of the program.

Figure 5: Example of a link-seeking question. The generator cannot embed links to sources in the main text (second answer), unless some retrieved context happens to mention the link (third answer).

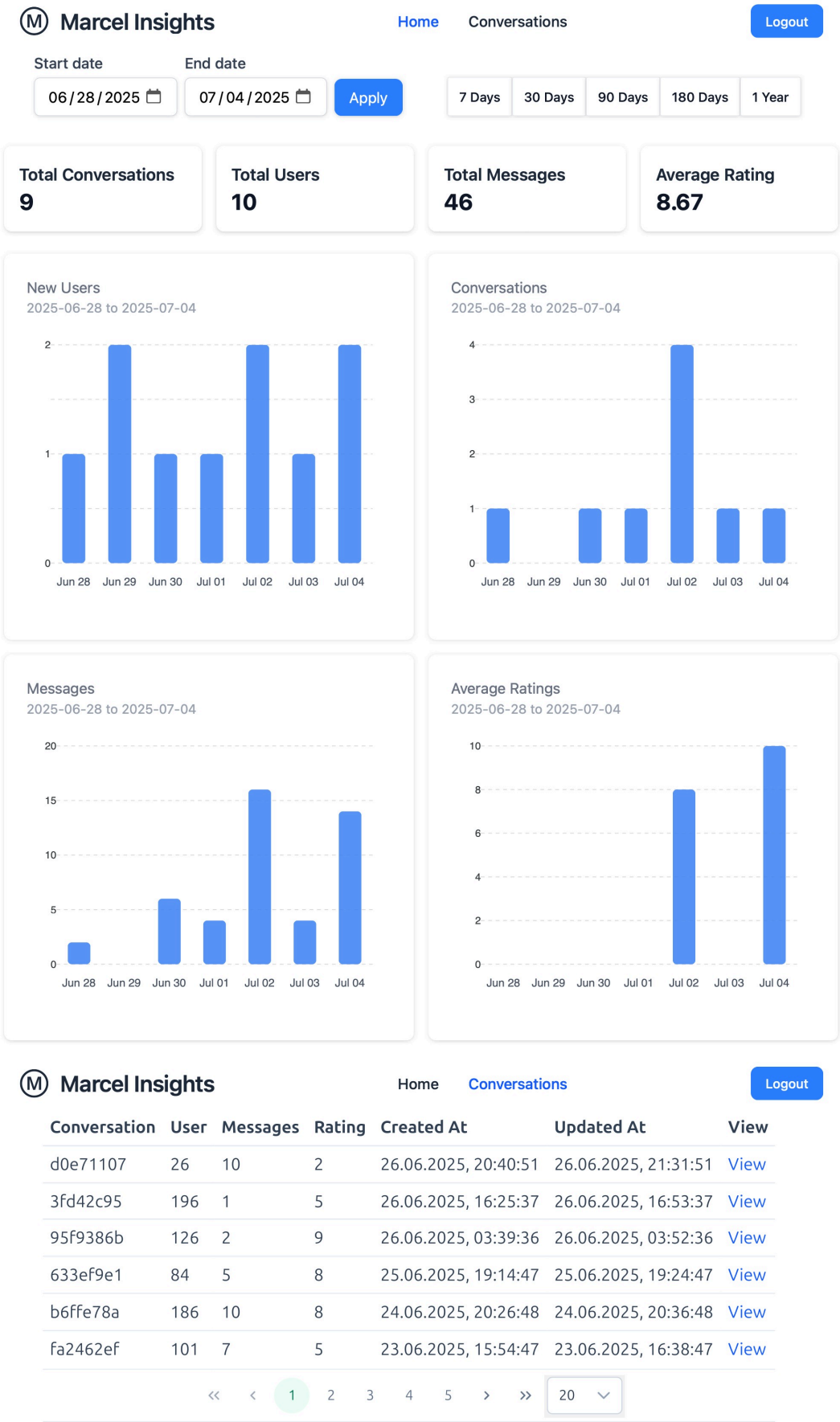


Figure 6: Overview of admin-facing UI which is only accessible to authorized users. Top: A dashboard shows key usage statistics over a customizable time period. Bottom: List of conversations with links for manual review.

Alpha-GPT: Human-AI Interactive Alpha Mining for Quantitative Investment

Saizhuo Wang[1*], **Hang Yuan**[1*], **Leon Zhou**[3], **Lionel M. Ni**[1,4], **Heung-Yeung Shum**[1,2], **Jian Guo**[2]
[1] HKUST, [2] IDEA Research, [3] Columbia University, [4] HKUST-GZ
{swangeh, hyuanak}@connect.ust.hk, leon.zhou@columbia.edu, ni@ust.hk, {hshum,guojian}@idea.edu.cn
[*]Equal contribution

Abstract

One of the most important tasks in quantitative investment research is mining new alphas (effective trading signals or factors). Traditional alpha mining methods, either hand-crafted factor synthesis or algorithmic factor mining (e.g., search with genetic programming), have inherent limitations, especially in implementing the ideas of quant researchers. In this work, we propose a new alpha mining paradigm by introducing human-AI interaction, and a novel prompt engineering algorithmic framework to implement this paradigm by leveraging the power of large language models. Moreover, we develop Alpha-GPT, a new interactive alpha mining system framework that provides a heuristic way to "understand" the ideas of quant researchers and outputs creative, insightful, and effective alphas. We demonstrate the effectiveness and advantage of Alpha-GPT via a number of alpha mining experiments. In particular, we evaluated Alpha-GPT's performance in the **WorldQuant International Quant Championship 2024**, where it demonstrated results comparable to those of top-performing human participants, ranking among **top-10** over 41000 teams worldwide. These findings suggest Alpha-GPT's significant potential in generating highly effective alphas that may surpass human capabilities in quantitative investment strategies.

1 Introduction

A trading alpha (Tulchinsky, 2019) is a financial signal or a function with predictive power over excess return or risk, and they are usually expressed via symbolic rules or formulas (machine learning alphas are getting more popular recently but they are not discussed in this work). Alphas play a vital rule in trading economics, and most research work in quantitative investment focuses on how to find good alphas. See (Kakushadze, 2016) for a number of such formulaic alphas (e.g., $-\frac{close-open}{(high-low)+0.001}$ computes the increase from open price to close price relative to the intraday volatility, and the negative sign indicates a potential mean-reversion effect).

Traditionally, alpha mining has two paradigms (Figure 1). The first paradigm relies on manual modeling. Quant researchers attempt to translate their ideas/intuitions about financial markets into formulaic alphas, test their effectiveness and significance through backtest experiments, and analyze the reasons of success and failure. Usually, this process is repeated for many rounds to improve the performance of alphas. The success of this paradigm depends heavily on the talent and expertise of individuals and suffers from the problems of inefficiency and labor cost. On the other hand, the second paradigm seeks alphas through search algorithms such as genetic programming (Zhang et al., 2020). Since the search space, composed of all possible combinations for hundreds of operators and operands (features), is incredibly large, it is extremely compute-intensive to find satisfactory alphas during the alpha search process.

Both of these paradigms exhibit common shortcomings. Firstly, it is a difficult process to find a precise and concise formulaic expression that encapsulates one's ideas about trading signals or observed trading opportunities and patterns. Examples include the formulaic representation of technical analysis patterns such as Ascending Triangles (Lo et al., 2000) and Elliott Wave Theory (Elliott and Prechter, 2005), which exist but are hard to discover. Secondly, understanding and interpreting a large number of alphas selected by search algorithms is especially time-consuming and labor-intensive. Lastly, it is unreasonable to expect creative and effective alphas to come from the stroke-of-genius by researchers or the brute-force search by algorithms, but rather, it often comes from a repeated process of experimentation-and-analysis. However, designing and modifying the parameters and search configurations of algorithmic alpha min-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 196–206

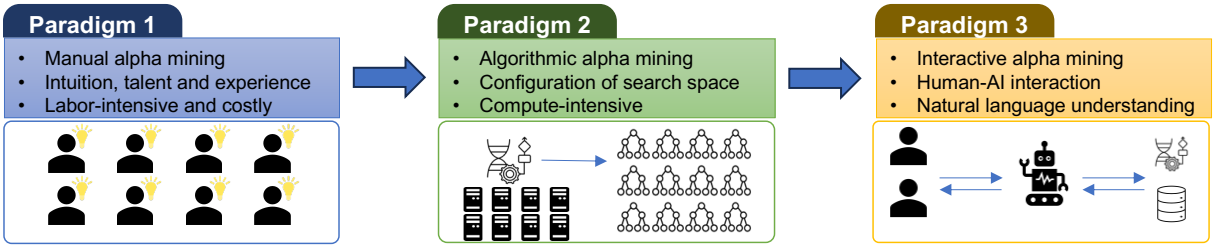


Figure 1: Evolution of alpha mining techniques.

ing systems is usually a menial task for researchers.

To address these challenges, we propose the third alpha mining paradigm which enhances human-AI interaction to improve the effectiveness and efficiency of alpha research. Based on this new paradigm, we propose the architecture of an interactive alpha mining system, termed **Alpha-GPT**. This system incorporates large language models (LLM) as a mediator between quantitative researchers and alpha search. Alpha-GPT has three key advantages. First, it can interpret users' trading ideas and translate them into fitting expressions, thanks to LLM's great natural language understanding and instruction-following capability. Secondly, Alpha-GPT can quickly understand, exploit and summarize top-performing alphas and meta-data via their natural/formal language expression, leveraging LLM's broad prior knowledge obtained via pretraining. Finally, the user can then suggest modifications to the alpha search, which the model will automatically make to future rounds of alpha mining, based on LLM's in-context learning and reasoning capabilities. This greatly simplifies the workflow (Figure 5) and allows the user to approach alpha mining from a high-level standpoint (in terms of abstract ideas).

Our contributions in this work can be summarized from these standpoints: (1) We define a new paradigm for alpha mining utilizing human-AI interaction to improve the effectiveness of alpha research. (2) We propose AlphaBot, an algorithm with domain knowledge compilation and decompilation methods to employ the LLM as a mediator for human-AI interaction. (3) We develop Alpha-GPT, a system to realize our proposed paradigm and a tool for quantitative researchers.

2 Agentic Workflow

Taking inspiration from the established process of human quantitative researchers, Alpha-GPT employs an agentic workflow to generate and refine trading alphas. As illustrated in Figure 2, this workflow is structured as an iterative process comprising three distinct stages: ideation, implementation, and review.

2.1 Ideation

The workflow is initiated in the **ideation** stage, wherein a quantitative researcher articulates a trading idea or market intuition using natural language. The principal agent in this stage is *Trading Idea Polisher*, whose primary goal is to formalize the researcher's nascent idea into a structured prompt suitable for machine processing. To accomplish this, the agent queries a **Database** containing a corpus of literature and detailed specifications of available data fields. By leveraging this external knowledge base, the *Trading Idea Polisher* augments the original query, disambiguates financial terminology, and incorporates contextual examples to ensure the precise capture of the user's intent.

2.2 Implementation

During the **implementation** stage, the refined idea from the preceding stage is operationalized into executable alpha expressions. The *Quant Developer* agent, which leverages a Large Language Model (LLM), processes the structured prompt to generate a set of initial "seed" alpha expressions. These mathematical formulations are intended to embody the specified trading concept and are cataloged in the **Alpha Database**. Following this, the **Alpha Compute Framework** employs algorithmic search enhancement methods, notably genetic programming, to iteratively evolve and improve this initial set of alphas. This process yields a more diverse and sophisticated population of candidate alphas optimized for performance.

2.3 Review

In the terminal stage, **review**, the candidate alphas undergo rigorous empirical evaluation. The *Analyst* agent coordinates this process, utilizing the

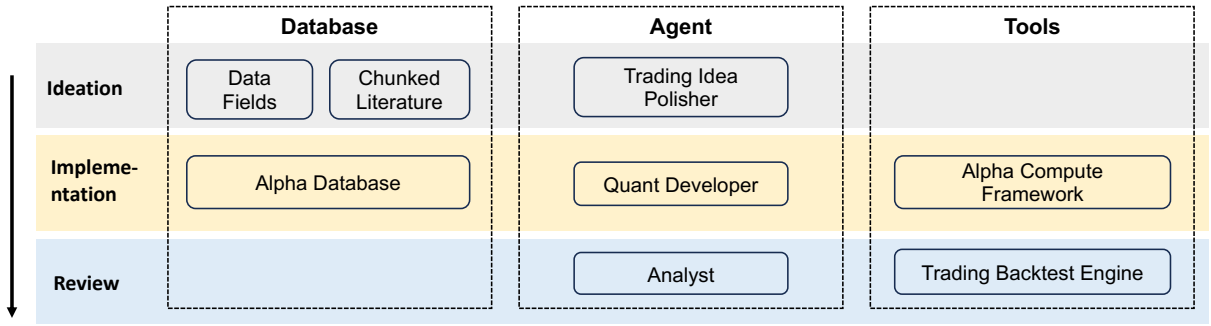


Figure 2: The agentic workflow of Alpha-GPT

Trading Backtest Engine as its primary analytical tool. This engine executes historical simulations to assess alpha performance against market data, generating quantitative metrics that include backtest returns, Information Coefficient (IC), and Sharpe ratio. The *Analyst* agent then synthesizes these outputs, providing natural language summaries and interpretations of the top-performing alphas to the researcher. This interactive feedback loop enables the researcher to provide further direction for subsequent rounds of alpha mining, fostering a collaborative human-AI discovery process.

3 Modes of Operation

As a practical assistant tool for quantitative research, Alpha-GPT is designed to operate in two distinct modes: *interactive mode* and *autonomous mode*. In interactive mode, the system functions as a collaborative partner, where human researchers provide input and guidance throughout the agentic workflow. This approach is predicated on the recognition that human domain expertise and intuition in trading and investment often surpass the current capabilities of LLMs. In contrast, the autonomous mode enables the system to generate and iterate upon trading ideas independently. This mode is particularly useful when faced with exceptionally large quantitative databases, where it can perform a rapid and reliable bootstrap of satisfactory alphas that human researchers can subsequently analyze and develop further.

3.1 Interactive Mode

In the interactive mode (an example pipeline shown in Figure 5), Alpha-GPT serves as an intelligent interface that bridges the gap between a researcher's conceptual ideas and their empirical validation. The human researcher remains central to the discovery process, initiating the workflow by providing trading ideas in natural language and offering feedback at the review stage of each iteration. In this collaborative paradigm, Alpha-GPT acts as a co-pilot responsible for translating these abstract concepts into precise, formulaic alpha expressions. It then manages the computationally intensive tasks of alpha enhancement using methods like genetic programming and executes rigorous backtesting for performance evaluation. Finally, the system synthesizes the complex results into comprehensible natural language summaries, facilitating human review and decision-making to guide the next cycle. This synergy between human intuition and the system's advanced computational capabilities serves to accelerate the research cycle.

3.2 Autonomous Mode

The autonomous mode is engineered for the systematic exploration of large-scale quantitative databases, which can contain tens of thousands of data fields. In such scenarios, providing the complete documentation of all available data to an LLM would overwhelm its context window, both in terms of token limits and information density. To surmount this challenge, Alpha-GPT employs a hierarchical Retrieval-Augmented Generation (RAG) strategy, as depicted in Figure 3. This strategy enables the LLM agent to autonomously discover novel trading ideas by navigating the database in a structured, top-down manner.

The process commences with the LLM agent analyzing the existing **Alpha Database** to learn the characteristics of previously successful alphas (RAG#0). Guided by this initial analysis, the agent then queries the **High-level Categories** of the full database, such as 'Price-Volume' or 'Sentiment', to identify broad, promising domains for new alpha discovery without retrieving excessive detail (RAG#1). Following this, the agent performs a more focused query on the corresponding **Second-**

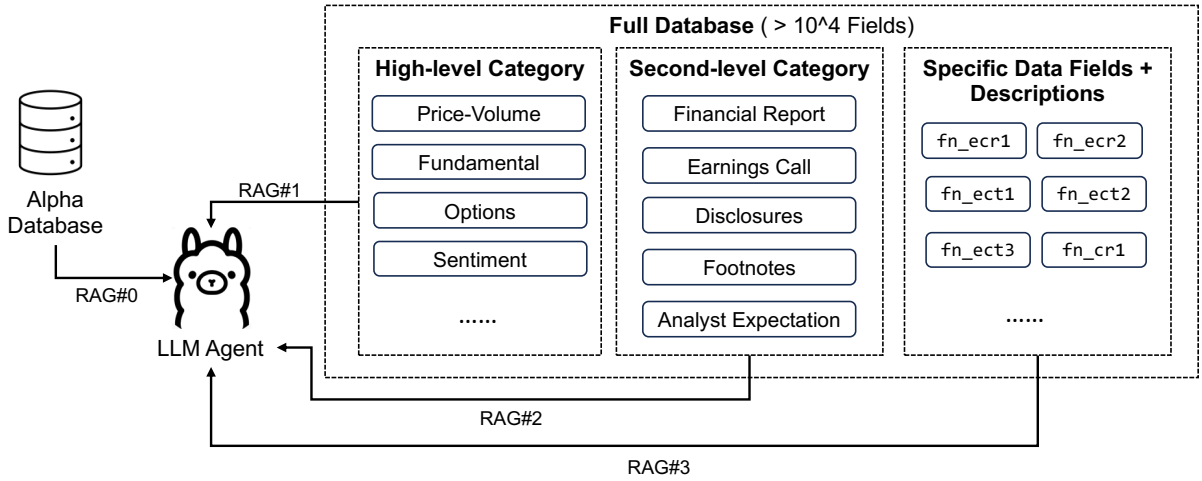


Figure 3: Alpha-GPT's hierarhical RAG in autonomous mode for large-scale quant database.

level Categories, like 'Earnings Call', to progressively narrow the search space (RAG#2). In the final step, the agent retrieves the detailed descriptions for **Specific Data Fields** within the chosen sub-category, and armed with this granular information, it can formulate a novel, concrete trading idea and generate the associated alpha expression (RAG#3). This hierarchical framework allows Alpha-GPT to methodically explore a vast and complex feature space, effectively managing context size while continuously generating novel ideas.

4 System Architecture

The overall system architecture of Alpha-GPT is illustrated in Figure 6. It is a multi-layered framework composed of a user-facing interface, a core LLM agent, an algorithmic mining engine, and a computation acceleration layer.

4.1 WebUI and LLM Agent

The top layers of the architecture facilitate human-AI interaction. The Web-based User Interface (WebUI) is the primary entry point for a quantitative researcher. It includes a **Dialog Box** for natural language interaction, a **Mining Session Manager** to organize distinct research threads, and an **Alpha Mining Dashboard** for comprehensive visualization of experiments and performance analytics. The LLM Agent, termed as the **AlphaBot** layer, serves as the core intelligence of the system. It employs a standard prompt engineering pipeline to translate user intent into structured tasks. This process leverages Retrieval-Augmented Generation (RAG) over a vector database of financial literature and historical alphas to ground the model's outputs. The agent's responses are then processed through a structured output parsing and validation module to ensure the generation of syntactically correct and semantically valid alpha expressions for the backend systems.

4.2 Backend Systems

Algorithmic Alpha Mining This layer serves the search enhancement function in Alpha-GPT. It implements an algorithmic workflow by taking the seed alphas generated by AlphaBot and iteratively improving them based on received search commands and configurations. The layer consists of four modules. The **Alpha Search Enhancement** module uses techniques like genetic programming to generate a diverse set of alpha candidates. Qualified alphas are then filtered by the **Evaluation and Backtesting** module, which assesses performance against historical data. These alphas are further pruned and scored by the **Alpha Selection** module to remove redundancies and identify the most valuable signals. Finally, the **Alpha Deployment** module prepares the finished alphas for live trading, ensuring the smoothness and correctness of real-time computation.

Alpha Computation Acceleration Alpha computation requires processing vast amounts of financial data, and the computational overhead of handling high-frequency data makes acceleration a key requirement. The alpha computation acceleration layer employs several key techniques to meet these demands, including the use of streaming algorithms for rolling window computations, vectorized computation to leverage hardware concurrency, SIMD/SIMT instructions for parallel

Table 1: Operators used in the experiment

Type	Operators
time-series	shift,ts_corr,ts_cov, ts_decayed_linear, ts_min, ts_max, ts_argmax, ts_argmin, ts_argmaxmin_diff, ts_max_diff, ts_min_diff, ts_mean, ts_median, ts_zscore_scale, ts_maxmin_scale, ts_skew, ts_kurt, ts_delta.ts_delta_ratio, ts_ir, ts_decayed_linear, ts_ema, ts_percentile, ts_linear_reg, ts_rank, ts_sum, ts_product, ts_std,
cross-sectional	zscore_scale, winsorize_scale, normed_rank,cwise_max, cwise_min
group-wise	grouped_demean, grouped_max, grouped_min, grouped_sum, grouped_mean, grouped_std, grouped_zscore_scale, grouped_winsorize_scale,
element-wise	relu, neg, abs, log, sign, pow,pow_sign, round, add, minus, cwise_mul, div,greater,less, normed_rank_diff

data processing, memory optimization techniques like pre-allocation, and GPU acceleration for data-intensive tasks.

5 Evaluations

In order to assess the impact of Alpha-GPT on enhancing researchers' productivity in identifying relevant factors, we carry out a combination of quantitative and qualitative studies. The quantitative experiments aim to validate the effectiveness of Alpha-GPT by evaluating its performance based on given sets of trading ideas or databases, while the qualitative experiments (Section 5.5) aim to showcase successful instances of its application. The results below are intended to verify the following questions: **(1)** Can Alpha-GPT improve quant research efficiency via human-AI interaction? **(2)** Can the algorithmic search enhancement module improve the quality of generated alpha? **(3)** Can Alpha-GPT ultimately lead to better alphas?

5.1 Experimental Setup

Without further specifications, the experiments below are conducted with the following setups.

Data and operators We use intraday volume-price data of Chinese and US stocks. The data include the basic candlestick chart data (OHLCV), volume-weighted average price (VWAP), and sector data. The operators we use include 19 basic operators implemented in (Guo et al., 2023) including time-series operations, cross-sectional operations, group-wise operations and basic element-wise operations, as shown in Table 1. Besides, we also incorporated operators from existing libraries such as scipy and torch.

Knowledge Library We construct the knowledge library based on the alphas proposed in (Kakushadze, 2016) and a proprietary alpha base. For each alpha, we first decompose it into sub-expressions and explain them. Then we explain the combination of these sub-expressions to form the whole trading idea. Document embeddings are indexed via Faiss[1]. Note that we only employed external memory when generating alphas for trading ideas that align well with those in the alpha base. Importantly, the knowledge library serves as an auxiliary resource to enhance interpretability and consistency, rather than as a source of direct alpha reuse. Alpha-GPT remains capable of producing novel alphas beyond the scope of the library, and our experiments confirm that a large portion of generated alphas are not present in either the literature or the proprietary base. The inclusion of these resources thus does not compromise novelty but instead provides grounding and domain context for the generation process.

LLM and Adapter We used Llama3 70B (Grattafiori et al., 2024) as the chat model and BGE-M3 (Chen et al., 2024) as the embedding model.

5.2 Efficiency Improvement

We evaluate Alpha-GPT's ability to improve research efficiency by assessing its effectiveness in translating trading ideas into alphas and its capacity to develop stronger alphas through iterative refinements.

Translation Consistency To verify Alpha-GPT's ability to enhance researchers' efficiency by providing accurate and high-quality factors, we conducted a comparative study. We collected generated alphas based on a trading idea dataset from both Alpha-GPT and a group of human quant researchers. The human group comprised five quant researchers with experience ranging from 0.5 to 2 years. The trading idea dataset was randomly split into five parts, with each human researcher tasked with writing alphas based on a specific split. For evaluation, we prompted GPT-4 to score the generated alphas on a scale of 1 to 10 (with 10 being the highest score) and select the superior one. The average results are presented in Table 3. The results show that the factors generated by Alpha-GPT consistently outperformed those produced by human researchers. This outcome strongly indicates Alpha-GPT's effectiveness in improving research efficiency by accurately translating trading ideas into high-quality factors. This experiment demon-

[1]https://github.com/facebookresearch/faiss

Table 2: WorldQuant International Quant Championship 2024 Stage 2 Results (by June 25th, 2024)

	Number of Qualified Alphas Generated	Total Score	In-sample Score	Out-of-sample Score
Worldwide Top-1	103	52058	57899	50111
Worldwide Top-10	47	47112	42303	48715
Regional Top-1	91	50920	55890	49264
Regional Top-10	74	35999	26292	39325
Alpha-GPT	81	48866	65505	43319

Table 3: Consistency comparison between a junior human researcher and Alpha-GPT

	Score	Win rate
Human	6.81	13.40%
Alpha-GPT	8.16	86.60%

Table 4: Alpha IC comparison between different stages. "Seed" means the initial alpha generated by Alpha-GPT. "SE" means 10 rounds of search enhancement on the initial alpha. "IT+SE" means after 1 round of interaction and then 10 rounds of search enhancement.

Alpha	Seed	SE	IT + SE
IC	0.58%	1.23%	2.23%

strates Alpha-GPT's potential to significantly enhance the productivity of quant research teams, particularly in the crucial task of transforming conceptual trading ideas into concrete, implementable factors.

Human-AI Iterative Refinement We also verify the effectiveness of Alpha-GPT in helping improve alpha research in through human-AI interaction. We first simulated a human user using another LLM (GPT-4) with specifically designed prompts. For each trading idea in the dataset, this simulated human user will send it to Alpha-GPT and interact with it for another round, based on the explanation generated by Alpha-GPT in the first round. Then, we evaluate the IC of the factors that are generated initially, after search enhancement, and after 1 round of interaction & search enhancement. The result is shown in Table 4, where consistent improvements in factor IC demonstrates the effectiveness of interaction.

5.3 Search Enhancement

To validate the effectiveness of the alpha mining layer in consistently enhancing factors, we analyzed the information coefficient (IC) of alphas generated through multiple rounds of search enhancement, both in-sample and out-of-sample. Figure 4 illustrates the IC curves over 20 iterations, revealing several key insights. Both in-sample and out-of-sample ICs show a sharp initial increase (iterations 0 to 5), indicating rapid improvement of initial factors. The in-sample IC (blue line) demonstrates a general upward trend throughout, suggesting continuous factor enhancement on training data. Notably, the out-of-sample IC (orange line) stabilizes after the initial rise, indicating that improvements generalize well to unseen data and mitigating overfitting concerns. Both curves appear to converge around the 15th iteration, suggesting an optimal stopping point for the enhancement process.

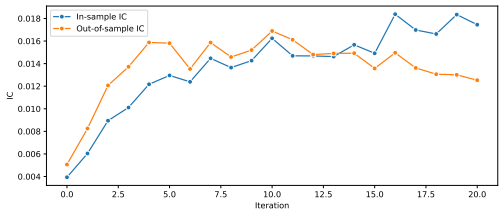


Figure 4: Search Enhancement curve

5.4 Stronger Alphas

To evaluate Alpha-GPT's ability to generate superior alphas and investment strategies, we designed an automated testing scenario simulating collaboration between human researchers and AI. We created a meta-database of operands (data fields) and operators with detailed descriptions. A specially prompted LLM was then used to systematically explore these fields and generate alphas with strong performances, simulating a human researcher interacting with Alpha-GPT to search for high-performing alphas. This process incorporates elements similar to traditional methods such as genetic programming, but with the search guided by the LLM.

High-frequency Trading Competition We evaluated Alpha-GPT in following the same evaluation

Table 5: Alpha-GPT's comparison with human-crafted factors

	Return	Sharpe	MDD
Top-1 (Human)	**21%**	**6.88**	1.61%
Top-5%	16%	5.42	**1.59%**
Top-10%	13%	4.16	3.58%
Alpha-GPT	14%	5.47	2.36%

protocol of a concluded alpha competition in high-frequency trading.[2] Specifically, we incorporated self-improving mechanism (Wang et al., 2024) to generate factors and compared the result with human leaderboards, as shown in Table 5. It can be seen that Alpha-GPT achieved Top 5%-10% performance of human participants. Notably, the initial competition duration was one month, but Alpha-GPT was able to reach a comparable performance level in just one day.

WorldQuant International Quant Competition For a more practical and challenging scenario, we deployed our automation to the WorldQuant International Quant Championship (IQC) 2024 [3], the premier competition in formulaic alpha mining that involves more than 41,000 participants from over 100 countries and 5,000 universities. The competition offers a vast exploration space with over 5,000 operand data fields spanning price-volume, fundamentals, derivatives, news sentiment, and more, along with over 100 operators of various types. The platform applies strict criteria for alpha qualification, considering factors such as alpha return, turnover, and Sharpe ratio. Importantly, our evaluation was conducted in real time during the official competition period, ensuring that no future information leakage occurred. As presented in Table 2, the results demonstrate that our automated Alpha-GPT system can generate performant alphas, ranking among the top 10 worldwide and top 3 regionally. In particular, Alpha-GPT produces a comparable number of qualified alphas to top human competitors and achieves the highest in-sample score. The system's out-of-sample score is also highly competitive, indicating that alphas generated based on the LLM's prior knowledge generalize well and possess strong underlying logic. These impressive results underscore Alpha-GPT's potential to achieve superhuman performance in alpha mining.

[2]Joinquant alpha competition **link**

[3]WorldQuant IQC **link here**.

5.5 Qualitative Results

Idea-Formula Consistency We demonstrate that Alpha-GPT can generate formulaic alphas that are consistent with the user's given trading idea. Figure 7 illustrates the generated alpha expressions based on given trading ideas and their correspondence to the patterns in the candlestick chart. The candlestick chart is plotted from the weekly data of the S&P500 index from 2020 to 2023. The first trading idea aims to capture the divergence of two moving average prices with differing lookback windows and the generated factor successfully reflects this curve. The second trading idea characterizes the breakout signals of Bollinger bands, and the corresponding alpha is a binary signal that gets activated when the upper bound is crossed. The third trading idea aims to capture three consecutive bullish movements on the candlestick chart, and the generated alpha successfully identified those patterns. These examples demonstrate that the generated alphas correctly capture the trading ideas.

Alpha Explanation Figure 8 presents examples of alpha expressions generated by Alpha-GPT based on given trading ideas, and the corresponding natural language explanations of these alphas also generated by Alpha-GPT. From these examples we can see that Alpha-GPT can provide appropriate explanations of the generated alphas, relieving the burden of human researchers to interpret these expressions by themselves.

6 Related Work

A lot of algorithms have been studied for formulaic alpha mining. Examples include Monte Carlo random search, Markov-chain Monte Carlo (Jin et al., 2020), genetic programming (Cui et al., 2021) and their variants (Zhang et al., 2020), and reinforcement learning (Yu et al., 2023). However, these methods all require the user to directly define the algorithmic configurations, providing limited interactivity compared with Alpha-GPT. Meanwhile, LLMs such as GPT (Brown et al., 2020) have demonstrated emergent abilities (Wei et al., 2022a) and achieved superior performance on various tasks. Besides, LLMs have also shown great reasoning (Wei et al., 2022b; Yao et al., 2023) and planning capabilities (Yao et al., 2022). In this way, an LLM can be regarded as a core thinking module and be integrated with various peripheral tools (Schick et al., 2023) to form intelligent LLM-powered agents (Weng, 2023).

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, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*. ArXiv: 2005.14165.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv preprint*. ArXiv:2402.03216 [cs].

Can Cui, Wei Wang, Meihui Zhang, Gang Chen, Zhaojing Luo, and Beng Chin Ooi. 2021. AlphaEvolve: A Learning Framework to Discover Novel Alphas in Quantitative Investment. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2208–2216, Virtual Event China. ACM.

R.N. Elliott and R.R. Prechter. 2005. *R.N. Elliott's Masterworks: The Definitive Collection*. New Classics Library.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 Herd of Models. *arXiv preprint*. ArXiv:2407.21783 [cs].

Jiadong Guo, Jingshu Peng, Hang Yuan, and Lionel Ming-shuan Ni. 2023. HXPY: A High-Performance Data Processing Package for Financial Time-Series Data. *Journal of Computer Science and Technology*, 38(1):3–24.

Jian Guo, Saizhuo Wang, Lionel M. Ni, and Heung-Yeung Shum. 2022. Quant 4.0: Engineering Quantitative Investment with Automated, Explainable and Knowledge-driven Artificial Intelligence. *arXiv preprint*. ArXiv:2301.04020 [cs, q-fin].

Ying Jin, Weilin Fu, Jian Kang, Jiadong Guo, and Jian Guo. 2020. Bayesian Symbolic Regression. *arXiv preprint*. ArXiv:1910.08892 [stat].

Zura Kakushadze. 2016. 101 Formulaic Alphas. *arXiv:1601.00991 [q-fin]*. ArXiv: 1601.00991.

Andrew W. Lo, Harry Mamaysky, and Jiang Wang. 2000. Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *The Journal of Finance*, 55(4):1705–1765. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/0022-1082.00265.

Scott Lundberg, Ryan Serrao, and other contributors. 2024. slundberg/shap: A game theoretic approach to explain the output of any machine learning model.

Myle Ott, Sam Shleifer, Min Xu, Priya Goyal, Quentin Duval, and Vittorio Caggiano. 2021. Fully Sharded Data Parallel: faster AI training with fewer GPUs.

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 [cs].

Igor Tulchinsky. 2019. Introduction to Alpha Design. In *Finding Alphas*, pages 1–6. John Wiley & Sons, Ltd.

Saizhuo Wang, Hang Yuan, Lionel M. Ni, and Jian Guo. 2024. QuantAgent: Seeking Holy Grail in Trading by Self-Improving Large Language Model. *arXiv preprint*. ArXiv:2402.03755 [cs, q-fin].

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. 2022a. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Lilian Weng. 2023. LLM Powered Autonomous Agents. Section: posts.

Yufei Wu, Daniele Magazzeni, and Manuela Veloso. 2021. How Robust are Limit Order Book Representations under Data Perturbation? In *ICML Workshop on Representation Learning for Finance and E-Commerce Applications*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv preprint*. ArXiv:2305.10601 [cs].

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2022. ReAct: Synergizing Reasoning and Acting in Language Models.

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. 2023. Generating Synergistic Formulaic Alpha Collections via Reinforcement Learning. ArXiv:2306.12964 [cs, q-fin].

Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. 2020. AutoAlpha: an Efficient Hierarchical Evolutionary Algorithm for Mining Alpha Factors in Quantitative Investment. *arXiv preprint*. ArXiv:2002.08245 [q-fin].

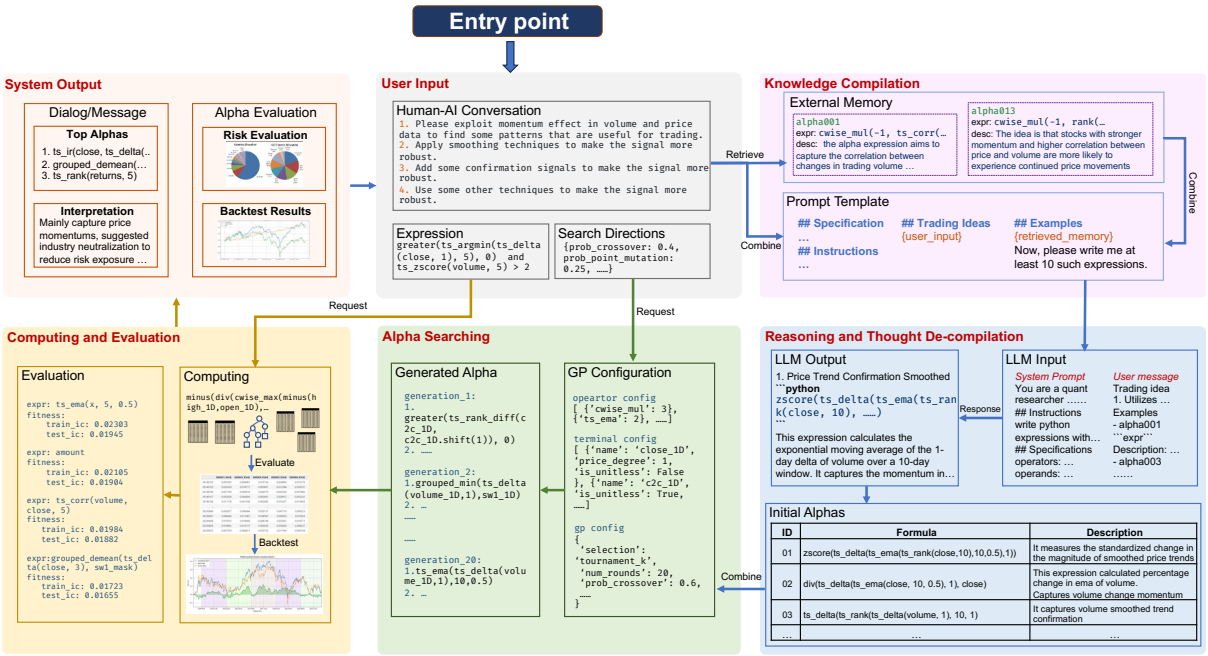


Figure 5: Alpha-GPT internal working pipeline: After a user inputs their ideas, the system goes into the knowledge compilation module. It uses external memory to pull similar examples, and combines them into the system prompt. The module passes everything to the LLM which creates valid alpha expressions and config files. These alphas are evaluated via Alpha Search, and results are presented to the user along with an interpretation provided by the Thoughts Decompiler.

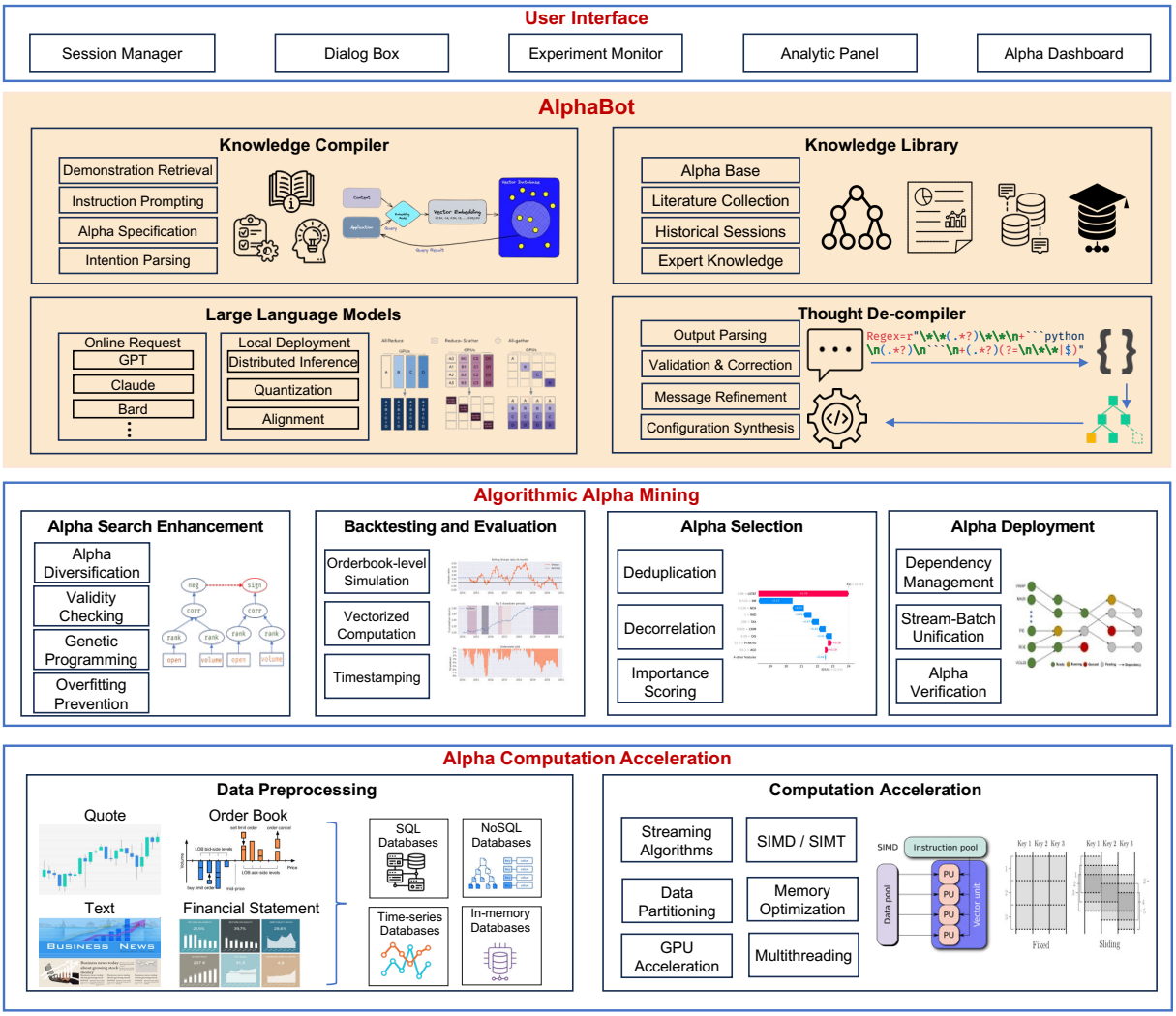


Figure 6: Alpha-GPT system architecture. The AlphaBot layer is the key contribution of this paper and the lower-level modules is integrated from our existing systems. Part of this figure is cited from (Guo et al., 2022; Wu et al., 2021; Ott et al., 2021; Lundberg et al., 2024).

(a) Golden-cross pattern (b) Bollinger bands (upper breakout) (c) Three white soldiers

Figure 7: Trading patterns and the corresponding alphas generated by Alpha-GPT that capture them.

Trading idea: *Flow of Funds*
Alpha: `div(cwise_mul(cwise_max(minus(close,shift(close, 1)), 0), amount, cwise_mul(close, volume)))`
Explanation: This expression calculates the ratio between the inflow of funds on days where the price increased and the product of the closing price and the trading volume of the current day. This ratio represents the intensity of the inflow of funds relative to the trading value.

Trading idea: *Volume-price correlation*
Alpha: `zscore_scale(ts_corr(close, volume, 20))`
Explanation: This alpha expression aims to capture the correlation between the daily close prices and the trading volume. A high positive correlation may indicate increased trading activity and potentially signal a price trend. The z-score scaling further helps identify extreme or outlier correlation values.

Trading idea: *Shadow*
Alpha:`div(cwise_max(minus(high,open),minus(high,close)),minus(high, low))`
Explanation: The trading idea behind this alpha expression is to identify situations where the upper shadow of a candlestick is relatively large compared to the entire range of the candlestick. This can indicate potential selling pressure or resistance at higher price levels.

Trading idea: *Momentum*
Alpha: `ts_delta(ts_rank(div(ts_delta(close,1),close),10),1)`
Explanation: This alpha expression calculates the time series delta between the current value and the previous value of the rank of the ratio between the time series delta of the daily closing price and the current closing price over a period of 10 days. The trading idea is to capture the change in the ranking of stocks based on the relative price changes, which can provide insights into shifts in market sentiment or price momentum.

Figure 8: Alphas generated based on trading ideas and the corresponding explanations generated by Alpha-GPT.

AgentDiagnose: An Open Toolkit for Diagnosing LLM Agent Trajectories

Tianyue Ou Wanyao Guo Apurva Gandhi Graham Neubig Xiang Yue
Carnegie Mellon University

Abstract

Large Language Model (LLM) agents produce rich, multi-step trajectories that interleave observations, internal reasoning, and tool actions. However, most evaluation pipelines focus solely on end-task success, leaving the agent's decision-making process opaque and poorly understood. We introduce AgentDiagnose, an open-source, modular framework for diagnosing agent trajectories. The present release fully supports the web domain, and AgentDiagnose is architect as an extensible, open platform with compatibility for most agent trajectories. AgentDiagnose consists of (i) an *evaluation* module that quantifies five core agentic competencies—backtracking & exploration, task decomposition, observation reading, self-verification, and objective quality—and (ii) a *visualization* module that highlights trajectory semantics through t-SNE action embeddings, interactive word clouds, and state-transition timelines. On a set of 30 manually annotated trajectories, our automatic metrics achieve a mean Pearson correlation of 0.57 with human judgments, rising to 0.78 for task decomposition. Furthermore, filtering the 46k-example NNetNav-Live dataset with AgentDiagnose and fine-tuning a Llama-3.1-8B model on the top 6k trajectories improves WebArena success rates by 0.98, despite using only 13% of the original data. AgentDiagnose thus serves as both a diagnostic lens for agent analysis and a practical tool for curating higher-quality training data. The toolkit and demo are publicly available.[1][2]

1 Introduction

Large Language Model (LLM) agents are rapidly gaining traction in domains such as web navigation (Xu et al., 2025; Murty et al., 2025), GUI automation (Xie et al., 2024; Qin et al., 2025), software engineering (Wang et al., 2025; Jimenez et al., 2024), social interaction (Jhamtani et al., 2025), and even healthcare (Tang et al., 2024).

Unlike traditional models that produce a single output, LLM agents operate over multi-step interaction trajectories. Each trajectory interleaves external observations, internal reasoning steps, and executable actions, producing an output sequence of the form: $T = \{(o_1, r_1, a_1), (o_2, r_2, a_2), \ldots, (o_n, r_n, a_n)\}$, where o_i represents the observation at step i, r_i the agent's reasoning, and a_i the resulting action. These trajectories can span hundreds or thousands of steps (Anthropic, 2025), making it difficult to analyze what drives an agent's success or failure.

Existing agent evaluation pipelines focus primarily on whether an agent completes a task successfully. While this outcome-oriented evaluation is useful, it leaves the agent's decision-making process opaque. Inspection tools such as AgentXRay (Murty et al., 2024) and AgentOps (AgentOps-AI, 2025) present step-level breakdowns of trajectories, visualizing the agent's observations, reasoning, and actions. However, they mainly provide readable replays of what the agent did, rather than diagnosing how well the agent reasoned about the task or how effectively it corrected mistakes along the way.

Complementary to these, evaluator frameworks such as AgentEval (LangChain-AI, 2025) judge an agent's trajectory against a reference solution, often using an LLM-as-a-judge. These approaches provide holistic scores but require gold-standard trajectories for comparison and overlook agent-internal competencies such as task decomposition or exploration.

In this paper, we introduce AgentDiagnose, an open-source diagnostic toolkit designed to fill this gap. AgentDiagnose helps researchers and practitioners go beyond success/failure judgments and surface key properties of agentic behavior. It consists of two core modules: 1) **Evaluation module:** Automatically scores five agentic competen-

[1] https://AgentDiagnose.live/
[2] https://github.com/oootttyyy/AgentDiagnose

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 207–215

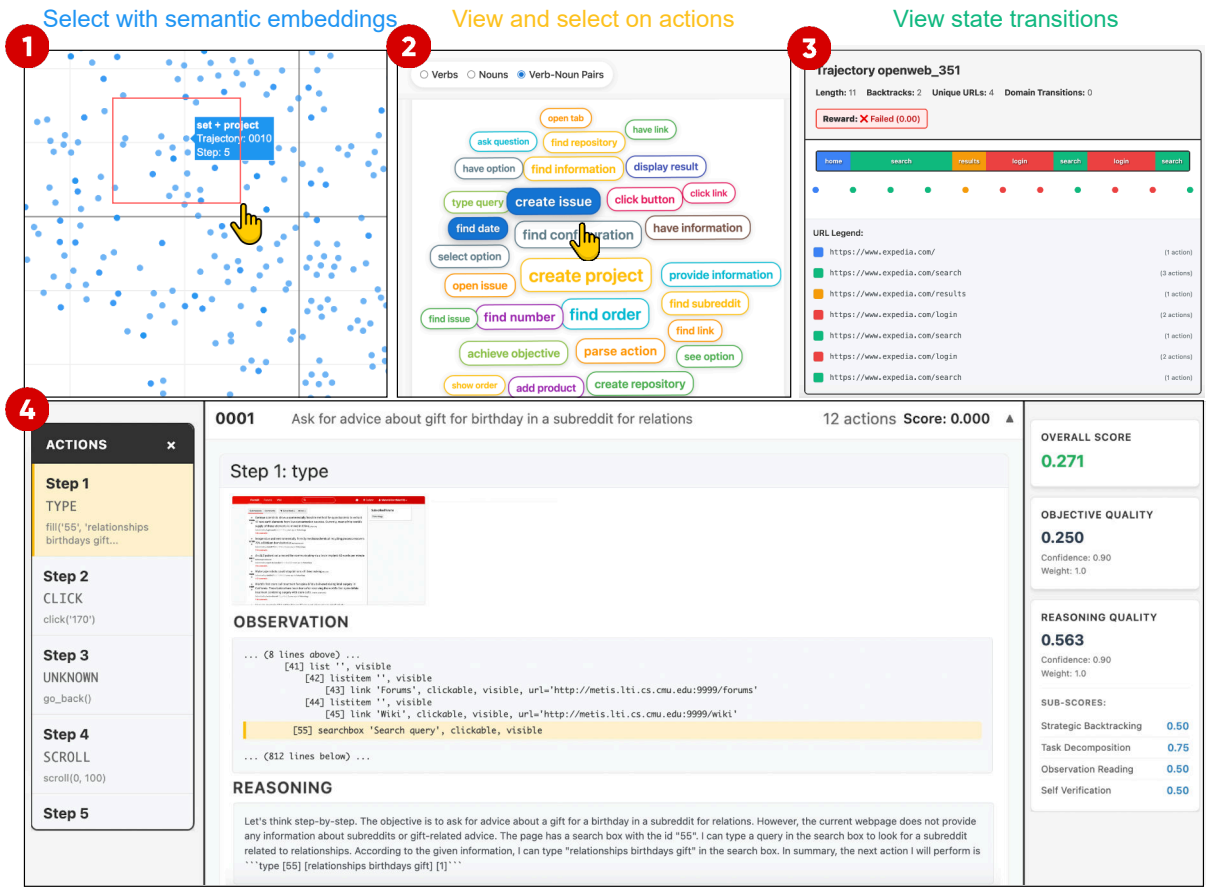


Figure 1: Overview of dashboard. 1 shows selecting points of interests in semantic embedding plot. 2 shows selecting on phrases of interests in word cloud. Selecting in 1 and 2 will display corresponding trajectories in trajectory_viewer 4. 3 shows navigation graph. 4 the trajectory_view panel displays details of all trajectories with evaluator scores summaries.

cies—backtracking & exploration, task decomposition, observation reading, self-verification, and objective quality—without requiring reference trajectories. 2) **Visualization module:** Provides interactive plots (e.g., t-SNE embeddings, word clouds, state transition timelines) that make it easy to explore semantic patterns in trajectories and pinpoint problematic behaviors.

We demonstrate AgentDiagnose's utility both as an analytical tool and as a data selection pipeline. Automatic trajectory scores correlate well with human annotations (mean Pearson $r = 0.57$, up to 0.78 for task decomposition). Furthermore, fine-tuning an LLM agent on AgentDiagnose-filtered high-quality trajectories from the NNetNav-Live dataset leads to a significant performance boost: a Llama-3.1-8B model trained on only the top 13% of trajectories outperforms one trained on the full dataset. In summary, AgentDiagnose provides a much-needed microscope for diagnosing agentic reasoning and offers practical levers for improving dataset quality and agent performance.

2 Related Work

Trajectory Inspection and Monitoring : A number of tools have been developed to visualize and monitor agent trajectories. For example, AgentLab's "Agent X-Ray" interface (Chezelles et al., 2025), IBM's "Agent Trajectory Explorer" (Desmond et al., 2025), OpenHand's "Trajectory Visualizer" (All-Hands-AI, 2025) provide step-by-step replay of an agent's interactions with a web environment, showing each observation, the agent's intermediate reasoning, and the action taken. Similarly, these inspection frameworks focus on rendering the raw trajectory in a human-readable form, often as a sequence of browser screens or tool API calls with the agent's thoughts. In parallel, agent observability platforms like AgentOps (AgentOps-AI, 2025), SEAVIEW (Bula et al., 2025) emphasize monitoring metrics and logging each decision step. AgentOps provides a developer dashboard to trace

agent behavior and detect anomalies or policy violations in real time. However, while these tools excel at presenting or recording the trajectory, they do not analyze the agent's decision-making quality beyond basic logging. For instance, they typically do not quantify how well an agent backtracked from errors or whether its plan was well-structured – gaps that our work aims to fill.

Trajectory Evaluation Frameworks : Beyond visualization, recent efforts have looked at evaluating the quality of agent trajectories. One approach is to compare the agent's sequence of actions against an expected or ideal sequence for the task. AgentEvals (Chase and Contributors, 2023), Vertex AI Agent Eval (Google Cloud, 2025) are frameworks that automates such comparisons. It can perform a trajectory match evaluation or use an LLM-as-a-judge to score how closely an agent's trajectory aligns with a reference solution. This method provides a holistic success measure of a trajectory when a reference is available. Our proposed toolkit differs by diagnosing trajectories along general-purpose dimensions without needing a predetermined correct sequence.

3 AgentDiagnose

AgentDiagnose can be used as a terminal tool or a visualization dashboard. It consists of two modules. (1) An evaluation module that gives a numeric evaluation score on key properties of trajectories. (2) A visualization module that directly displays key properties that are difficult to quantize into scores.

Key Trajectory Metrics Trajectories contain rich information that can be easily buried in the long chain of observation, reasoning, and actions. For instance, agents' adeptness in exploring alternatives (Shen et al., 2025) is a key measure of an agent's ability but it is not immediately clear how good an agent is in this regard just by looking at the raw sequence of steps. AgentDiagnose supports adding customized evaluators to access any property of interest. And as an example, we implemented five such evaluators to expose key agentic properties from trajectories as listed in Figure 2:

- **Backtracking and Exploration**: exploring alternatives is a key aspect of good agents. AgentDiagnose measures and exposes how good an agent is in backtracking when it heads into a wrong path. AgentDiagnose looks for patterns of backtracking and exploration in agents' sequence of actions and their corresponding reasoning. It uses a designed criterion to assign numeric scores based on the patterns it has identified.

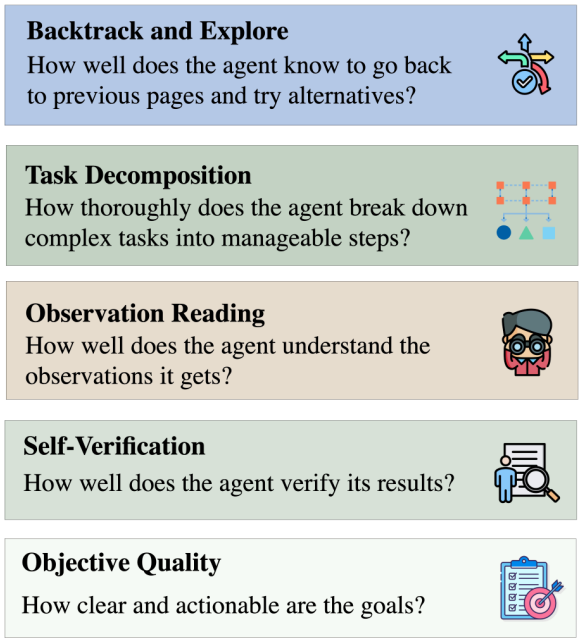


Figure 2: Description of key trajectory properties.

- **Task Decomposition**: knowing how to best decompose complex tasks into manageable sub-tasks is a key ability of successful agents (Prasad et al., 2023). Most of the decomposition happens at the first step, but studies have shown that revisiting and updating plans as agent interacts with the environment further improves performance. To help expose agents' abilities to decompose tasks, AgentDiagnose finds patterns in reasoning traces of the trajectory and assign numeric scores with our evaluation criteria. It exposes agents' abilities in decomposing complex tasks into manageable steps.

- **Self-verification**: AgentDiagnose measures how well the agent is in self-verifying its course of actions against the task goal. Self-verification is beyond checking if the final answer is satisfying the task requirement. Reasoning within each step of execution should reflect agent's awareness of the overarching goal.

- **Observation Reading**: correctly finding the right information from the observation is a major challenge for agents (Cheng et al., 2024). AgentDiagnose helps evaluate on the

agents' abilities to correctly understand the observation and capture relevant information.

- **Objective Quality**: the clarity and actionability of the task goal are a deciding factor in a trajectory's quality. Without a well-defined goal, it is difficult to determine how well the agent has completed the task. AgentDiagnose also utilizes a scoring criterion to measure how clear and actionable an objective is.

Key Property Visualizer However, not all information is best suited to be summarized by a numeric score. For instance, to understand the action distribution of agents, a t-SNE embedding plot of action verbs provides a more comprehensive view (van der Maaten and Hinton, 2008). AgentDiagnose comes with a visualizer that displays trajectories' intrinsic information through a combination of t-SNE embedding plots, word cloud, and timeline plots.

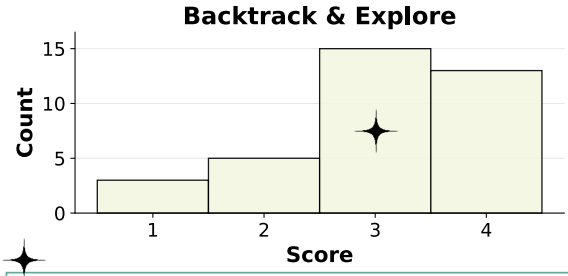
AgentDiagnose provides methods to extract out action verbs and nouns within agents' reasoning as shown in Figure 1 1. Currently AgentDiagnose has six options built in: root verbs, root nouns, root verb-noun pairs, all verbs, all nouns, and all verb-noun pairs. Verbs and nouns are extracted from agents' reasoning using Berkeley Neural Parser (benepar) (Kitaev et al., 2018). They are embedded with embedding models of choice, with the default being Qwen3-Embedding-0.6B (Zhang et al., 2025). AgentDiagnose generates t-SNE plots for each embedding. Users can select areas of interest on the semantic embedding plot. Once selected, users can inspect further on the trajectories of interests in the 4 view_trajectory tab, as shown in Figure 1.

AgentDiagnose also provides word cloud visualization on various components in agents' output. As shown in Figure 1 2, users can view the root verb, root noun, and root verb-noun pairs in the form of word cloud. In addition to action terms, AgentDiagnose builds word cloud for n-grams of reasoning output of agents. User can customize n-grams of n to display in a word cloud to uncover patterns in agents' reasoning. Similar to t-SNE plots, AgentDiagnose's word cloud supports selection of any one or multiple of the phrases in the word cloud to display their corresponding trajectories in the view_trajectory tab.

As shown in Figure 1 3, AgentDiagnose provides visualization of navigation paths within trajectories for direct view of the state transitions in each trajectories. Each color block on the timeline represents a state (a URL in this case), while each dot represents an action performed while in this state. Agent behaviors such as backtracking and parallel exploration can be easily identified through this view.

4 Usage Cases

4.1 Curate High Quality Trajectories



Goal: Find the best Peloton bike seat.

...

Step 9:

Let's think step-by-step. We've already browsed through Peloton Bike accessories on Amazon and sorted them by price. We also seem to have visited a product page with 2,826 ratings. ..... We can see that its value is 'Peloton Bike accessories', which confirms our search query. However, to determine if we're looking at the best product, we need to analyze the product's details and possibly other customer reviews. Since I couldn't find any extra details about the product in this accessibility tree and I also can't find the 'next' button for looking at other products, I think it's good idea to go back and try a different search term or sort option. In summary, the next action I will perform is ```go_back```

...

Figure 3: Top: Backtrack & explore score distribution from the evaluator. Bottom: corresponding step that exhibits exploratory behavior.

The automatic evaluator of key properties is a fast way to identify strengths and weaknesses on a trajectory sample. AgentDiagnose's evaluator is implemented both as a python extension that can be invoked with a single command line in the terminal, as shown in 1, and as a part of the dashboard for visualization. Upon running the evaluator, results are immediately stored locally. Users can further choose to spawn them as interactive bar plots in separate tabs in the visualization dashboard. One use-case of the evaluated scores is to filter for high quality training data. Here we show an example of evaluation results. As seen in Figure 3, this set of trajectories exhibits nice backtracking and exploration ability, with most of the scores in the threes and fours. As in shown in the detail reasoning, one such trajectory demonstrates the agents' intent to explore additional Peloton products to make sure it is finding the "best one."

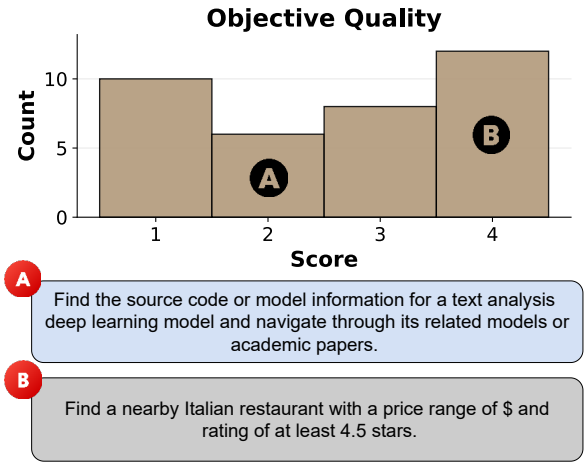


Figure 4: Top: Objective quality score distribution from the evaluator. Bottom: corresponding objectives that are scored above.

Evaluation on objective quality of the batch indicates objective of this batch has mixed quality. As shown in Figure 4, around half of the objectives in the batch have scores of ones and twos. Corresponding objectives are shown in the figure. Higher scores of objective correspond to more actionable and specific task objectives. Objective A, as shown in the figure, does not have a clear end goal and therefore is not a good training sample. Based on these scores, we are able to select high quality training data that demonstrates good backtrack and explore behaviors as well as high quality task objectives. Users can further display the plots on dashboard. To select and view the trajectories of a particular score, users can select score ranges on the bar plot displayed in the dashboard, corresponding trajectories will then be displayed in the view_trajectory tab. As users go through the trajectories, a mini-panel of scores will also be displayed for reference, as shown in Figure 1. Detailed scoring criteria are shown in Appendix A.

4.2 Visualize Intrinsic Properties

We follow the trajectory inspection paradigm:

- Rapidly skim trajectories to identify emerging patterns.
- Select key patterns for detailed, in-depth analysis.

AgentDiagnose provides methods to visualize various intrinsic properties of trajectories that would otherwise buried in sequence views.

Embedding Plot As shown in (1) of Figure 1, users can inspect clustering of actions verb-noun

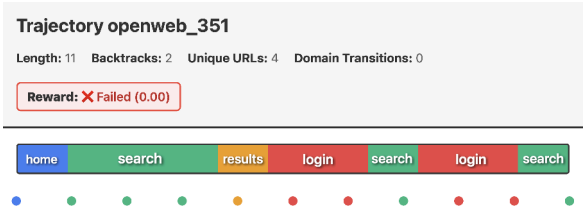


Figure 5: Screenshot of the navigation graph.

embeddings. For example, when an agent fails in setting up GitLab projects, users want to inspect trajectories that involved handling GitLab projects. To do so, they can select from the embedding plot as in (1) of Figure 1, on the semantic embeddings surrounding the point of "set project". Surrounding the point of "set project" are points of "create project", "fork project" etc. By selecting this region, users can pull up the corresponding trajectories in the view_trajectory page for detailed investigation. Semantic embedding plot allows users to scan through patterns and easily focus on any interesting semantic patterns and their corresponding trajectories.

Word Cloud As shown in (2) of Figure 1, users can use word clouds to identify common patterns in an agent's reasoning. AgentDiagnose supports word cloud for actions and n-grams in reasoning. In the example of (2) Figure 1, verb-noun pairs in agents' reasoning are displayed by their frequencies of appearance. We can see the verb-noun pair "create project" has appeared frequently. We can click on any one or multiple of the tabs to display their corresponding trajectories in the view_trajectory tab.

Listing 1: Command-line to run evaluator.

```
python evaluate_trajectories.py \
  --input ./input.json \
  --scorers reasoning_quality \
            objective_quality \
            navigation_path \
  --output-json output.json
```

Navigation Graph Users can use navigation path to quickly spot navigation-related errors. As shown in (3) of Figure 1, the navigation path abstracts out the state transitions from the trajectories so it is clear where the agent has been. One use case is to identify navigation-related errors. In the example shown Figure 5, the failed trajectory of openweb_351 is stuck in a loop between login and search page. Considering that the agent has used search page successfully early in the trajectory, the possible error may be in handling login page. The

Dimension	r	ρ	τ
backtrack & explore	0.39	0.43	0.36
task decomposition	0.78	0.86	0.76
observation reading	0.62	0.63	0.60
self-verification	0.56	0.58	0.55
objective quality	0.54	0.61	0.56
Overall	0.57	0.62	0.56

Table 1: Correlation coefficients (Pearson r, Spearman ρ, Kendall τ) between human annotated scores and evaluator's on five key properties.

abstraction view of navigation path allows users to quickly scan through the path agent has taken within each trajectory.

5 Evaluation

5.1 Compare to Human Evaluation

To evaluate the accuracy of our evaluator, we manually annotated 30 trajectories on the five key properties and compare with the model evaluated scores. We give the same input of agent reasoning-action sequence and scoring criteria to both the model and annotator. We measured the agreement with Pearson (Pearson, 1895), Spearman (Spearman, 1904), and Kendall-Tau correlation coefficients (Kendall, 1938). Overall, AgentDiagnose's evaluator achieves a 0.57 Pearson correlation coefficient, 0.62 Spearman correlation and 0.56 Kendall-Tau correlation, as shown in Table 1. Overall, it showcases a positive correlation between human scoring and AgentDiagnose's evaluator scoring. In particular, agreement rate is the highest in the case of task decomposition, reaching 0.78 Pearson correlation, showing that task decomposition behavior is relatively easy to identify and differentiate. In contrast, backtracking and exploration only reaches a 0.39 Pearson correlation, indicating the intrinsic difficulty in extracting agents backtracking behaviors and evaluating their precision and correctness.

5.2 Improve Agent Performance With AgentDiagnose

AgentDiagnose's evaluator act as effective training data selector. We experimented by fine-tuning on the full NNetNav-Live dataset and a high quality subset selected by AgentDiagnose's evaluator. We applied AgentDiagnose's evaluator on the 46k training samples of the NNetNav-Live dataset, using Gemini2.5-pro as the LLM judge (Gemini Team, Google, 2025), and we selected

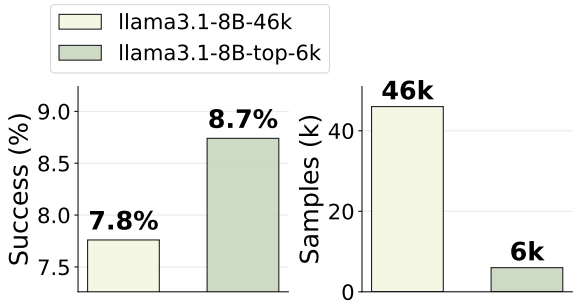


Figure 6: Comparison of WebArena success rate(left) and number of samples used in training(right) by llama3.1-8B-46k and llama3.1-8B-top-6k.

6k top scoring samples as our high quality subset. Following the setup in NNetNav, we finetuned a LLama3.1-8B model on our high quality subset and evaluated on WebArena with browserGym (Zhou et al., 2023), (Chezelles et al., 2025). Results are shown in Figure 6: llama3.1-8B-top-6k outperforms llama3.1-8B-46k despite only using 13% of the training samples. llama3.1-8B-top-6k achieves 0.98 higher success rate than Llama-3.1-8B -46k. With AgentDiagnose evaluator's filtering ability, it could act as data quality controller in training data collection and provide valuable signal on trajectory's quality beyond the success and failure measurement.

6 Conclusion

We demonstrated AgentDiagnose, an open-source framework that brings much-needed diagnostic visibility to the rapidly growing field of LLM agents. By coupling a lightweight, LLM-powered evaluation module with an interactive visualization module, AgentDiagnose exposes five critical competencies: backtracking & exploration, task decomposition, observation reading, self-verification, and objective quality—that are largely invisible to traditional trajectory inspectors. And by improving agent's performance on web navigation task, we showcase our trajectory-centric quality signals can translate directly into performance gains.

7 Limitations

AgentDiagnose is designed primarily with web-navigation tasks in mind. Its applicability to other use cases, such as coding, as well as multimodal settings.

Current AgentDiagnose's evaluator system leverages LLM-based evaluation. Future iterations could improve reliability by training specialized evaluators with fine-grained human annotations.

8 Ethics Statement

Human annotation Thirty trajectories were annotated for the correlation study. Annotators are college student volunteers. No demographic attributes beyond language proficiency were collected. The task required no sensitive information.

Broader Impact By shining light on how agents reason, rather than solely whether they succeed, AgentDiagnose can help researchers identify failure modes, debug unsafe behaviors, and design more transparent systems.

References

AgentOps-AI. 2025. Agentops: Python sdk for ai agent monitoring, llm cost tracking, benchmarking, and more. https://github.com/AgentOps-AI/agentops. MIT License.

All-Hands-AI. 2025. All-hands-ai/trajectory-visualizer. https://github.com/All-Hands-AI/trajectory-visualizer. GitHub repository, accessed July 3, 2025.

Anthropic. 2025. Claude opus 4. https://www.anthropic.com/claude/opus. Accessed: 2025-07-01.

Timothy Bula, Saurabh Pujar, Luca Buratti, Mihaela Bornea, and Avirup Sil. 2025. Seaview: Software engineering agent visual interface for enhanced workflow. *Preprint*, arXiv:2504.08696.

Harrison Chase and LangChain Contributors. 2023. Langchain. https://www.langchain.com/. Accessed: 2025-07-03.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Annual Meeting of the Association for Computational Linguistics*.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. 2025. The browsergym ecosystem for web agent research. *Preprint*, arXiv:2412.05467.

Michael Desmond, Ja Young Lee, Ibrahim Ibrahim, James M. Johnson, Avirup Sil, Justin MacNair, and Ruchir Puri. 2025. Agent trajectory explorer: Visualizing and providing feedback on agent trajectories. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(28):29634–29636.

Gemini Team, Google. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. Technical report, Google DeepMind, Mountain View, CA. Version 2.5: Pro and Flash variants; introduces million-token context window, multimodal input, advanced 'Deep Think' reasoning.

Google Cloud. 2025. Evaluate gen ai agents. Documentation for evaluating generative AI agents on Vertex AI.

Harsh Jhamtani, Jacob Andreas, and Benjamin Van Durme. 2025. Lm agents for coordinating multi-user information gathering. *Preprint*, arXiv:2502.12328.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? *Preprint*, arXiv:2310.06770.

Maurice G. Kendall. 1938. *A New Measure of Rank Correlation*, volume 30. Biometrika Trust.

Nikita Kitaev, Lukasz Kaiser, and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686. Association for Computational Linguistics.

LangChain-AI. 2025. Agentevals: Readymade evaluators for agent trajectories. https://github.com/langchain-ai/agentevals. Version 0.0.8.

Shikhar Murty, Dzmitry Bahdanau, and Christopher D. Manning. 2024. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. 2025. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild. *Preprint*, arXiv:2410.02907.

Karl Pearson. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. 2023. Adapt: As-needed decomposition and planning with language models. In *NAACL-HLT*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, and Aviral Kumar. 2025. Thinking vs. doing: Agents that reason by scaling test-time interaction. *Preprint*, arXiv:2506.07976.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2024. Medagents: Large language models as collaborators for zero-shot medical reasoning. *Preprint*, arXiv:2311.10537.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, and 5 others. 2025. Openhands: An open platform for ai software developers as generalist agents. *Preprint*, arXiv:2407.16741.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

Yiheng Xu, Dunjie Lu, Zhennan Shen, Junli Wang, Zekun Wang, Yuchen Mao, Caiming Xiong, and Tao Yu. 2025. Agenttrek: Agent trajectory synthesis via guiding replay with web tutorials. *Preprint*, arXiv:2412.09605.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2023. Webarena: A realistic web environment for building autonomous agents. *ArXiv*, abs/2307.13854.

A Evaluation Criteria

```
Backtracking (1-4): How well does the agent know to go back to previous pages and try alteratives?
- 4: Excellent - The agent accurately recognizes when it has taken a wrong path and take explicit actions to go back to a previous page to try alteratives
- 3: Good - The agent takes explicit actions to go back to try alteratives most of the time when it takes a wrong path
- 2: Mediocre - The agent has considered going back or trying alteratives, but has made mistakes in doing so
- 1: Poor - The agent has never considered trying alternatives or going back to previous states
- N/A: There is not a need to go back to previous states because the agent has taken the right path throughout the trajectory
```

```
Task decomposition (1-4): How thoroughly does the agent break down complex tasks into manageable steps?
- 4: Excellent - The agent breaks down complex tasks into detailed steps that cover the entire task
- 3: Good - The agent breaks down complex tasks, but not in all cases or leaves out steps
- 2: Mediocre - The agent breaks down complex tasks, but in very poor way
- 1: Poor - The agent makes no attempt in breaking down complex tasks
```

```
Observation reading (1-4): How well does the agent understands the observations it gets?
- 4: Excellent - The agent summarizes the observation accurately in each step and immediately notice the important information on the page
- 3: Good - The agent summarizes the observation in each step, but sometimes misses important information
- 2: Mediocre - The agent only summarizes the observation in some steps
- 1: Poor - The agent almost never summarizes the observation
```

```
Self-verification (1-4): How well does the agent verify its results?
- 4: Excellent - The agent checks carefully on its results against the objective throughout the trajectory
- 3: Good - The agent checks its results against the objective sometimes, but has room to improve. If it has done better checking, it could have done better on the task
- 2: Mediocre - The agent shows signs of attempting to verify its results
- 1: Poor - The agent never verifies its results against the objective
```

```
Objective-quality (1-4): How clear and actionable are the goals?
- 4: Excellent - Objective contains clear, specific, actionable goals with concrete success criteria
- 3: Good - Objective is mostly actionable with some clear goals
- 2: Mediocre - Objective has a mix of actionable elements and vague exploratory elements
- 1: Poor - Objective is entirely about exploration with no concrete targets
```

τ TAU-EVAL: A Unified Evaluation Framework for Useful and Private Text Anonymization

Gabriel Loiseau[1,2] **Damien Sileo**[2] **Damien Riquet**[1] **Maxime Meyer**[1] **Marc Tommasi**[2]
[1]Hornetsecurity, Hem, France
[2]Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 - CRIStAL, F-59000 Lille, France
gabriel.loiseau@inria.fr

Abstract

Text anonymization is the process of removing or obfuscating information from textual data to protect the privacy of individuals. This process inherently involves a complex trade-off between privacy protection and information preservation, where stringent anonymization methods can significantly impact the text's utility for downstream applications. Evaluating the effectiveness of text anonymization proves challenging from both privacy and utility perspectives, as there is no universal benchmark that can comprehensively assess anonymization techniques across diverse, and sometimes contradictory contexts. We present TAU-EVAL, an open-source framework for benchmarking text anonymization methods through the lens of privacy and utility task sensitivity. A Python library[1], code, documentation and tutorials[2] are publicly available.

1 Introduction

Privacy protection is a cornerstone of modern legal frameworks, encapsulated in regulations such as the European Union's General Data Protection Regulation (GDPR) and the United States' California Consumer Privacy Act (CCPA). These regulations underline the urgency of protecting personal data, particularly in text-based formats—a common medium for sensitive information sharing in domains like healthcare, legal proceedings, and social media. Text anonymization has emerged as a critical tool for this purpose (Lison et al., 2021), modifying texts to hide identifiable attributes while aiming to preserve their usefulness for downstream applications. However, this process inherently creates a tension: excessive anonymization risks rendering the text unusable for practical tasks, while insufficient redaction leaves private information vulnerable to exposure.

[1] https://pypi.org/project/tau-eval
[2] https://github.com/gabrielloiseau/tau-eval

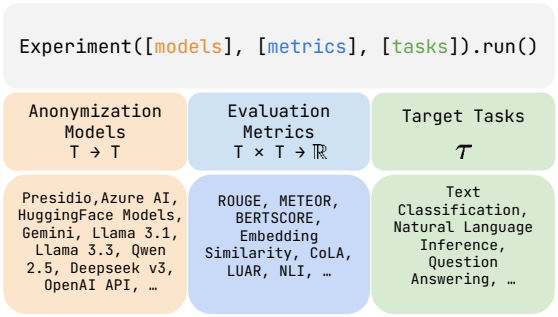


Figure 1: Summary of features in TAU-EVAL implementation. Each component can be customized and easily expanded though the system structure. We also include built-in examples for each component. TAU-EVAL relies on a core *Experiment* structure that encapsulate *Models* evaluated on *Tasks* in order to gather *Metrics*.

Current research on text anonymization often prioritizes privacy preservation at the expense of utility, relying on reference-based metrics like ROUGE, BERTScore, or METEOR to measure textual fidelity (Staab et al., 2024a; Pilán et al., 2022). While these metrics assess surface-level content retention and whether the resulting anonymized text lands in the same distribution, they fail to account for the context-dependent utility of anonymized texts (Yang et al., 2025; Loiseau et al., 2025). For instance, a medical report anonymized for public research must retain clinically relevant patterns, whereas a legal document might require syntactic integrity for compliance analysis. Anonymization methods that perform well on generic metrics may strip task-specific features, undermining the value of the data for real-world applications. Consequently, the privacy-utility trade-off remains poorly quantified, leaving practitioners without actionable insights for domain-specific anonymization scenarios (Riabi et al., 2024).

To bridge this gap, we introduce TAU-EVAL (Text Anonymization Utilities Evaluation), a framework designed to systematically evaluate both privacy preservation and task-aware utility

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 216–227
November 4-9, 2025

loss in text anonymization systems. Unlike existing frameworks, TAU-EVAL integrates privacy and utility evaluations across various downstream tasks, enabling granular analysis of how anonymization impacts domain-specific applications. It supports full specification of privacy and utility task targets, which empowers practitioners to evaluate anonymization strategies for their unique requirements and create reproducible evaluation benchmarks.

We showcase TAU-EVAL's versatility through experiments on two privacy and eight utility tasks. With our framework, we support the development of context-aware anonymization systems that balance privacy with the functional needs of domains like healthcare, social science, and law.

2 TAU-EVAL

In this section, we present TAU-EVAL's core design principles, architecture, and the functionality of its main components.

Problem formulation We formalize text anonymization as a text-to-text transformation task that applies privacy-preserving objectives (e.g., NER-based redaction, authorship obfuscation) to an input text while aiming to preserve its utility. To holistically evaluate anonymization methods, we propose a two-pronged framework: (1) generic metric analysis, measuring surface-level fidelity between original and anonymized texts, and (2) task utility loss quantification, assessing downstream performance degradation caused by anonymization.

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ denote a dataset of N samples, where x_i is an original text and y_i its task-specific label (e.g., classification targets). An anonymization algorithm $\mathcal{A} : x \to x'$ transforms each x_i into its private counterpart x'_i, yielding the anonymized dataset $\mathcal{D}_{\text{priv}} = \{(x'_i, y_i)\}_{i=1}^{N}$. For each sample pair (x_i, x'_i), we compute a similarity metric $s(x_i, x'_i)$ to quantify the text transformation. The overall generic fidelity of $\mathcal{A}$ is then:

$$S(\mathcal{A}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} s(x_i, x'_i),$$

providing a task-agnostic measure of anonymization's impact on textual integrity.

To evaluate task-specific utility degradation, we first train a model f_θ on $\mathcal{D}$, achieving a baseline performance $\mathcal{U}_{\text{orig}}$ on a held-out test set. We then anonymize the test set using $\mathcal{A}$ and evaluate f_θ on $\mathcal{D}_{\text{priv}}$, obtaining $\mathcal{U}_{\text{priv}}$. This setup mirrors real-world scenarios where users apply anonymization to data before feeding it to a pre-trained task model, which they cannot retrain or modify. The sensitivity for task $\mathcal{T}$ is: $\Delta_{\mathcal{T}}(\mathcal{A}) = \mathcal{U}_{\text{orig}} - \mathcal{U}_{\text{priv}}$, where $\Delta_{\mathcal{T}}$ captures the performance drop attributable to anonymization. Methods with high $S(\mathcal{A}, \mathcal{D})$ may still induce significant $\Delta_{\mathcal{T}}$ if anonymization perturbs task-relevant features, underscoring the necessity of joint analysis. More complex task schemes are also possible, such as training models on partially anonymized datasets to evaluate performance on anonymized outputs (Zhai et al., 2022).

2.1 Design Principles

TAU-EVAL is guided by three core principles: ease of use, modularity, and customizability, making it a flexible and accessible framework for evaluation in NLP.

Ease of use TAU-EVAL enables researchers to build complete evaluation pipelines with minimal code. It offers a simple interface and sensible defaults to support rapid development and experimentation.

Modularity The framework is composed of independent components that can be used selectively. This avoids unnecessary processing and supports a wide range of use cases.

Customizability Thanks to its modular design, users can easily integrate custom anonymization models, define new features, and implement specialized metrics, allowing for tailored evaluation workflows.

2.2 Core Elements and Functionalities

TAU-EVAL is an open-source Python framework designed to unify and streamline the evaluation of text anonymization systems. By abstracting fragmented experimental workflows into a modular and extensible framework, the library enables researchers to benchmark anonymization models against diverse metrics and tasks with minimal coding effort. Below, we detail its core architecture, integration capabilities, and workflow. To accommodate the rapid evolution of text generation models and datasets, TAU-EVAL leverages the Hugging Face ecosystem. It natively supports datasets (Lhoest et al., 2021) within tasks, including local storage and custom preprocessing pipelines, models (Wolf et al., 2020), and evaluations metrics (Von Werra

et al., 2022). TAU-EVAL's architecture (Figure 1) revolves around three modular pillars:

Anonymization Models: The framework treats anonymizers as black-box text-to-text functions. Requiring only a lightweight `Anonymizer` interface (`anonymize(text) -> text`). This perspective accommodates a broad spectrum of anonymization techniques; from traditional sequence-to-sequence architectures to modern large-scale language models. Preconfigured templates simplify the use of Hugging Face models (e.g., T5, BART, GPT-2) and LLM APIs via LiteLLM[3], enabling rapid prototyping.

Evaluation Metrics: Metrics are designed to capture diverse dimensions of performance. We integrate measures that assess both the retention of semantic content (e.g., overlap-based and semantic similarity scores) and intrinsic properties of anonymized text (e.g., fluency and reference-less evaluations), the framework provides a comprehensive basis for quantifying the efficacy of anonymization methods. Metrics are computed on each pair $(\mathcal{D}, \mathcal{D}_{\text{priv}})$ taken from task datasets. We implement widely used metrics for text anonymization taken from text generation evaluation, such as ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), BERTScore (Zhang et al., 2020b). We also support sentence-transformers similarity (Reimers and Gurevych, 2019), and reference-less metrics based on language models such as CoLA (Warstadt et al., 2019), or Perplexity (Jelinek et al., 1977).

Target Tasks: Tasks contextualize the evaluation by framing anonymization within downstream applications. We abstract common-use scenarios (e.g., classification, multiple-choice inference) into a unified task paradigm, TAU-EVAL facilitates systematic comparisons across a variety of use cases. We utilize the tasksource ecosystem (Sileo, 2024) to empower rapid access to more than 600 structured datasets and tasks preprocessing with the `AutoTask` class.

```
from tasknet import AutoTask
imdb = AutoTask('imdb')
dynahate = AutoTask('dynahate')
```

Using tasknet, we provide a streamlined interface between evaluation tasks and the Hugging Face trainer for efficient fine-tuning. This integration simplifies dataset importing, preprocessing, and model configuration, allowing researchers to concentrate on designing text anonymization pipelines rather than building evaluation frameworks. Additionally, we support additional built-in tasks that don't require model training, such as sensitive entity detection. Table 3 in appendix gives an overview of all features currently integrated into TAU-EVAL.

[3] https://www.litellm.ai/

3 Usage and Customization

This section provides implementation details for conducting evaluation experiments and explains how to further customize TAU-EVAL for specific research needs.

3.1 Running Experiments

Running an experiment involves three steps: (1) Implement custom anonymizers (see Section 3.2) or load preconfigured models present inside `tau_eval.models`. (2) Choose from built-in options or add custom metrics and tasks. (3) Instantiate an `Experiment` object with models, tasks, and metrics. All results are logged for visualization and comparison.

Experiment The `Experiment` class takes care of orchestrating the evaluation of each anonymization model. It extracts task information from each task, generates the anonymized task version, and computes each chosen metric, while training relevant models if needed. It relies on `ExperimentConfig`, a class storing specific user-defined configuration for evaluation.

ExperimentConfig The `ExperimentConfig` class provides ways to customize the experiment evaluation. In particular, it allows the user to specify which prediction model to train (from a local model or on the Hugging Face Hub) and fine-tune it on tasks, setting training hyperparameters, or more advanced tasks strategies (should we train the classifier on generated data or not) and logging options.

`Experiment` results can be stored either as a dictionary variable for immediate use or serialized to a `.json` file. While the dictionary format is convenient for most direct analysis, JSON serialization helps to store experiment results for versioning and future use.

3.2 Customization

Models Anonymization models are defined as a class which allow initialization i.e. model loading and inference with the anonymization method. A new anonymization model can easily be instantiated using this interface:

```
class TestModel(Anonymizer):
    def __init__(self):
        self.name = "Test Model"

    def anonymize(self, text) -> str:
        # Implement anonymization logic

    def anonymize_batch(self,
                        texts: list[str]) ->
                        ↪ list[str]:
        # Performs batch anonymization for larger
        ↪ datasets
```

Metrics Metrics are implemented as functions that accept one or two text inputs and return one or more scores. Experiments can utilize both TAU-EVAL's built-in metrics and custom functions, provided they follow the signature `Callable[[str | list[str], str | list[str]], dict]`. The experiment framework automatically detects and handles both built-in and custom metrics.

Tasks Tasks enable data loading and classification model integration. While tasksource and tasknet simplify access to the Hugging Face Hub, users can create more sophisticated tasks by implementing the `CustomTask` interface. This interface requires a `dataset` attribute containing a Hugging Face dataset and an `evaluate(self, new_texts: list[str]) -> dict` method that processes anonymized texts. During experiments, the system anonymizes the task dataset and passes it to the evaluation method.

3.3 Visualization

In the evaluation of anonymization systems, TAU-EVAL offers a suite of visualization tools designed to streamline the comparative analysis of models across specific tasks. Our system enables to assess performance through one-to-one model comparisons, enhancing interpretability. Additionally, TAU-EVAL supports the explicit definition of trade-offs between key metrics (e.g., privacy vs. utility), allowing users to explore the nuanced relationships between competing objectives. To further refine analysis, the framework provides flexible filtering mechanisms, permitting users to isolate experimental results based on models, tasks, or evaluation metrics. We also ensure a more structured and granular assessment of anonymization techniques by serializing parts of the anonymized datasets, which is useful for qualitative analysis and to relaunch experiments without model inference.

```
from tau_eval import Experiment, ExperimentConfig
from tau_eval.models.presidio import (
    UniquePlaceholderPerEntity,
    EntityDeletion,
    UniformPlaceholder,
    CategoryPlaceholder,
    FakerPlaceholder,
)
from tau_eval.tasks import DeIdentification
from tasknet import Classification
from datasets import load_dataset

m1 = UniquePlaceholderPerEntity()
m2 = EntityDeletion()
m3 = UniformPlaceholder()
m4 = CategoryPlaceholder()
m5 = FakerPlaceholder()

mednli = Classification(
    dataset=load_dataset("bigbio/mednli"),
    s1="sentence1", s2="sentence2", y="label"
)
pii = DeIdentification(
    dataset="ai4privacy/pii-masking-400k"
)

config = ExperimentConfig(
    exp_name="test-experiment",
    classifier_name="answerdotai/ModernBERT-base",
    train_task_models=True,
    train_with_generations=False,
)

Experiment(models=[m1,m2,m3,m4,m5],
           metrics=["bertscore"],
           tasks=[mednli, pii],
           config=config
).run()
```

Listing 1: Example running an experiment comparing different de-identification strategies for the MedNLI and `pii-masking` datasets. BERTScore will be computed on each dataset for each model.

4 Experiments

To demonstrate the potential of TAU-EVAL, we evaluate the utility loss of anonymization methods across two privacy objectives (PII redaction and authorship obfuscation) and eight downstream utility tasks, spanning diverse domains to capture task-sensitive utility loss. Below, we detail our benchmarking setup.

Privacy Tasks To quantify privacy risks, we focus on two objectives. First, personally identifiable information (PII) redaction involves automatically

Model	PRIVACY	IMDB	DYNASENT	TOXICITY	DYNAHATE	ANLI	MEDNLI	FRAUD	FAKE NEWS
PII Redaction									
Original	0	95.2	77.6	75.6	83.0	56.5	66.3	99.7	98.5
Presidio	66.4	95.1	77.5	75.7	80.0	49.6	66.1	97.1	97.9
Gemini-flash-1.5-8b	96.6	93.7	77.1	56.8	45.0	50.2	65.3	98.3	78.9
Gemini-flash-1.5	98.4	94.8	77.4	53.2	41.6	47.8	61.9	97.3	77.1
Llama-3.1-8b	99.0	94.6	71.8	54.2	49.6	47.7	51.6	92.0	92.7
Llama-3.1-70b	98.7	95.6	74.6	60.8	51.2	49.2	59.8	98.8	85.2
Llama-3.3-70b	98.7	95.5	77.0	69.4	53.1	48.9	63.7	98.7	88.9
Qwen-2.5-7b	95.0	94.9	74.2	65.1	57.5	50.1	61.5	98.6	94.8
Qwen-2.5-72b	97.8	95.3	76.8	61.1	55.9	52.5	60.4	98.7	88.5
Phi-4	97.2	95.1	75.3	58.0	44.7	51.6	61.4	97.5	84.6
Authorship									
Original	0.2	95.2	77.6	75.6	83.0	56.5	66.3	99.7	98.5
StyleRemix	70.4	82.9	73.2	50.8	47.5	42.5	47.3	97.3	73.9
Gemini-flash-1.5-8b	80.6	94.3	70.9	52.4	52.7	45.0	45.9	96.7	47.9
Gemini-flash-1.5	65.6	95.2	71.1	56.9	57.7	44.2	46.2	96.8	58.4
Llama-3.1-8b	76.1	93.8	62.6	53.3	55.1	40.7	42.0	94.1	51.5
Llama-3.1-70b	69.7	95.2	65.1	51.3	57.7	42.3	41.3	95.3	54.8
Llama-3.3-70b	69.3	94.3	65.9	52.9	58.6	41.1	41.8	95.9	60.0
Qwen-2.5-7b	66.1	91.9	65.4	58.2	63.5	43.5	46.5	96.2	73.8
Qwen-2.5-72b	60.2	94.5	70.7	59.3	66.0	40.3	46.6	96.0	75.7
Phi-4	63.4	94.2	66.4	49.9	59.0	44.4	48.7	95.6	59.6
Random	10.1	50.0	33.3	50.0	50.0	33.3	33.3	50.8	50.4

Table 1: Task-specific degradation of each anonymization model (%). The PRIVACY column highlights the models performance over the two privacy datasets: `pii-masking` and IMDB62.

identifying and replacing sensitive personal data that could potentially expose individual identities (e.g., names, addresses, etc.). This task is evaluated using the synthetic `pii-masking` dataset[4], which generates realistic PII in contextual scenarios while avoiding exposure of real private data. This synthetic approach enables safe benchmarking of anonymizers' ability to mask sensitive entities without compromising genuine user privacy, we report the masked entity recall as a privacy goal to maximize. Second, authorship obfuscation is a privacy task that tests whether unique authorship features can be used to identify or re-identify an individual's textual content despite anonymization attempts. The task leverages the IMDB62 corpus (Seroussi et al., 2014), a widely adopted benchmark for authorship attribution. Here, anonymization aims to obfuscate stylistic fingerprints, testing resilience against authorship inference attacks, we evaluate this task with accuracy scores.

Utility Tasks To comprehensively measure task-specific utility degradation, we select eight classification tasks spanning diverse domains and linguistic complexity, each chosen to probe distinct challenges in privacy-utility trade-offs. IMDB movie reviews (Maas et al., 2011) and the adversarially constructed DynaSent dataset (Potts et al., 2021) serve as baselines for *sentiment analysis*. *Toxicity and hate speech detection* (Jigsaw Toxic Comments (cjadams et al., 2019), DynaHate (Vidgen et al., 2021)) evaluate anonymization's impact on socially critical tasks, where over-redaction may obscure harmful language. ANLI (Nie et al., 2020), an adversarial *natural language inference* (NLI) dataset, assesses whether anonymization preserves logical consistency between premises and hypotheses. MedNLI (Romanov and Shivade, 2018) addresses the medical domain's acute privacy needs, where anonymization must protect patient identities without distorting diagnostic inferences. CLAIR (Radev, 2008) (*fraudulent email detection*) and FakeNews detection[5] examines anonymization's effect on stylistic and structural cues critical for identifying malicious intent and misinformation. We add further details for each task in Appendix A.

For class-balanced datasets (IMDB, DynaSent, ANLI, MedNLI, FRAUD, FakeNews), we report accuracy; for imbalanced tasks (Toxicity, DynaHate), F1 scores are used. All tasks are fine-tuned on ModernBERT-large (Warner et al., 2025), selected for its balance of efficiency and state-of-the-art NLP performance. We also include a random classifier predicting the distribution of target labels as a baseline.

Anonymization Models We benchmark two families of anonymization methods, each of them

[4] https://hf.co/datasets/ai4privacy

[5] https://hf.co/datasets/GonzaloA/fake_news

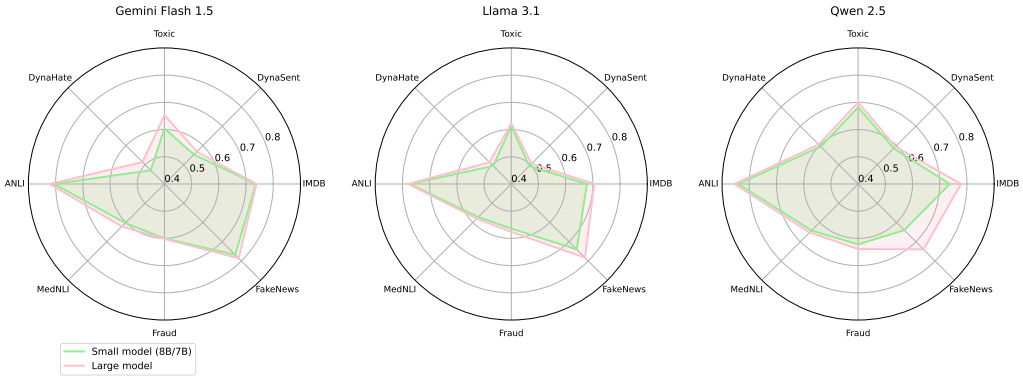


Figure 2: SBERT cosine similarity difference between each tested model and its smaller version.

corresponding to one or the other privacy task. For PII redaction, we evaluate the NER-based Presidio anonymizer (Mendels et al., 2018) as a baseline and compare it against four state-of-the-art LLMs: Gemini Flash 1.5 (Team et al., 2024), Llama-3.1/3.3 (Grattafiori et al., 2024), Qwen-2.5 (Qwen et al., 2025), and Phi-4 (Abdin et al., 2024) prompted for entity replacement. For authorship obfuscation, we test StyleRemix (Fisher et al., 2024), a state-of-the-art baseline, alongside the same LLMs repurposed with style-transfer prompts to disrupt stylistic cues while retaining task-relevant content. Model size effects are analyzed by comparing parameter variants (e.g., Llama-3.1-8B vs. 70B). We access instruction-tuned language models through the OpenRouter API.

5 Results

Table 1 summarizes the performance of anonymization methods across privacy and utility tasks. LLMs emerge as the most effective anonymizers for both PII redaction and authorship obfuscation, achieving best privacy protection. However, this comes at a pronounced cost to utility. Toxicity and hate detection tasks exhibit the steepest performance drops (up to 42% utility decrease for Gemini-flash-1.5 on DynaHate), likely due to LLMs' fine-tuning safeguards against reproducing harmful content. Notably, in the authorship setting, model size inversely correlates with privacy and utility: smaller models tend to do more content modifications to the original text (see Figure 2), leading to better privacy and worse utility. Utility degradation varies dramatically across tasks. Sentiment analysis tasks like IMDB and DynaSent exhibit resilience to anonymization, whereas ANLI, MedNLI and fake news detection degrade significantly. Crucially, no single anonymization method dominates across all tasks, but each familly of models exibit similar performances accross tasks.

Task	BERTScore	METEOR	ROUGE	SBERT	CoLA
IMDB	0.349	0.304	0.349	0.096	0.141
DynaSent	0.735	0.691	0.705	0.735	0.573
Tox	0.681	0.726	0.681	0.516	0.576
DynaHate	0.007	0.051	0.095	0.317	0.051
ANLI	0.627	0.573	0.720	0.352	0.544
MedNLI	0.867	0.764	0.838	0.882	0.705
Fraud	0.518	0.464	0.464	0.597	0.420
FakeNews	0.699	0.759	0.774	0.248	0.323
Average	0.560	0.542	0.578	0.468	0.417

Table 2: Correlation of different metrics with task performance for all utility tasks.

We also performed a correlation analysis between general purpose metrics commonly used in state-of-the-art for utility preservation and task-specific performance measures, computing Kendall's Tau correlation scores as reported in Table 2. The results strongly emphasize the limitations of generic metrics in reliably predicting task utility across different domains. No single metric consistently demonstrates strong correlation with actual utility loss across the evaluated tasks. The sole exception to this pattern was observed in the MedNLI task, where moderate to strong correlations were found. This anomaly can be potentially attributed to the relatively short text size in MedNLI samples, which may result in a more direct relationship between surface-level text modifications and task performance.

6 Related Work

Most existing research addressing similar challenges has primarily focused on evaluating model robustness against sentence-level perturbations. Tools like TextFlint (Wang et al., 2021) offer diverse text noising methods, but emphasize evalu-

ation of downstream task models rather than analyzing the noise mechanisms themselves. PrivKit (Cunha et al., 2024) represents a notable exception in its attempt to establish a comprehensive privacy-utility evaluation pipeline for heterogeneous data types. However, it was primarily designed for location and facial data rather than natural language processing applications and consequently lacks the unified framework necessary to support the diverse requirements of NLP evaluation tasks. Furthermore, PrivKit operates as a standalone system without integration capabilities for popular machine learning frameworks, limiting its accessibility and adoption in existing workflows.

7 Conclusion

In this paper we introduce TAU-EVAL, an evaluation framework enabling unified, reproducible benchmarking of text anonymization systems across both privacy risks and task-specific utility degradation. We study the task sensitivity of language models instructed for anonymization on eight downstream tasks and justify its relevance as a complement to task-agnostic metrics. Our experiments confirm the complexity of achieving a clear trade-off, revealing that no single method dominates across tasks.

Limitations

While our framework advances the evaluation of task-sensitive anonymization, several limitations highlight directions for future work. First, our study operates within a monolingual (English) context. Multilingual anonymization introduces unique challenges (Riabi et al., 2024). Low-resource languages further compound these issues due to sparse PII detection tools and limited benchmark datasets, restricting the generalizability of findings to global contexts.

Second, utility loss measurements depend on the choice of task classifier. While we present evaluations using ModernBERT-large for cross-task comparability, domain-specific architectures (e.g., ClinicalBERT (Huang et al., 2019), CamemBERT-bio (Touchent and de la Clergerie, 2024)) or multilingual models (e.g., XLM-T (Barbieri et al., 2022)) may yield divergent utility trade-offs. For instance, medical anonymization evaluated via a clinically fine-tuned model could reveal subtler impacts on diagnostic inference than our general-purpose setup captures.

Finally, we did implement LLMs mainly using privacy-focused prompts, and could have implemented utility-specific LLMs that perform anonymization when given specific downstream task requirements. Such task-aware anonymization would fundamentally shift the privacy-utility trade-off by limiting data usage to predefined tasks. This presents an interesting direction for future research.

Ethics and Broader Impact Statement

While TAU-EVAL facilitates the benchmarking of anonymization methods, it does not itself perform anonymization or guarantee privacy. Users are responsible for interpreting results in context and for deploying anonymization strategies that align with applicable legal, ethical, and domain-specific standards. We emphasize that TAU-EVAL should not be used as a substitute for legal compliance (e.g., GDPR) but as a research tool to assist in privacy-aware NLP development.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Calvin Bao and Marine Carpuat. 2024. Keep it Private: Unsupervised privatization of online text. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8678–8693, Mexico City, Mexico. Association for Computational Linguistics.

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, Marseille, France. European Language Resources Association.

Iyadh Ben Cheikh Larbi, Aljoscha Burchardt, and Roland Roller. 2023. Clinical text anonymization, its influence on downstream NLP tasks and the risk of re-identification. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 105–111, Dubrovnik, Croatia. Association for Computational Linguistics.

Hanna Berg, Aron Henriksson, and Hercules Dalianis. 2020. The impact of de-identification on downstream named entity recognition in clinical text. In *Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis*, pages 1–11, Online. Association for Computational Linguistics.

cjadams, Daniel Borkan, inversion, Jeffrey Sorensen, Lucas Dixon, Lucy Vasserman, and nithum. 2019. Jigsaw unintended bias in toxicity classification. Kaggle.

Mariana Cunha, Guilherme Duarte, Ricardo Andrade, Ricardo Mendes, and João P. Vilela. 2024. Privkit: A toolkit of privacy-preserving mechanisms for heterogeneous data types. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy*, CODASPY '24, page 319–324, New York, NY, USA. Association for Computing Machinery.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *J. Mach. Learn. Res.*, 22(1).

Jillian Fisher, Skyler Hallinan, Ximing Lu, Mitchell L Gordon, Zaid Harchaoui, and Yejin Choi. 2024. StyleRemix: Interpretable authorship obfuscation via distillation and perturbation of style elements. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4172–4206, Miami, Florida, USA. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and Ahmad Al-Dahle et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *ArXiv*, abs/1904.05342.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

Hemanth Kandula, Damianos Karakos, Haoling Qiu, and Brian Ulicny. 2024. Improving authorship privacy: Adaptive obfuscation with the dynamic selection of techniques. In *Proceedings of the Fifth Workshop on Privacy in Natural Language Processing*, pages 137–142, Bangkok, Thailand. Association for Computational Linguistics.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.

Philippe Laban, Tobias Schnabel, Paul Bennett, and Marti A. Hearst. 2021. Keep it simple: Unsupervised simplification of multi-paragraph 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 6365–6378, Online. Association for Computational Linguistics.

Thomas Lampoltshammer, Lőrinc Thurnay, and Gregor Eibl. 2019. *Impact of Anonymization on Sentiment Analysis of Twitter Postings*, pages 41–48.

Lukas Lange, Heike Adel, and Jannik Strötgen. 2020. Closing the gap: Joint de-identification and concept extraction in the clinical domain. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6945–6952, 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, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Pierre Lison, Ildikó Pilán, David Sánchez, Montserrat Batet, and Lilja Øvrelid. 2021. Anonymisation models for text data: State of the art, challenges and future directions. 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 4188–4203.

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6826–6847, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Gabriel Loiseau, Damien Sileo, Damien Riquet, Maxime Meyer, and Marc Tommasi. 2025. TAROT: Task-oriented authorship obfuscation using policy optimization methods. In *Proceedings of the Sixth Workshop on Privacy in Natural Language Processing*, pages 14–31, Albuquerque, New Mexico. Association for Computational Linguistics.

Cedric Lothritz, Bertrand Lebichot, Kevin Allix, Saad Ezzini, Tegawendé Bissyandé, Jacques Klein, Andrey Boytsov, Clément Lefebvre, and Anne Goujon. 2023. Evaluating the impact of text de-identification on downstream NLP tasks. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 10–16, Tórshavn, Faroe Islands. University of Tartu Library.

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.

Omri Mendels, Coby Peled, Nava Vaisman Levy, Sharon Hart, Tomer Rosenthal, Limor Lahiani, et al. 2018. Microsoft Presidio: Context aware, pluggable and customizable pii anonymization service for text and images.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Constantinos Patsakis and Nikolaos Lykousas. 2023. Man vs the machine in the struggle for effective text anonymisation in the age of large language models. *Scientific Reports*, 13(1):16026.

Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. 2022. The text anonymization benchmark (TAB): A dedicated corpus and evaluation framework for text anonymization. *Computational Linguistics*, 48(4):1053–1101.

Christopher Potts, Zhengxuan Wu, Atticus Geiger, and Douwe Kiela. 2021. DynaSent: A dynamic benchmark for sentiment analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2388–2404, Online. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Dragomir Radev. 2008. Clair collection of fraud email. ACL Data and Code Repository, ADCR2008T001.

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.

Arij Riabi, Menel Mahamdi, Virginie Mouilleron, and Djamé Seddah. 2024. Cloaked classifiers: Pseudonymization strategies on sensitive classification tasks. In *Proceedings of the Fifth Workshop on Privacy in Natural Language Processing*, pages 123–136, Bangkok, Thailand. Association for Computational Linguistics.

Rafael A. Rivera-Soto, Olivia Elizabeth Miano, Juanita Ordonez, Barry Y. Chen, Aleem Khan, Marcus Bishop, and Nicholas Andrews. 2021. Learning universal authorship representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 913–919, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, Brussels, Belgium. Association for Computational Linguistics.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2014. Authorship attribution with topic models. *Computational Linguistics*, 40(2):269–310.

Damien Sileo. 2024. tasksource: A large collection of NLP tasks with a structured dataset preprocessing framework. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15655–15684, Torino, Italia. ELRA and ICCL.

Robin Staab, Mark Vero, Mislav Balunovic, and Martin Vechev. 2024a. Large language models are anonymizers. In *ICLR 2024 Workshop on Reliable and Responsible Foundation Models*.

Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. 2024b. Beyond memorization: Violating privacy via inference with large language models. In *The Twelfth International Conference on Learning Representations*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, and Libin Bai et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Rian Touchent and Éric de la Clergerie. 2024. CamemBERT-bio: Leveraging continual pre-training for cost-effective models on French biomedical data. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 2692–2701, Torino, Italia. ELRA and ICCL.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1667–1682, Online. Association for Computational Linguistics.

Leandro Von Werra, Lewis Tunstall, Abhishek Thakur, Sasha Luccioni, Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, and Helen Ngo. 2022. Evaluate & evaluation on the hub: Better best practices for data and model measurements. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 128–136, Abu Dhabi, UAE. Association for Computational Linguistics.

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, Qinzhuo Wu, Zhengyan Li, Chong Zhang, Ruotian Ma, Zichu Fei, Ruijian Cai, Jun Zhao, Xingwu Hu, Zhiheng Yan, Yiding Tan, Yuan Hu, Qiyuan Bian, Zhihua Liu, Shan Qin, Bolin Zhu, Xiaoyu Xing, Jinlan Fu, Yue Zhang, Minlong Peng, Xiaoqing Zheng, Yaqian Zhou, Zhongyu Wei, Xipeng Qiu, and Xuanjing Huang. 2021. TextFlint: Unified multilingual robustness evaluation toolkit for natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 347–355, Online. Association for Computational Linguistics.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

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.

Eric Xing, Saranya Venkatraman, Thai Le, and Dongwon Lee. 2024. Alison: Fast and effective stylometric authorship obfuscation. In *AAAI*.

Tianyu Yang, Xiaodan Zhu, and Iryna Gurevych. 2025. Robust utility-preserving text anonymization based on large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28922–28941, Vienna, Austria. Association for Computational Linguistics.

Wanyue Zhai, Jonathan Rusert, Zubair Shafiq, and Padmini Srinivasan. 2022. Adversarial authorship attribution for deobfuscation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7372–7384, Dublin, Ireland. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. Pegasus: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Datasets & Task Details

Our evaluation framework employs datasets that span privacy preservation and task-specific utility, carefully selected to reflect real-world anonymization challenges across diverse domains. Below, we elaborate on their structure, relevance, and role in quantifying the privacy-utility trade-off.

pii-masking This synthetic dataset simulates real-world privacy risks by generating text records containing 63 classes of personally identifiable information (PII) (see https://hf.co/ai4privacy), including names, addresses, medical IDs, and financial data. Unlike real-world corpora, synthetic generation avoids ethical concerns while enabling controlled benchmarking of anonymizers' ability to mask sensitive entities. We sample 5,000 English texts from its 400k corpus, ensuring representation of all PII categories. The dataset's structured noise injection (e.g., realistic address formats, contextualized medical terms) tests anonymizers' precision in distinguishing sensitive from non-sensitive content which is a critical capability for GDPR/CCPA compliance.

IMDB62 Derived from the IMDb movie review corpus (Seroussi et al., 2014), this dataset contains 62,000 texts from 62 distinct authors, equally distributed. We focus on a 10-class subset (first 10 authors) to evaluate authorship obfuscation. The task challenges anonymizers to disrupt stylistic fingerprints (e.g., syntactic patterns, lexical preferences).

IMDB (Maas et al., 2011): A classic binary sentiment classification task (positive/negative) on 50k movie reviews. Its balanced distribution and informal language (e.g., user-generated reviews) test anonymization's impact on opinion-driven text utility.

DynaSent (Potts et al., 2021): A ternary sentiment dataset (positive/neutral/negative) constructed via the Dynabench platform (Kiela et al., 2021), where adversarial examples are iteratively generated to exploit model weaknesses. DynaSent's complexity evaluates whether anonymization exacerbates or mitigates robustness to subtle linguistic perturbations.

Toxicity (cjadams et al., 2019): A binary classification task identifying toxic language ("rude, disrespectful, or disruptive" content) in online comments. The dataset's inherent class imbalance ($\approx 10\%$ toxic) tests anonymization's impact on minority class performance and harmful content detection, a critical consideration for moderation systems.

DynaHate (Vidgen et al., 2021): Built using Dynabench's adversarial framework, this dataset targets hate speech detection with examples designed to bypass automated filters. Its adversarial nature stresses anonymization's ability to preserve subtle hate indicators (e.g., dog whistles, coded language) while removing identifiers.

ANLI (Nie et al., 2020): An adversarial natural language inference (NLI) dataset where premises and hypotheses are iteratively refined to challenge model reasoning. By anonymizing both text spans, we test consistency in logical inference: a measure of whether anonymization disrupts semantic relationships critical for NLI.

MedNLI (Romanov and Shivade, 2018): A medical NLI dataset derived from clinical notes, requiring models to infer entailment/contradiction relationships between patient descriptions and hypotheses. Anonymization here risks altering clinically relevant entities (e.g., medications, symptoms), making it a benchmark for domain-specific utility preservation.

Fraud (Radev, 2008): A corpus of fraudulent emails spanning 1988–present, where anonymization must balance redacting PII (e.g., names, account numbers) with preserving stylistic cues (e.g., urgency markers, grammatical errors) indicative of fraud.

Fake News : A binary classification task on 45k news articles (sourced from https://hf.co/datasets/GonzaloA/fake_news) that evaluates anonymization's impact on semantic coherence in misinformation contexts. Articles often contain named entities (e.g., politicians, organizations) whose anonymization may alter the perceived credibility of claims.

B Motivations from Related Work

This section presents a concise overview of the background and related work on evaluating the utility of text anonymization, providing the foundation for the motivations of this paper.

Prior work on text anonymization utility has been most extensive in the medical domain (Lange et al., 2020; Ben Cheikh Larbi et al., 2023), where protecting Protected Health Information (PHI) is ethically and legally required. Studies show that excessive anonymization can reduce clinical value (Berg et al., 2020). Outside the medical domain, utility assessments are narrower, often limited to sentiment analysis (Yang et al., 2025; Patsakis and Lykousas, 2023; Lampoltshammer et al., 2019). Lothritz et al. (2023) provide a broader evaluation

focused on GLUE tasks (Lothritz et al., 2023) but target name redaction only, overlooking semantic and syntactic effects. Recent findings highlight that pseudonymization impacts tasks differently (Riabi et al., 2024), motivating systematic, multi-task evaluation.

Other domains, such as authorship anonymization, emphasize identity protection and utility is often assessed narrowly through text generation metrics like n-gram preservation and fluency (Xing et al., 2024; Fisher et al., 2024; Kandula et al., 2024; Bao and Carpuat, 2024). The rise of large language models as both privacy attackers (Staab et al., 2024b) and anonymizers (Staab et al., 2024a) underscores the need for comprehensive evaluation protocols that combine task-agnostic and task-specific measures.

C Prompts

Figures 3 and 4 show the prompts used for authorship privacy and PII redaction tasks.

User: `Rewrite the text to significantly alter its style, tone, and word choice while preserving the original meaning. Use figurative or descriptive language, vary sentence structures, adjust tone (e.g., formal to conversational), and employ unique synonyms. Avoid retaining distinctive stylistic markers. Respond only with the transformed text.`

Figure 3: Authorship obfuscation prompt

User: `Anonymize user-submitted text by replacing personally identifiable information (PII) like names, dates, locations, and professions with alternate values. Ensure the rewritten text is natural, coherent, and contextually consistent while preserving the original meaning and tone. Respond only with the transformed text.`

Figure 4: PII redaction prompt

Already Implemented Models
Pseudonymization (Riabi et al., 2024)
– Entity Deletion
– Uniform Placeholder
– Category-Specific Placeholder
– Unique Placeholder per Entity
– Unique Substitute per Entity
Authorship Obfuscation
– StyleRemix (Fisher et al., 2024)
– Simplification (Laban et al., 2021)
– Back Translation (Fan et al., 2021)
– Paraphrasing (Zhang et al., 2020a)
[+] *preconfigured templates for HF models & API-based LLMs*
Already Implemented Metrics
Reference-based
– ROUGE (Lin, 2004)
– METEOR (Banerjee and Lavie, 2005)
– BERTScore (Zhang et al., 2020b)
– NLI (Liu et al., 2022)
– LUAR similarity (Rivera-Soto et al., 2021)
– SBERT similarity (Reimers and Gurevych, 2019)
Reference-less
– CoLA (Warstadt et al., 2019)
– Perplexity (Jelinek et al., 1977)
[+] *preconfigured templates for HF evaluate metrics*
Already Implemented Tasks
– De-identification
– IMDB Authorship Attribution
– Text Anonymization Benchmark (Pilán et al., 2022)
– Sequence Classification
– Token Classification
– Multiple Choice
– Sequence-to-Sequence
[+] *preconfigured templates for tasksource tasks*

Table 3: Overview of already implemented models, metrics, and tasks at the time of submission (July 2025).

ViDove: A Translation Agent System with Multimodal Context and Memory-Augmented Reasoning

Yichen Lu[1,2*†], Wei Dai[1,4*†], Jiaen Liu[1,8*†], Ching Wing Kwok [1,3*], Zongheng Wu[1,5*], Xudong Xiao[1,7], Ao Sun[9], Sheng Fu[1], Jianyuan Zhan [7], Yian Wang [6], Takatomo Saito[2], Sicheng Lai [1†],

[1]Pigeon AI, [2]Carnegie Mellon University, [3]Fudan University,

[4]University of California San Diego, [5]University of Toronto, [6]University of California Irvine,

[7]University of Illinois Urbana-Champaign, [8]Institute of Science Tokyo,

[9]Hong Kong University of Science and Technology

Abstract

LLM-based translation agents have achieved highly human-like translation results and are capable of handling longer and more complex contexts with greater efficiency. However, they are typically limited to text-only inputs. In this paper, we introduce **ViDove**, a translation agent system designed for multimodal input. Inspired by the workflow of human translators, ViDove leverages visual and contextual background information to enhance the translation process. Additionally, we integrate a multimodal memory system and long-short term memory modules enriched with domain-specific knowledge, enabling the agent to perform more accurately and adaptively in real-world scenarios. As a result, ViDove achieves significantly higher translation quality in both subtitle generation and general translation tasks, with a 28% improvement in BLEU scores and a 15% improvement in SubER compared to previous state-of-the-art baselines. Moreover, we introduce **DoveBench**, a new benchmark for long-form automatic video subtitling and translation, featuring 17 hours of high-quality, human-annotated data. Our project is shown at https://github.com/pigeonai-org/ViDove.

1 Introduction

Recent advances in Large Language Models (LLMs) have demonstrated remarkable capabilities in Machine Translation (MT) tasks (Robinson et al., 2023; Gao et al., 2023a; Xu et al., 2024a; Zhu et al., 2024). The integration of autonomous agent frameworks with LLM-based MT has shown promising results in enhancing translation capabilities, pushing modern translation systems closer to human-level professional performance (Wu et al., 2024a; Wang et al., 2025a; Guo et al., 2024a; Peter et al., 2024). Through the incorporation of long-short memory system and multi-agent strategies, these approaches have achieved significant improvements in both translation quality and efficiency, enabling LLM-based MT to handle document-level translation effectively (Wang et al., 2025a).

Professional human translators often rely on more than just text to ensure accurate translations. For example, in a cooking video, "fold" could mean combining ingredients or folding a napkin—visual cues like stirring motions and ingredients clarify meaning, while audio tone and emphasis convey intent and emotion (Sulubacak et al., 2019; Shen et al., 2024; et al., 2025). While some Multimodal Machine Translation(MMT) studies incorporate limited visual or audio inputs, they typically cannot handle document-level translation or take entire video as input(Lu et al., 2025; Lv et al., 2025). However, most existing LLM-based MT systems focus solely on textual input, missing out on these valuable contextual signals.

To close the gap between LLM-based translation and professional human performance, we present **ViDove**, a multimodal translation agent that integrates visual, audio, and textual inputs. Built on Retrieval-Augmented Generation (RAG)(Arslan et al., 2024; Abootorabi et al., 2025; Zhai, 2024) and recent Multimodal LLMs(MLLMs) advances(Lu et al., 2024a; Xu et al., 2025; Lu et al., 2024b; Chu et al., 2024), ViDove uses a memory system for domain-specific knowledge, multimodal context, and instruction customization (Long et al., 2024; Ding et al., 2024). Specialized agents handle these inputs to produce more accurate, human-like translations and subtitles. ViDove achieves state-of-the-art results, with a **28%** BLEU and **15%** SubER improvement over baselines. We also introduce **DoveBench**, a video automatic subtitling and translation benchmark with 17 hours of high-quality human-annotated subtitles to support future research.

The key innovations of our work include:

[*]Equal Contribution

[†]Project Lead

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 228–243

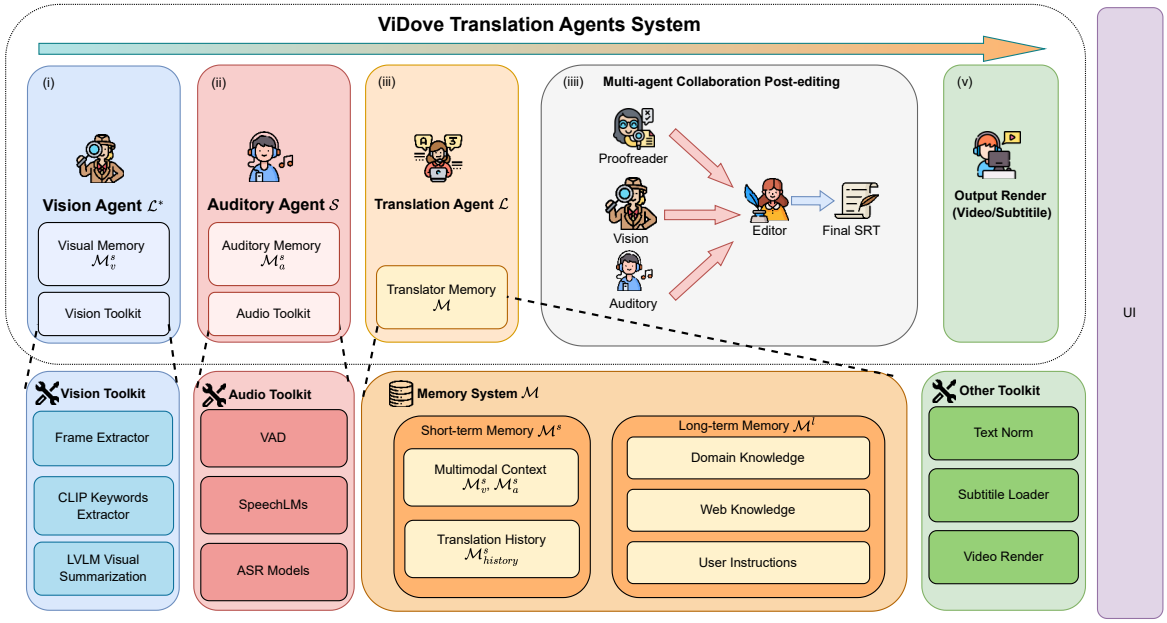


Figure 1: **Architecture of the ViDove Translation Agents System.** The system consists of five modules: (i) Vision Agent $\mathcal{L}^*$ and (ii) Auditory Agent $\mathcal{S}$ extract multimodal cues; (iii) Translation Agent $\mathcal{L}$ utilizes memory $\mathcal{M} = \{\mathcal{M}^s, \mathcal{M}^l\}$ for context-aware translation; (iv) a multi-agent post-editing module refines the output via collaboration; (v) Output Render generates final subtitles and video.

- **Multimodal Multi-Agent Collaboration.** A modular translation agent system that simulates human translator workflows by integrating audio, visual, and textual modalities through specialized agents and their interactions, achieving performance comparable to or better than existing baseline systems.
- **Memory-Augmented Reasoning.** A long-short term memory system for managing multimodal and domain-specific knowledge during translation.
- **DoveBench.** A long-form video automatic subtitling and translation benchmark that reflects real-world subtitling challenges.

2 Related Works

ViDove is a multimodal translation agent framework that enables cooperative translation through multimodal grounding and memory-guided reasoning. Our work builds on two main areas: multimodal machine translation systems and multi-agent, whose integration remains underexplored.

Machine Translation: Traditional MT systems (Wu et al., 2016; Team and et al., 2022) perform well at the sentence level but struggle with limited contextual cues, particularly in multimodal scenarios. To address this, prior studies (Li et al., 2022; Zuo et al., 2023; Lin et al., 2020; Lan et al., 2023) have introduced visual grounding, yet remain constrained to sentence-level tasks with limited context handling. LLM-based MT (Robinson et al., 2023; Hendy et al., 2023; He et al., 2024; Jiao et al., 2023) achieves strong performance by leveraging longer context, but typically treats LLMs as black-box translators rather than reasoning agents with human-like interpretive capabilities.

RAG for Machine Translation: Recent studies have increasingly leveraged RAG to enhance MT. Some works applied RAG to improve the quality and cultural grounding of MT by utilizing diverse retrieval pipelines, knowledge graphs, and multi-task fine-tuning setups (Bouthors et al., 2024; Conia et al., 2024; Wang et al., 2024; Anonymous, 2024). Concurrently, other works have specifically addressed low-resource languages, showing significant gains by augmenting prompts with retrieved bilingual dictionaries and example sentences (Merx et al., 2024; Chang et al., 2025). ViDove effectively combines the strengths of these approaches by architecting a multi-agent system where RAG serves as the core information storage and exchange protocol.

Multi-Agent Systems and Translation Agent: Recent work on multi-agent LLM systems (Li et al., 2023a; Hu et al., 2021; Guo et al., 2024b; Cheng

et al., 2024; Li et al., 2023c; Ma et al., 2024; Wang et al., 2023; Xu et al., 2024b) shows that dividing complex tasks among specialized agents with distinct roles enhances reasoning and decision-making. These agents collaborate through structured coordination, often leveraging tools (Li et al., 2023b; Ruan et al., 2023; Wu et al., 2023) and memory systems (Ding et al., 2023; Singhal et al., 2023a,b) to maintain context and task knowledge.

This paradigm has recently been explored in machine translation. For example, TransAgents (Wu et al., 2024a) improves translation quality through multi-agent critique, where agents evaluate and refine each other's outputs. DelTA (Wang et al., 2025b) ensures document-level consistency using memory modules. Both approaches enhance translation quality in domain-specific or long-form scenarios.

However, these existing approaches remain confined to the textual modality and overlook the potential of MLLMs (Lu et al., 2024a; Xu et al., 2025; Chu et al., 2024) in enhancing translation quality. To bridge this gap and bring machine translation closer to human-like performance, we propose ViDove, a novel framework that leverages recent advances in both LLM-based agent systems and MLLMs.

3 ViDove

In this section, we first introduce the characteristics of long-form video subtitle generation and translation, then describe each agent in Fig. 1 in detail.

3.1 Preliminary

For clarity, we define our primary notation as follows: $\mathcal{V}$ denotes the input video, $\mathcal{L}$ is the translator Agent, $\mathcal{L}^*$ represents the visual agent, and $\mathcal{S}$ signifies the auditory agent. Long-short term memory modules are represented by $\mathcal{M} = \{\mathcal{M}^s, \mathcal{M}^l\}$, capturing both short-term and long-term contexts. The prompt list P contains guiding prompts for analysis tasks. Video chunks are represented by $\mathcal{C}_i$, each consisting of visual ($\mathcal{V}_i$) and audio ($\mathcal{A}_i$) components. Transcripts are denoted by T_i, with translated transcripts represented by T_i^*. The pipeline framework can be formulated as algorithm 1.

3.2 Long-form Video Automatic Subtitling

Long-form(>10min) video automatic subtitling requires the system to process a video $\mathcal{V}$, which includes a synchronized audio stream $\mathcal{A}$. Unlike typical sentence-level translation, this task operates at the document level and demands precise alignment of translated subtitles with the corresponding timestamps. Moreover, in contrast to standard document-level MT, the input does not contain any original text. Instead, the system must first perceive the video by "listening" to the audio—and, when necessary, "observing" the visual content—to transcribe the source language before translating it. These multimodal and multi-stage requirements—speech recognition, visual grounding, and context-aware translation—make the task considerably more complex than traditional MT, multimodal MT, or document-level MT (DocMT). Addressing this challenge calls for a holistic, agent-based approach capable of perception, reasoning, and translation.

Algorithm 1 ViDove

Require: Input video $\mathcal{V}$, Multi-agent translation agent $\mathcal{L}$ as in Algorithm 2, Visual Agent $\mathcal{L}^*$, auditory agent $\mathcal{S}$, Long-short term memory $\mathcal{M} = \{\mathcal{M}^s, \mathcal{M}^l\}$, prompt list P

Ensure: Translated video, original and translated transcript tuple (V^*, T^*, T)

Initialize: $\mathcal{M}^s \leftarrow \emptyset$, $\mathcal{M}^l \leftarrow$ Knowledge Base ▷ Initialize memory modules

$\{(\mathcal{V}_i, \mathcal{A}_i)\}_{i=1}^k \leftarrow \mathcal{C}(\mathcal{V})$ ▷ Chunk decomposition

for each chunk $(\mathcal{V}_i, \mathcal{A}_i) \in \mathcal{C}$ **do**

$(\mathcal{M}_v^s, cue_v) \leftarrow \mathcal{L}^*(\mathcal{V}_i, \mathcal{M}_v^s, \mathcal{M}_{domain}^l, p_{analysis}), \quad p_{analysis} \in P$ ▷ Update visual cues and short-term memory

$cue_a \leftarrow \mathcal{S}(\mathcal{A}_i)$

$T_i \leftarrow (cue_a, cue_v)$

$(T_i^*, \mathcal{M}_{history}^s) \leftarrow \mathcal{L}(T_i, \mathcal{M}, p_{translation})$

end for

$T^*, T \leftarrow$ **Multi-agentPostProcess**$(T^*, T, \mathcal{M}, p_{pr})$ ▷ Final post-processing of translations

return (V^*, T^*, T)

3.3 Auditory Agent

In this section, we describe the workflow of ViDove's auditory agent, which provides audio-based contextual information and enriched transcriptions to support the Translation Agent.

Step 1: Chunk Splitting and Timestamp Extraction. We use Pyannote (Plaquet and Bredin, 2023a) to segment the input video $\mathcal{V}$ into k chunks, $\mathcal{C}$, based on speaker activity. Each chunk $\mathcal{C}_i$ contains a corresponding set of video frames $\mathcal{V}_i$ and an audio sequence $\mathcal{A}_i$.

Step 2: Auditory Information Extraction. ViDove integrates SOTA single-task models to extract key auditory features: speech transcription (Radford et al., 2022), background audio event detection (Chen et al., 2024, 2022), and speaker emotion recognition (Ma et al., 2023). In addition to these, we incorporate recent Speech Language Models(SpeechLMs) (Tang et al., 2024; Chu et al., 2024), which offer a unified approach for extracting rich audio cues. For each chunk, the extracted auditory information, denoted as cue_a, is stored in the multimodal contextual memory $\mathcal{M}_a^s \in \mathcal{M}_{multimodal}^s$ to support downstream translation.

Step 3: Audio Transcription and Timestamp Refinement. ViDove's auditory agent uses either a SpeechLM or an ASR model to transcribe the audio sequence $\mathcal{A}i$, leveraging the multimodal contextual memory $[\mathcal{M}_a^s, \mathcal{M}_v^s] \in \mathcal{M}_{multimodal}^s$. This fusion enables enhanced transcription through keyword injection and agent-based reasoning.

3.4 Vision Agent

ViDove implements a visual agent to gather informative visual cues from video input $\mathcal{V}$. These cues are used to enhance speech recognition (Lu et al., 2024a; Wu et al., 2024b) and are passed through a memory system to support more sophisticated sentence comprehension. This visual cue allows the agent to resolve textually ambiguousness and accurately interpret domain-specific terminology during translation. The ViDove system is designed to be model-agnostic and supports multiple vision-language backends, offering deployment flexibility across local and remote environments.

The pipeline is designed to process pre-segmented video chunks

$$\mathcal{V} = \{\mathcal{V}_0, \mathcal{V}_1, ..., \mathcal{V}_k\}, k = |V|$$

, from which key frames are extracted to represent salient visual moments. These frames are then analyzed by the selected vision-language model $\mathcal{L}^*$, providing a high-level semantic understanding cue_v of the visual context.

$$cue_v = \mathcal{L}^*(v_i, \mathcal{M}_{vision}^s, M_{domain}^l, p_{analysis})$$

Where $\mathcal{M}$ is multi-modal memory for ViDove system.

3.5 Memory system

ViDove's memory system, denoted as $\mathcal{M} = \{\mathcal{M}^s, \mathcal{M}^l\}$, is implemented using LLaMA-Index(Liu, 2022). This memory system stores and organizes multimodal information, enabling the system to deliver contextually informed and consistent translations.

3.5.1 Short-term Memory ($\mathcal{M}^s$)

The short-term memory ($\mathcal{M}^s$) is tailored to the current video translation task. It holds information specific to each video chunk ($\mathcal{C}_i$), which includes visual ($\mathcal{V}_i$) and audio ($\mathcal{A}_i$) components. Its contents include:

Translation History: Records of prior translations within the same video, ensuring consistency in terminology and phrasing across transcripts (e.g., from T_i to T_i^*).

Visual Cues: Contextual data extracted from $\mathcal{V}_i$, such as scene descriptions or objects, that helps to disambiguate textual content.

Audio Cues: Auditory information extracted from $\mathcal{A}_i$, denoted as cue_a, including transcriptions and other speaker information, stored in the multimodal contextual memory $\mathcal{M}_a^s \in \mathcal{M}_{multimodal}^s$.

This component provides immediate context, supporting accurate and coherent translations within a single video.

3.5.2 Long-term Memory ($\mathcal{M}^l$)

The long-term memory ($\mathcal{M}^l$) is designed for adaptability across multiple translation tasks. It accumulates knowledge over time, storing:

Domain Knowledge: This type of knowledge captures specialized community language to ensure accurate video translations for a diverse, multilingual audience.

Web Knowledge: General information sourced from the web, implemented by Tavily(Tavily AI, 2025), offering broader context.

This component enhances ViDove's flexibility, enabling it to handle diverse domains and improve performance over time. The memory system acts as a centralized repository, providing essential information that supports the translation process by ensuring consistency and contextual relevance.

3.6 Multi-agent Translation

ViDove's translation process relies on a multi-agent system featuring three specialized agents—*Translator*, *Proofreader*, and *Editor*—that work together to produce high-quality subtitle translations. These agents collaborate by accessing the unified memory system $\mathcal{M} = \{\mathcal{M}^s, \mathcal{M}^l\}$, ensuring consistency and context throughout the workflow.

	DoveBench				BigVideo	
	BLEU(↑)	BLEURT(↑)	SubER(↓)	SubSONAR(↑)	BLEU(↑)	sCOMET(↑)
Gemini-2.5-Flash	8.11	17.21	103.46	0.31	26.43	0.75
Qwen-2.5-Omni	14.60	13.83	108.94	0.39	10.67	0.58
VideoCaptioner	12.65	14.62	85.75	**0.41**	**30.36**	**0.75**
Whisper + DelTA	18.26	12.30	86.83	0.28	29.09	0.69
ViDove	**23.51**	**19.55**	**73.38**	0.39	26.05	0.73

Table 1: ViDove compared with different baseline system on DoveBench and BigVideo.

Agents and Their Roles

Translator Agent ($\mathcal{L}_t$): This agent generates the initial translation (T_i^*). To achieve this, it interacts with the memory system by retrieving *translation history* and *visual cues* from short-term memory ($\mathcal{M}^s$) to maintain intra-video consistency, while simultaneously drawing upon *domain knowledge* from long-term memory ($\mathcal{M}^l$) to ensure domain-specific accuracy and stylistic alignment from the outset.

Proofreader Agent ($\mathcal{L}_{pr}$): Proofreader agent focuses on refining the initial translation by correcting grammar, style, and terminology. It interacts with memory by accessing *domain knowledge* from long-term memory ($\mathcal{M}^l$) to ensure linguistic precision and adherence to specialized terminology, while referencing the *translation history* in short-term memory ($\mathcal{M}^s$) to maintain consistency with prior segments. The proofreader agent generates revision suggestions for the editor agent, which makes the final decision on whether to apply them. A sample interaction log is provided in Appendix A.1.1.

Editor Agent ($\mathcal{L}_{ed}$): The editor agent performs the final quality check and applies necessary modifications to ensure the translation quality with the full modality context. It receives suggestions from the proofreader agent and decides whether to adopt them. It also accepts user instructions, enabling more free-form and practical interactions between the user and agents. To verify contextual accuracy, the editor agent leverages the memory system by retrieving *visual cues*, *audio cues*, and *translation history* from the short-term memory ($\mathcal{M}^s$). It also queries *web knowledge* from the long-term memory ($\mathcal{M}^l$) to incorporate broader external context and ensure logical consistency.

4 DoveBench

To the best of our knowledge, there is currently no standardized open-sourced benchmark for long-form video automatic subtitling (Sec 3.2). To address this gap, we introduce **DoveBench**, an open-source benchmark designed specifically for this task. DoveBench contains approximately 17 hours of video data, each annotated with Chinese (ZH) subtitles translated by professional translators. The average video length is around 20 minutes, reflecting typical durations found in real-world scenarios. Detailed statistics of DoveBench are provided in Appendix B.

5 Experiments

In this section, we first describe the evaluation datasets, baseline systems, metrics, and ViDove's configuration. We then present and analyze experiment results.

5.1 Datasets and Metrics

We evaluate long-form video automatic subtitling on DoveBench and MMT on BigVideo (Kang et al., 2023). To assess translation quality, we use BLEU (Freitag et al., 2020), BLEURT (Sellam et al., 2020), and sCOMET (Rei et al., 2020). For subtitle quality—capturing both translation accuracy and timestamp alignment—we adopt SubER (Wilken et al., 2022) and SubSONAR (Gaido et al., 2024), specifically for evaluating on DoveBench.

5.2 Baselines and ViDove Configuration

We compare ViDove against four baseline systems. Gemini-2.5-flash (Google DeepMind and Google Research, 2025) serves as a proprietary baseline, and Qwen-2.5-Omni (Xu et al., 2025) represents an open-source alternative. Both models are single MLLMs capable of processing video input

and generating subtitle (SRT) output through carefully designed prompts. For system-level baselines, we include VideoCaptioner (Weifeng2333, 2024), an open-source cascaded pipeline for video subtitling, and DelTA (Wang et al., 2025a), a state-of-the-art text-based translation agent. Since DelTA does not support audio or video input natively, we use `whisper-large-v3` (Radford et al., 2022) to first transcribe the audio for the system. For ViDove, we use Gemini-2.5-flash (Google DeepMind and Google Research, 2025) as the auditory agent, while all other agents are powered by GPT-4o (OpenAI, 2024). Note that neither the baseline models nor ViDove undergo any additional fine-tuning during the experiments. Detailed prompts for the baseline models and ViDove are provided in Appendix C.1, C.2 and A.3.

5.3 Experiment Results

Table 1 presents the evaluation results of ViDove and baseline systems on both DoveBench and BigVideo.

On DoveBench, ViDove consistently outperforms all baselines across all metrics, achieving the highest BLEU (23.51), BLEURT (19.55), and the lowest SubER (73.38). Compared to the strongest baseline, Whisper + DelTA, ViDove improves BLEU by 28.8%, BLEURT by 58.9%, and reduces SubER by 15.5%. These results indicate that ViDove not only produces more accurate translations but also aligns subtitles more precisely in time. However, despite ViDove's leading performance, the absolute scores—especially BLEU and SubER—remain relatively modest. This highlights the intrinsic difficulty of long-form video automatic subtitling. To date, no existing system has achieved fully satisfactory results on this task, underscoring its complexity and open research nature.

In contrast, Gemini-2.5-flash and Qwen-2.5-Omni perform poorly on DoveBench, with high SubER values (103.46 and 108.94, respectively) and low BLEU scores. Although we carefully engineered prompts and applied post-processing to ensure fair evaluation, these single-model MLLMs struggle on long-form inputs. Their limited instruction-following capability, combined with a tendency to hallucinate or ignore constraints as input length increases, leads to misaligned, incomplete, or off-topic subtitles—even when the task is clearly specified.

On BigVideo, which focuses on sentence-level MMT, ViDove remains competitive. While it is not specifically optimized for short-form translation tasks, it achieves BLEU (26.05) and sCOMET (0.73) scores close to the best-performing models, such as Gemini-2.5-flash and VideoCaptioner. This demonstrates ViDove's generalizability and robustness.

Model	BLEU (↑)	SubER (↓)	BLEURT (↑)
ViDove (full)	**15.84**	**76.26**	17.11
w/o domain memory	14.86	77.31	**17.84**
w/o domain memory & vision	14.56	77.55	17.50
w/o Proofreader	13.56	80.76	16.93

Table 2: Ablation study of ViDove under single-column setting.

5.4 Ablation Study

To assess the contribution of different components in ViDove, we conduct an ablation study by removing modules. Our ablation study is conducted on a subset of DoveBench's StarCraft 2 Category. As shown in Table 2, removing the domain memory significantly reduces BLEU and SubER scores, though BLEURT slightly improves—likely due to more generic paraphrasing. Excluding the proofreader agent causes the sharpest quality drop, highlighting its role in correction and consistency. While the visual module has limited impact on BLEU or BLEURT, it helps the editor correct entity-level terms (e.g., names and objects), improving factual accuracy and user experience beyond what metrics capture.

6 Conclusion

In this work, we introduced **ViDove**, a multimodal translation agent system for long-form video inputs. Our model outperforms the strongest existing baselines by up to 28.8% in BLEU and 15.5% in SubER, demonstrating significant improvements in both translation accuracy and subtitle alignment. We also release a new benchmark for the challenging task of long-form video automatic subtitling. Compared to prior work, ViDove offers a more practical and scalable solution for video automatic subtitling and translation.

Acknowledgments

We thank the FGA Subtitle Group, MetricSubs, and Star-Pigeon Group for their support in providing high-quality human-annotated datasets. This work was supported in part by funding from Pigeon AI.

References

Mohammad Mahdi Abootorabi, Amirhosein Zobeiri, Mahdi Dehghani, Mohammadali Mohammadkhani, Bardia Mohammadi, Omid Ghahroodi, Mahdieh Soleymani Baghshah, and Ehsaneddin Asgari. 2025. Ask in any modality: A comprehensive survey on multimodal retrieval-augmented generation. *Preprint*, arXiv:2502.08826.

Anonymous. 2024. RAG picking helps: Retrieval augmented generation for machine translation. In *Submitted to ACL Rolling Review - August 2024*. Under review.

Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. 2024. A survey on rag with llms. *Procedia Computer Science*, 246:3781–3790.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Maxime Bouthors, Josep Crego, and Francois Yvon. 2024. Retrieving examples from memory for retrieval augmented neural machine translation: A systematic comparison. *Preprint*, arXiv:2404.02835.

Chen-Chi Chang, Chong-Fu Li, Chu-Hsuan Lee, and Hung-Shin Lee. 2025. Enhancing low-resource minority language translation with llms and retrieval-augmented generation for cultural nuances. *Preprint*, arXiv:2505.10829.

Sanyuan Chen, Yu Wu, Chengyi Wang, Shujie Liu, Daniel Tompkins, Zhuo Chen, and Furu Wei. 2022. Beats: Audio pre-training with acoustic tokenizers. *Preprint*, arXiv:2212.09058.

Wenxi Chen, Yuzhe Liang, Ziyang Ma, Zhisheng Zheng, and Xie Chen. 2024. Eat: Self-supervised pre-training with efficient audio transformer. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 3807–3815. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, and Xiuqiang He. 2024. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *Preprint*, arXiv:2401.03428.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, Chang Zhou, and Jingren Zhou. 2024. Qwen2-audio technical report. *Preprint*, arXiv:2407.10759.

Simone Conia, Daniel Lee, Min Li, Umar Farooq Minhas, Saloni Potdar, and Yunyao Li. 2024. Towards cross-cultural machine translation with retrieval-augmented generation from multilingual knowledge graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16343–16360, Miami, Florida, USA. Association for Computational Linguistics.

Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. 2023. Integrating action knowledge and llms for task planning and situation handling in open worlds. *Autonomous Robots*, 47(8):981–997.

Yihao Ding, Kaixuan Ren, Jiabin Huang, Siwen Luo, and Soyeon Caren Han. 2024. Pdf-mvqa: A dataset for multimodal information retrieval in pdf-based visual question answering. *Preprint*, arXiv:2404.12720.

Aaron Grattafiori et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Alec Radford et al. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

Emilio Villa-Cueva et al. 2025. Cammt: Benchmarking culturally aware multimodal machine translation. *Preprint*, arXiv:2505.24456.

Markus Freitag, David Grangier, and Isaac Caswell. 2020. BLEU might be guilty but references are not innocent. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 61–71, Online. Association for Computational Linguistics.

Marco Gaido, Sara Papi, Matteo Negri, Mauro Cettolo, and Luisa Bentivogli. 2024. SBAAM! Eliminating Transcript Dependency in Automatic Subtitling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Bangkok, Thailand.

Yuan Gao, Ruili Wang, and Feng Hou. 2023a. How to design translation prompts for chatgpt: An empirical study. *Preprint*, arXiv:2304.02182.

Zhifu Gao, Shiliang Zhang, Ian McLoughlin, and Zhijie Yan. 2023b. Paraformer: Fast and accurate parallel transformer for non-autoregressive end-to-end speech recognition. *Preprint*, arXiv:2206.08317.

Google. 2024. Google translate. `https://translate.google.com`. Accessed: 2024-07-05.

Google DeepMind and Google Research. 2025. Gemini 2.5 flash. Model card, Google AI Studio / Vertex AI. Enhanced multimodal large language model with extended capabilities in text, image, audio, and video.

Shoutao Guo, Shaolei Zhang, Zhengrui Ma, Min Zhang, and Yang Feng. 2024a. Agent-simt: Agent-assisted simultaneous machine translation with large language models. *Preprint*, arXiv:2406.06910.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024b. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8048–8057. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Zhiwei He, Tian Liang, Wenxiang Jiao, Zhuosheng Zhang, Yujiu Yang, Rui Wang, Zhaopeng Tu, Shuming Shi, and Xing Wang. 2024. Exploring human-like translation strategy with large language models. *Transactions of the Association for Computational Linguistics*, 12:229–246.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *Preprint*, arXiv:2302.09210.

Junyan Hu, Parijat Bhowmick, Inmo Jang, Farshad Arvin, and Alexander Lanzon. 2021. A decentralized cluster formation containment framework for multirobot systems. *IEEE Transactions on Robotics*, 37(6):1936–1955.

Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine. *Preprint*, arXiv:2301.08745.

Liyan Kang, Luyang Huang, Ningxin Peng, Peihao Zhu, Zewei Sun, Shanbo Cheng, Mingxuan Wang, Degen Huang, and Jinsong Su. 2023. BigVideo: A large-scale video subtitle translation dataset for multimodal machine translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8456–8473, Toronto, Canada. Association for Computational Linguistics.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment anything. *Preprint*, arXiv:2304.02643.

Zhibin Lan, Jiawei Yu, Xiang Li, Wen Zhang, Jian Luan, Bin Wang, Degen Huang, and Jinsong Su. 2023. Exploring better text image translation with multimodal codebook. *Preprint*, arXiv:2305.17415.

Bei Li, Chuanhao Lv, Zefan Zhou, Tao Zhou, Tong Xiao, Anxiang Ma, and JingBo Zhu. 2022. On vision features in multimodal machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6327–6337, Dublin, Ireland. Association for Computational Linguistics.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023b. Api-bank: A comprehensive benchmark for tool-augmented llms. *Preprint*, arXiv:2304.08244.

Wenhao Li, Dan Qiao, Baoxiang Wang, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. 2023c. Semantically aligned task decomposition in multi-agent reinforcement learning. *Preprint*, arXiv:2305.10865.

Huan Lin, Fandong Meng, Jinsong Su, Yongjing Yin, Zhengyuan Yang, Yubin Ge, Jie Zhou, and Jiebo Luo. 2020. Dynamic context-guided capsule network for multimodal machine translation. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20. ACM.

Jerry Liu. 2022. LlamaIndex.

Xinwei Long, Jiali Zeng, Fandong Meng, Zhiyuan Ma, Kaiyan Zhang, Bowen Zhou, and Jie Zhou. 2024. Generative multi-modal knowledge retrieval with large language models. *Preprint*, arXiv:2401.08206.

Chenyu Lu, Shiliang Sun, Jing Zhao, Nan Zhang, Tengfei Song, and Hao Yang. 2025. Multimodal machine translation with visual scene graph pruning. *Preprint*, arXiv:2505.19507.

Yichen Lu, Jiaqi Song, Xuankai Chang, Hengwei Bian, Soumi Maiti, and Shinji Watanabe. 2024a. Syneslm: A unified approach for audio-visual speech recognition and translation via language model and synthetic data. *Preprint*, arXiv:2408.00624.

Yichen Lu, Jiaqi Song, Chao-Han Huck Yang, and Shinji Watanabe. 2024b. FastAdaSP: Multitask-adapted efficient inference for large speech language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 440–451, Miami, Florida, US. Association for Computational Linguistics.

Jinze Lv, Jian Chen, Zi Long, Xianghua Fu, and Yin Chen. 2025. Topicvd: A topic-based dataset of video-guided multimodal machine translation for documentaries. *Preprint*, arXiv:2505.05714.

Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. 2024. Large language models play starcraft ii:benchmarks and a chain of summarization approach. In *Advances in Neural Information Processing Systems*, volume 37, pages 133386–133442. Curran Associates, Inc.

Ziyang Ma, Zhisheng Zheng, Jiaxin Ye, Jinchao Li, Zhifu Gao, Shiliang Zhang, and Xie Chen. 2023. emotion2vec: Self-supervised pre-training for speech emotion representation. *Preprint*, arXiv:2312.15185.

Raphaël Merx, Aso Mahmudi, Katrina Langford, Leo Alberto de Araujo, and Ekaterina Vylomova. 2024. Low-resource machine translation through retrieval-augmented llm prompting: A study on the mambai language. *Preprint*, arXiv:2404.04809.

et al. OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Anishka Peter, Mai Dang, Michael Liu, Joaquin Dominguez, and Nibhrat Lohia. 2024. Multi-agent translation team (matt): Enhancing low-resource language translation through multi-agent workflow. *SMU Data Science Review*, 8(3):3.

Alexis Plaquet and Hervé Bredin. 2023a. Powerset multi-class cross entropy loss for neural speaker diarization. In *Proc. INTERSPEECH 2023*.

Alexis Plaquet and Hervé Bredin. 2023b. Powerset multi-class cross entropy loss for neural speaker diarization. In *INTERSPEECH 2023*, interspeech_2023. ISCA.

Qwen, :, and An Yang et al. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *Preprint*, arXiv:2212.04356.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Nathaniel Robinson, Perez Ogayo, David R. Mortensen, and Graham Neubig. 2023. ChatGPT MT: Competitive for high- (but not low-) resource languages. In *Proceedings of the Eighth Conference on Machine Translation*, pages 392–418, Singapore. Association for Computational Linguistics.

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, and Rui Zhao. 2023. Tptu: Large language model-based ai agents for task planning and tool usage. *Preprint*, arXiv:2308.03427.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*.

Huangjun Shen, Liangying Shao, Wenbo Li, Zhibin Lan, Zhanyu Liu, and Jinsong Su. 2024. A survey on multi-modal machine translation: Tasks, methods and challenges. *Preprint*, arXiv:2405.12669.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, Blaise Agüera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. 2023a. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, Nenad Tomasev, Yun Liu, Renee Wong, Christopher Semturs, S. Sara Mahdavi, Joelle Barral, Dale Webster, Greg S. Corrado, Yossi Matias, Shekoofeh Azizi, Alan Karthikesalingam, and Vivek Natarajan. 2023b. Towards expert-level medical question answering with large language models. *Preprint*, arXiv:2305.09617.

Umut Sulubacak, Ozan Caglayan, Stig-Arne Grönroos, Aku Rouhe, Desmond Elliott, Lucia Specia, and Jörg Tiedemann. 2019. Multimodal machine translation through visuals and speech. *Preprint*, arXiv:1911.12798.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. 2024. Salmonn: Towards generic hearing abilities for large language models. *Preprint*, arXiv:2310.13289.

Tavily AI. 2025. Tavily. Accessed: 2025-07-05.

NLLB Team and et al. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Suramya Tomar. 2006. Converting video formats with ffmpeg. *Linux journal*, 2006(146):10.

Jiaan Wang, Fandong Meng, Yingxue Zhang, and Jie Zhou. 2024. Retrieval-augmented machine translation with unstructured knowledge. *Preprint*, arXiv:2412.04342.

Shenzhi Wang, Chang Liu, Zilong Zheng, Siyuan Qi, Shuo Chen, Qisen Yang, Andrew Zhao, Chaofei Wang, Shiji Song, and Gao Huang. 2023. Avalon's game of thoughts: Battle against deception through recursive contemplation. *Preprint*, arXiv:2310.01320.

Yutong Wang, Jiali Zeng, Xuebo Liu, Derek F. Wong, Fandong Meng, Jie Zhou, and Min Zhang. 2025a. Delta: An online document-level translation agent based on multi-level memory. *Preprint*, arXiv:2410.08143.

Yutong Wang, Jiali Zeng, Xuebo Liu, Derek F. Wong, Fandong Meng, Jie Zhou, and Min Zhang.

2025b. Delta: An online document-level translation agent based on multi-level memory. *Preprint*, arXiv:2410.08143.

Weifeng2333. 2024. Videocaptioner: An open-source cascaded system for video subtitling. https://github.com/WEIFENG2333/VideoCaptioner. Accessed: 2025-07-04.

Patrick Wilken, Panayota Georgakopoulou, and Evgeny Matusov. 2022. SubER - a metric for automatic evaluation of subtitle quality. In *Proceedings of the 19th International Conference on Spoken Language Translation (IWSLT 2022)*, pages 1–10, Dublin, Ireland (in-person and online). Association for Computational Linguistics.

Minghao Wu, Jiahao Xu, and Longyue Wang. 2024a. TransAgents: Build your translation company with language agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 131–141, Miami, Florida, USA. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Yihan Wu, Yichen Lu, Yifan Peng, Xihua Wang, Ruihua Song, and Shinji Watanabe. 2024b. Enhancing audio-visual speech recognition through bifocal preference optimization. *Preprint*, arXiv:2412.19005.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Preprint*, arXiv:1609.08144.

Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. 2024c. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. *Preprint*, arXiv:2211.06687.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024a. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *Preprint*, arXiv:2401.08417.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Wang, Yunfei Chu, and Junyang Lin. 2025. Qwen2.5-omni technical report. *Preprint*, arXiv:2503.20215.

Peng Xu, Andrea Madotto, Chien-Sheng Wu, Ji Ho Park, and Pascale Fung. 2018. Emo2vec: Learning generalized emotion representation by multi-task training. *Preprint*, arXiv:1809.04505.

Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. 2024b. Language agents with reinforcement learning for strategic play in the werewolf game. *Preprint*, arXiv:2310.18940.

Wenjia Zhai. 2024. Self-adaptive multimodal retrieval-augmented generation. *Preprint*, arXiv:2410.11321.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. *Preprint*, arXiv:2303.15343.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. *Preprint*, arXiv:2304.04675.

Yuxin Zuo, Bei Li, Chuanhao Lv, Tong Zheng, Tong Xiao, and JingBo Zhu. 2023. Incorporating probing signals into multimodal machine translation via visual question-answering pairs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14689–14701, Singapore. Association for Computational Linguistics.

A ViDove Details

A.1 Multi-agent Translation System

Auditory Agent	SpeechLM	SALMONN (Tang et al., 2024), Gemini-2.5-flash (Google DeepMind and Google Research, 2025) Qwen2-Audio (Chu et al., 2024), Qwen-2.5-Omni (Xu et al., 2025),
	ASR	Whisper-series (Radford et al., 2022) Paraformer(Gao et al., 2023b)
	Others	Emo2Vec(Xu et al., 2018), BEATs(Chen et al., 2022), EATs(Chen et al., 2024) CLAP(Wu et al., 2024c) Pyannote(Plaquet and Bredin, 2023b)
Visual Agent	VLMs	GPT4-o(OpenAI, 2024), Qwen2.5-VL(Bai et al., 2025)
	Others	CLIP(et al., 2021), SigCLIP(Zhai et al., 2023), SegAnything(Kirillov et al., 2023)
Translation Agent & Post-editing Team	LLMs	GPT-series(OpenAI, 2024), LLaMA-series(et al., 2024), Qwen-series(Qwen et al., 2025)
	Others	Google-translate(Google, 2024), NLLB(Team and et al., 2022)
Memory System	Web	Tavily(Tavily AI, 2025)
	Local	Llama-index(Liu, 2022)
Other Tools	-	FFmpeg(Tomar, 2006)
Evaluation Metrics	-	BLEU (Freitag et al., 2020), COMET (Rei et al., 2020), BLEURT (Sellam et al., 2020), SubER (Wilken et al., 2022), SubSONAR (Sulubacak et al., 2019)

Table 3: Base models and tools used in ViDove

Algorithm 3 Multi-agent Translation Pipeline

Require: Transcript chunk T_i, Memory $\mathcal{M} = \{\mathcal{M}^s, \mathcal{M}^l\}$, LLM Translator $\mathcal{L}_t$, Proofreader $\mathcal{L}_{pr}$, Editor $\mathcal{L}_{ed}$

Ensure: Translated transcript chunk T_i^*, updated short-term memory $\mathcal{M}^s_{history}$, and translation prompt $p_{translation}$

$translation\_history_{i,i-5} \leftarrow$ **retrieve**$(\mathcal{M}^s_{history}, T_i)$
$context_i \leftarrow$ **retrieve**$(\mathcal{M}^s_{context})$
$domain\_guide \leftarrow$ **query**$(\mathcal{M}^l_{domain}, T_i)$
$p_{translation} \leftarrow (p_{translation}, context_i, domain\_guide)$
$T_i^* \leftarrow \mathcal{L}_t(T_i, p_{translation})$
$T^*_{i,pr} \leftarrow \mathcal{L}_{pr}(T_i^*, \mathcal{M}^s, \mathcal{M}^l)$ ▷ Proofreader checks grammar, style, terminology
$T^*_{i,ed} \leftarrow \mathcal{L}_{ed}(T^*_{i,pr}, \mathcal{M}^s, \mathcal{M}^l)$ ▷ Editor ensures logical and contextual consistency
$\mathcal{M}^s_{history} \leftarrow (\mathcal{M}^s, T_i, T^*_{i,ed})$
return $(T^*_{i,ed}, \mathcal{M}^s_{history})$

A.1.1 Proofreader Agent

To further demonstrate the role of the proofreader agent in our pipeline, we present a series of log messages captured during a real session. The proofreader monitors intermediate translations and provides suggestions to correct terminology, sentence structure, and domain-specific references. We have provided examples(4) illustrating its intervention.

[Segment 130] PASS
[Segment 131] The term "pilum" in the source text seems to be a mistake or unclear. It might be intended to refer to "pylon" based on the term context provided. Consider verifying this with the editor.
[Segment 132] The translation of "sporter" as "孢子" is incorrect. Based on the term context, "sporter" might be a misinterpretation of "spore crawler," which should be translated as "孢子爬虫." Verify with the editor if "sporter" is indeed meant to be "spore crawler."
[Segment 133] The translation is missing a verb or context to make it a complete sentence. Consider adding context or a verb to improve fluency, such as "显然现在是Nagra的时刻，伙计。"
[Segment 134] The translation of "Spire" as "空军基地" is incorrect. According to the term context, "Spire" should be translated as "飞龙塔." Adjust the translation to reflect this terminology.

Table 4: proofreader log

The proofreader agent checks both the source and the translation, considering context information, which allows it to identify potential misinterpretations and suggest targeted corrections. In the above use case, it successfully detects anomalies in Segments 131 and 132. Segment 134 also showcases its ability to correct common terminology errors based on domain knowledge.

A.2 ViDove Demo Page

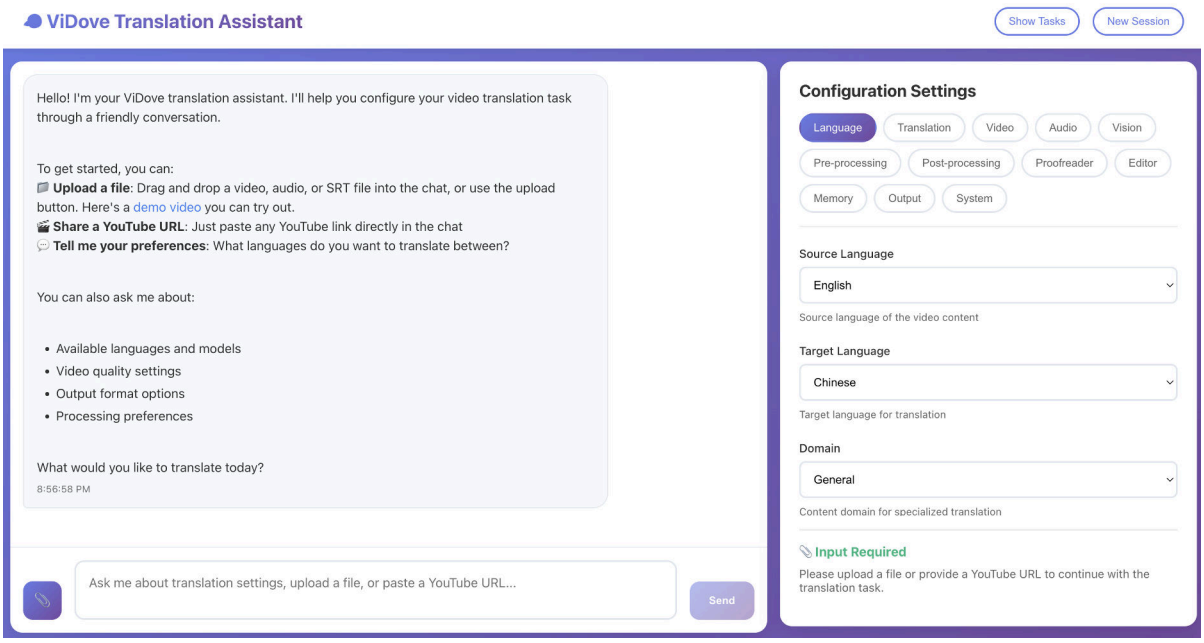


Figure 2: **User interface of ViDove**

A.3 Prompt for ViDove Agents

ViDove Translator Agent

" You are a professional translator. your job is to translate texts in domain of {domain} from {source language} to {target language}
you will be provided with a segment in source language parsed by line, where your translation text should keep the original meaning and the number of lines.
Keep every \n in the translated text in the corresponding place, and make sure to keep the same number of lines in the translated text.
You must break the translated sentence into multiple lines accordingly if original text breaks a complete sentence into different lines.
You should only output the translated text line by line without any other notation.
You current task is to translate the script in the domain of {domain} from {source language} to {target language}
Here are some supporting information including previous translation history, context documenting, supporting documents from internet and video clips description that might help you translate the text. Please refer to them if necessary. Previous translation history: \n
{Translation Histories}
if you detect any word is in the following context, please use it as a reference for current translation
{Context documents retrieved from knowledge base}
Here are some supporting documents that might help you translate the text, refer to them if necessary.:
{ supporting documents from web search }
Here are some descriptions of video clips that might help you translate the text, refer to them if necessary. :
{video clips descriptions}
Now please translate the following text from {source language} to {target language}
{text to be translated}
Your translation: "

ViDove Editor Agent

" You are an Editor ensuring overall translation quality and coherence, aligning the translation with the original video content in domain {domain}, you must ensure the term and style are aligned with the domain's language.
Segment index: {idx} Source text: {source}
Translated text: {translation}
Here is a provided suggestion for each segment, which may or may not useful for your revision, you may use the suggestion only if necessary (for example, term correctness). Note that the suggestion may not be accurate, the proofreader has less information comparing to you, so you need to double check before making revision. The proofreader may return "UNCLEAR" if they are not sure about the translation, they will specify the location and you need to check with other information provided to you to solve for unclear. If there is no suggestions, you may ignore this part, but still check with other modality context and long-term memory for correctness and coherence. Suggestion:
{suggestion if suggestion else "No suggestion provided."}
Your edit will also follow the following instruction if provided: User instruction: {user instruction if user introduction else "No user instruction provided."}
— Multimodal Context (Short-Term Memory) — Visual cues: You may use visual cues from the video to improve translation or make corrections, the source text might not be accurate, you need to check with the video context if provided: {visual context}
Audio cues: {audio context}
Translation context: You will be provided with the previous and next 5 segments' translations, which may help you understand the context and make corrections: Previous translation history (up to 5 segments): {Previous translation history} Past translation history (up to 5 segments): {Past translation history}
— Long-Term Memory — Long-term memory provides broader context and domain-specific knowledge, you may use it to improve translation or make corrections: {long term memory}
Notice: 1. Corrections or adjustments to better align text with the video context. 2. Suggestions for improving coherence across segments. 3. Logical consistency and any broader context adjustments. 4. Ensure the translation is accurate and aligned with the domain {domain}. 5. Ensure translation is smooth and fluent across segments. 6. To ensure the fluency in {target language}, you do not have to ensure translation be word by word accurate, but be sure to convey the same information.
— Important — Directly return the revised content only. "

ViDove Proofreader Agent

" You are a translation proofreader. Below are {number of segments} subtitle segments. Some are full sentences, some are fragments. Give **specific advice** for each one, but do not treat each segment separately you need information across segment.
Return suggestions in this format: Segment 0: [your comment here] Segment 1: [your comment here] ...
DO NOT return JSON. DO NOT rewrite the translation. Just return suggestion texts.
— {segments}
Short-term memory: {short term memory}
Term context: {local context}
Web memory context: {web search context}
Focus on: 1. Translation accuracy while sticking to domain {domain} (missing or incorrect meanings) 2. Fluency (grammar, spelling, repetition. Only if it affects understanding) and ensure the translation is smooth and fluent across segments. 3. Terminology (Use term context to edit idioms, ensure every sentence is translated into domain-specific language) 4. If you have no suggestions, return "PASS" for that segment. 5. Source text isn't 100% accurate. If you have doubt about the source text, return "UNCLEAR" and specify the location, editor will check the issue. 6. Only make suggestions if you believe revision is necessary. "

A.4 Translation Sample

Figure 3: **Vidove Video Cues Summarization for Translation Sample:** The image portrays a surreal and dramatic illustration featuring a large, exaggerated face of a bearded man on the left, expressing concern or thoughtfulness. On the right, a dark, patterned background showcases large white flames, with small skeletal figures interacting with them in the foreground. The scene is rendered in a monochromatic color scheme.

ORIGINAL TEXT	What happens when the devil walks among us?
GROUND TRUTH	若魔鬼化身凡人混迹其中，世界会变成什么样?
VIDOVE	当魔鬼行走在人间时会发生什么?
VIDEOCAPTIONER	当魔鬼在我们中间行走会发生什么?

Table 5: Case study for translation quality. Blue highlights translation deviations to VIDOVE, and red highlights deviations to VIDEOCAPTIONER.

B DoveBench

B.1 DoveBench Stats

DoveBench is a benchmark dataset designed to evaluate video translation and subtitling models. It contains a total of 50 videos, amounting to 17.23 hours of content and featuring 16,968 subtitle entries. The dataset's total text includes 189,157 words.

The dataset is composed of two distinct categories sourced from fan sub groups:

- **CS**: This category includes 23 Counter-Strike related videos from the "fazeclan galaxy archive" fan sub group. The videos in this section have an average duration of approximately 13 minutes (777.7 seconds).

- **SC2**: This category consists of 27 videos in the StarCraft 2 domain from the "StarPigeon Fan sub group". These videos are generally longer, with an average duration of over 27 minutes (1635.3 seconds).

A key feature of DoveBench is the inclusion of detailed ground truth, which is essential for evaluation. We provide human-annotated ground-truth subtitles for all videos. The dataset's comprehensive statistics on character counts, word counts, duration, and subtitle distribution make it a valuable resource for assessing the performance of video translation systems.

Category Statistics			**Overall Dataset Statistics**	
Statistic	**CS**	**SC2**	**Statistic**	**Overall**
			Number of Videos	50
Number of Videos	23	27	Total Duration	17.23 hours
			Average Duration	20.68 minutes
Total Duration	4.97 h (298.1 min)	12.27 h (735.9 min)	Total Subtitle Lines	16,968
			Avg. Subtitle Lines per Video	346.3
textscAverage Duration	12.96 min (777.7 s)	27.26 min (1635.3 s)	Total Words	189,157
			Avg. Words per Video	3,860

Table 6: Detailed statistics of the DoveBench dataset

C Baseline Systems Details

C.1 Gemini Prompt for DoveBench

Gemini Prompt (English Version)

" You are a professional transcription and translation assistant.
Please transcribe this audio/video file and translate it into Simplified Chinese. Carefully follow the instructions below:
1. Each segment should: - Contain a natural sentence or phrase in Simplified Chinese, not too long. - Have a valid start and end time in the format 'h:mm:ss,ms' (e.g., "0:00:01,229"). - Ensure the start time is less than the end time, and that each segment's start time equals the previous segment's end time (no overlap or gap). - If uncertain, round timestamps to the nearest 10 milliseconds.
2. Translation Guidelines: - First, accurately understand the original audio content. - Translate into natural, fluent Simplified Chinese. - Retain the meaning and tone of the original speech. - Keep proper nouns and technical terms accurate. - Maintain sentence boundaries suitable for subtitle readability.
3. Notes: - Proper nouns and technical terms — remain accurate. - Sentence boundaries — avoid breaking at unnatural pauses. - Chinese grammar and natural fluency.
Please provide the transcription and translation in the specified structured format.
The output language must be Chinese. "

C.2 Qwen-2.5-Omni

C.2.1 Prompts for DoveBench and BigVideo

Qwen 2.5 Omni Prompt (Chinese Version)

“翻译提供的视频中的说话内容到中文。只需要输出翻译内容原文，不要输出任何解释。”

C.2.2 Prompt Design and Video Processing Strategy

This section outlines the strategies employed for prompt design and video processing to optimize Qwen 2.5 Omni’s performance on the DoveBench and BigVideo datasets.

Prompt Language: Given the superior performance of Chinese prompts in enhancing Qwen 2.5 Omni’s instruction-following capabilities during preliminary evaluations, all experiments on both the DoveBench and BigVideo datasets exclusively utilized Chinese prompts.

Prompt Complexity: Specifically, this approach was adopted because initial trials with prompts designed similarly to those employed for Gemini demonstrated that Qwen exhibited a deficiency in instruction following when presented with complex instructions. This observation led to the selection of the aforementioned prompts, as they consistently yielded superior results.

Video Processing Strategy: Furthermore, due to Qwen 2.5 Omni’s constrained contextual understanding when processing video data, most videos within the datasets—even those only one to two minutes in duration—could not be processed directly. To address this limitation, videos were manually segmented into approximately ten-second clips. Each segment was then individually fed into the model, and the processed outputs for each segment were subsequently concatenated to form a complete SRT file.

Sanskrit Voyager: Unified Web Platform for Interactive Reading and Linguistic Analysis of Sanskrit Texts

Giacomo De Luca, Danilo Croce, Roberto Basili
Department of Enterprise Engineering
University of Rome Tor Vergata
Via del Politecnico 1, 00133, Rome, Italy
deluca@ing.uniroma2.it {croce,basili}@info.uniroma2.it

Abstract

Sanskrit Voyager is a web application for searching, reading, and analyzing the texts in the Sanskrit literary corpus. Unlike previous tools that require expert linguistic knowledge or manual normalization, Sanskrit Voyager enables users to search for words and phrases as they actually appear in texts, handling inflection, sandhi, and compound forms automatically while supporting any transliteration. The system integrates four core functionalities: (1) multi-dictionary lookup with morphological analysis and inflection tables; (2) real-time text parsing and annotation; (3) an interactive reader for over 900 digitalized texts; and (4) advanced corpus search with fuzzy matching and filtering. Evaluation shows over 92% parsing accuracy on complex compounds and substantially higher search recall than BuddhaNexus on challenging queries. Source code is publicly available under CC-BY-NC license, resource-efficient, and designed for both learners and researchers, offering the first fully integrated, user-friendly platform for computational Sanskrit studies [1].

1 Introduction

Sanskrit occupies a unique position as an ancient language whose texts remain highly relevant today. Major treatises of multiple religious traditions (Buddhism, Jainism, Vedic religions, Vaishnavism, Shaivism) and foundational texts on yoga are written in Sanskrit. With many authors arguing about the necessity of a dialogue still to come between Western and Indian philosophy (Garfield, 2014; Westerhoff, 2009), access to Sanskrit literature has become increasingly important. However, computational access remains limited: major digital libraries (e.g. GRETIL[2], SARIT[3], Muktabodha[4], DSBC[5], SanskritDocuments[6]) typically provide plain-text corpora with minimal linguistic annotation, while more advanced platforms such as the Digital Corpus of Sanskrit (DCS)[7] (Hellwig, 2007) and BuddhaNexus[8] focus either on annotated subsets or cross-text search but lack unified, interactive reading and analysis environments.

A key technical challenge is posed by Sanskrit's pervasive *sandhi* (euphonic word merging), productive compounding, and rich inflectional morphology, which obscure word boundaries and complicate dictionary lookup or search (Bhardwaj et al., 2018; Huet, 2009; Hellwig and Nehrdich, 2018; Sandhan et al., 2022). While recent advances, including byte-level transformers fine-tuned for Sanskrit (Nehrdich et al., 2024) and new hybrid approaches (De Luca, 2025), have improved automatic segmentation and analysis, available tools still require significant linguistic expertise and often force users to combine multiple platforms for even basic research or reading tasks.

In this paper, we present Sanskrit Voyager, an integrated web application that addresses these limitations through four core functionalities: (1) dictionary search capable of handling inflected, sandhi-affected, and compound forms across multiple dictionaries, also returning grammatical tagging and inflection tables; (2) real-time text analysis with automatic parsing and annotation for user uploaded texts or books; (3) an interactive Book Reader for more than 900 texts with optional machine translation; and (4) Corpus Search with advanced filtering and navigation. Unlike existing tools, which offer

[1] Application: https://www.sanskritvoyager.com
Source code: https://github.com/SanskritVoyager
Demo video: https://youtu.be/FCKlW4NKJEc

[2] https://gretil.sub.uni-goettingen.de
[3] https://sarit.indology.info/
[4] https://muktabodha.org/
[5] https://www.dsbcproject.org/
[6] https://sanskritdocuments.org/
[7] http://www.sanskrit-linguistics.org/dcs/
[8] https://github.com/dharmamitra/dharmanexus

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 244–253
November 4-9, 2025

these features only in isolation, Sanskrit Voyager integrates them into a unified, mobile-compatible platform, making Sanskrit texts accessible both to newcomers and advanced researchers. The system automatically normalizes input (including inflection, sandhi, compounding, and transliteration), enabling both accurate multi-dictionary lookup and morphological analysis for words as they appear in actual texts. This approach contrasts with existing tools, which typically require manual normalization or specialist linguistic knowledge. The platform covers over 900 texts from major digital corpora (including GRETIL and SARIT), providing on-demand word-by-word annotation and exploration. All processing is performed automatically and in real time, removing the need for manual pre-annotation. By bringing together these core functionalities within a single web application, Sanskrit Voyager addresses long-standing usability barriers and supports both comprehensive search and detailed linguistic study.

We review related work in Section 2, describe the system's design and functionalities in Sections 3 and 4, present evaluation results in Section 5, and conclude in Section 6.

2 Related Work

Sanskrit digital humanities have made significant advances in the last decade, yet current tools remain fragmented, with each addressing only a subset of the linguistic and technical challenges.

Sanskrit Word Segmentation and Analysis. The resolution of *sandhi* and compounds, collectively known as the Sanskrit Word Segmentation (SWS) (Krishna et al., 2017) task, has been the primary focus of computational linguistics for Sanskrit. Traditional rule-based systems (Huet, 2009), statistical models (Hellwig and Nehrdich, 2018), and recent hybrid approaches (Sandhan et al., 2022) have all targeted these problems, but the ambiguity and complexity of the language persist. Token-based transformer models also struggle due to pervasive word merging, a direct consequence of *sandhi*. Recently, byte-level transformer models such as ByT5 (Xue et al., 2022), especially those fine-tuned for Sanskrit (Nehrdich et al., 2024), have enabled substantial progress, allowing mature web applications like Dharmamitra to emerge. However, even state-of-the-art models face challenges such as oversplitting: for example, in the *bhagavadgītā* section of the Sandhikosh benchmark (Bhardwaj et al., 2018), correct splitting is considered *bhagavat+gītā*, thus losing the dictionary entry and meaning of the compound title. Our system builds on these advances with a cascading approach that combines fast, rule-based processing for simple terms with statistical models for more complex words, thus enabling efficient and robust analysis for around 70% of the words, while the remaining 30% are delegated to the more computationally expensive methods.

Dictionary Access and Linguistic Tools. The University of Cologne project (Cologne Digital Sanskrit Dictionaries) offers the largest collection of digitized Sanskrit dictionaries (42 in total), available via web or Python PyCDSL library. While these resources are rich, interfaces require users to select transliteration formats and to search using the lemma or exact root form, limiting usability for non-specialists. Features such as multi-dictionary search, diacritic-insensitive queries, or inflected word recognition are present, but only in isolation and not in combination. The Sanskrit Heritage dictionary offers inflected search in French. Inflected form search is also present on the Sanskrit-Dictionary[9] website. The Sanskrit Reader Companion (Huet and Goyal, 2013), also on the Heritage website, separately provides sandhi splitting. Some websites, like Kosha.app[10], provide collections of dictionaries. However, even with these tools, since a large portion of the terms have some form of sandhi or compounding, retrieving dictionary entries still requires significant linguistic knowledge. The recent Dharmamitra platform[11] represents a notable innovation, using ByT5-Sanskrit (Nehrdich et al., 2024) for automatic sandhi and stemming, but supports only a simplified Monier-Williams dictionary and is not by design a text reader interface.

Digital Corpora and Text Readers. Access to the Sanskrit textual tradition is similarly fragmented. GRETIL (Göttingen Register of Electronic Texts in Indian Languages) is the principal repository, offering over 800 texts with inconsistent TEI encoding and minimal formatting. SARIT provides higher-quality consistent XML for a smaller corpus (63 texts), while other collections like Muktabodha and the Digital Sanskrit Buddhist Canon (DSBC) focus on specialized traditions. The TITUS project[12] provides a broad but difficult-to-use corpus, due to its

[9] https://sanskritdictionary.com/
[10] https://kosha.app/
[11] https://dharmamitra.org/
[12] https://titus.uni-frankfurt.de

proprietary transliteration and restrictive licensing. BuddhaNexus (now DharmaNexus) [13] aggregates 1,700+ texts and enables cross-corpus search, but its interface for reading and analysis remains basic. In contrast, platforms for classical Greek, such as Scaife[14], enable integrated, interactive reading and search, something still missing for Sanskrit.

Shortcomings of Current Tools and Opportunities for Sanskrit Voyager. Despite substantial progress, existing platforms remain fragmented: some, like DCS (Hellwig, 2007), provide high-quality manual annotations for approximately 260 texts, offering gold-standard linguistic analysis but with limited corpus coverage and no real-time processing capabilities. Others aggregate large corpora but offer minimal linguistic support, limited search, or interfaces requiring specialist skills. Few systems combine sandhi, inflection, and compound handling in dictionary queries, and almost none provide cross-corpus search or mobile optimization. Formatting inconsistencies, restrictive licenses, and the absence of integrated, interactive reading further hinder accessibility for non-experts. Sanskrit Voyager addresses these gaps by providing, for the first time, a unified, web-based environment that combines real-time linguistic analysis, comprehensive and user-friendly dictionary search (handling inflected, sandhi-merged, and compounded forms in any transliteration), and large-scale interactive corpus reading and search.

3 System Architecture

Sanskrit Voyager is structured as a modular client-server architecture designed to balance computational efficiency, maintainability, and extensibility. The system is composed of a backend deployed on a cloud-based Dokku platform and a web-based frontend hosted on Vercel.

The backend, running on a DigitalOcean droplet under Dokku, is built around a Python application that centralizes all core linguistic operations: transliteration, sandhi and compound splitting, stemming, and dictionary lookup, implemented via the open-source *process-sanskrit* library[15]. Query preprocessing and database management are handled with SQLAlchemy ORM, facilitating modularity and future code reuse. PostgreSQL acts as both the main data storage and the core search engine for the platform. By leveraging advanced full-text indexing and ranking features, the system can efficiently search and retrieve relevant results across large and heterogeneous Sanskrit corpora. Multiple representations of each text (original, normalized, stemmed, etc.) are indexed and weighted differently, so that users can search effectively regardless of transliteration or input format. Redis is integrated as a caching layer for frequent or computationally expensive queries, supporting low-latency interaction even on modest hardware. The entire backend runs efficiently on a DigitalOcean droplet with 2 vCPUs, 4GB RAM, and 80GB disk storage, demonstrating the resource-efficiency of our architecture. All services are deployed as Dokku plugins, allowing for straightforward scaling, backup, and migration. System functionalities are exposed via a RESTful API (built with Flask), decoupling the frontend from the backend and enabling potential future integration with other research tools, workflows, or interfaces. The user-facing frontend is implemented as a single-page React application using Vite for development speed and performance optimization. Mantine UI, customized for accessibility and visual consistency, underpins the UI, making the application usable across devices and screen sizes. The mobile experience features a reimplemented interface with mobile-specific components, custom hooks for touch interactions, and optimized layouts. Comprehensive documentation, written in Docusaurus and visually integrated with the main application, is hosted separately to support reproducibility and onboarding of new users.

The architecture is designed for scalability and sustainability: all components are source-available and cloud-ready, allowing easy deployment, updates, and horizontal scaling. Additional modules, such as new NLP pipelines or dictionary sources, can be integrated with minimal effort, making the system flexible and future-proof.

4 Core Functionalities

Sanskrit Voyager provides four core features in an integrated environment: dictionary lookup for inflected and compound forms, real-time text parsing, interactive reading of digitized texts, and full-corpus search with fuzzy matching. Each module is outlined below.

Dictionary Search. Traditional Sanskrit dictionaries require users to master complex linguis-

[13] https://dharmamitra.org/nexus

[14] https://scaife.perseus.org/

[15] https://github.com/Giacomo-De-Luca/Process-Sanskrit

Figure 1: Dictionary Search interface for the query *pratiprasave*. The system analyzes the input, splits compounds, assigns grammatical features, and shows aggregated multi-dictionary results. Subwords and citations are clickable for further exploration. Sources are exapandable on hover.

tic transformations, such as recognizing inflected forms (e.g., "eṣyāmi" as the future of "i"), resolving sandhi, splitting compounds, and navigating various transliteration schemes. Sanskrit Voyager removes these barriers by allowing users to search for words exactly as they appear in texts—inflected, compounded, sandhi-merged, or in any common transliteration. Queries are automatically normalized (including diacritic-insensitive matching, so "*samskara*" matches "*saṃskāra*") and require no manual settings, making the system accessible even to users with no linguistic expertise.

The dictionary interface aggregates results from six major Sanskrit lexica (Monier-Williams, Apte, Cappeller, Macdonell, Grassmann, Edgerton) and the Concise Pali Dictionary[16], with over 246,000 unique entries. All abbreviations and cross-references are expanded, and Sanskrit terms within entries are interactive: clicking on a subword or citation navigates directly to the relevant entry or retrieves the cited passage via Corpus Search. Query processing relies on a modular, three-stage cascading system (De Luca, 2025). The pipeline starts with deterministic stemming and sandhi splitting

[16] https://buddhistuniversity.net/content/reference/concise-pali-dictionary

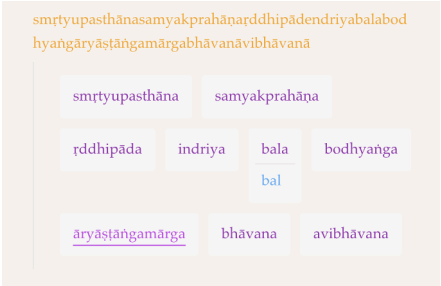


Figure 2: Text Analysis Tool output for a long compound. Each constituent is automatically split and displayed as an interactive card, with clickable access to dictionary entries and grammatical analysis.

via inflection tables and rewriting rules. A statistical parser (`sanskrit_parser` or ByT5-Sanskrit) is invoked for compounds or sandhi blocks, when SQL queries return null (as these are not in the inflection tables). The results from the previous step are scored on morphology and length and sorted by score. A right-truncation heuristic with rewriting is applied if the best score result from the previous step is below a threshold. This modular design allows efficient processing: simple lookups are deterministically processed in under 100 ms, while even the longest compounds complete within 3 seconds. For each matched stem, the system returns grammatical tagging, inflection, dictionary entries for the selected dictionaries and, where available, full inflection tables and a decomposition into sub-components (e.g., "*prati-prasava*" for "*pratiprasava*"). All results are fully interactive and can be further explored. This approach supports a wide range of users, from those interested in understanding terms commonly used in mantras or yoga postures to advanced researchers. It also enables ongoing extension to new dictionaries, parsing modules, or related languages.

Text Analysis Tool. The text analysis tool enables users to input Sanskrit text or to upload books in any transliteration format, which is automatically detected and transliterated. Each segment is transformed into an interactive element which is parsed on-click using the same cascading pipeline as the dictionary search, including sandhi removal, compound splitting, and grammatical analysis, with all possible parses displayed. Roots, stems, and grammatical tags are rendered as interactive elements. Users can click on subwords to instantly access their dictionary entries, and ambiguous forms are shown with all available interpretations. Color coding is used to vi-

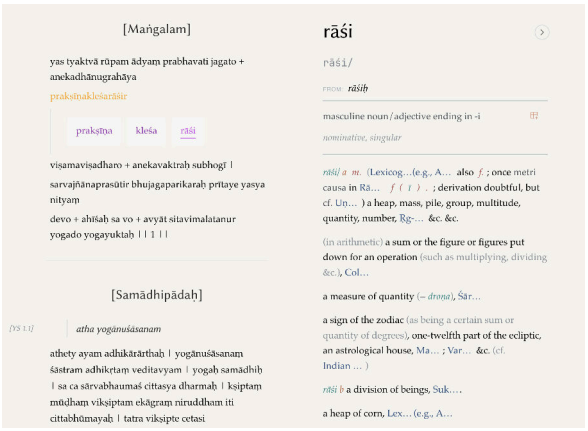


Figure 3: Book Reader: Interactive reading of digitized Sanskrit texts (here: Pātañjalayogaśāstra), with automatic segmentation of compounds (*prakṣīnakleśarāśir*), morphological annotation, and direct dictionary lookup (right panel).

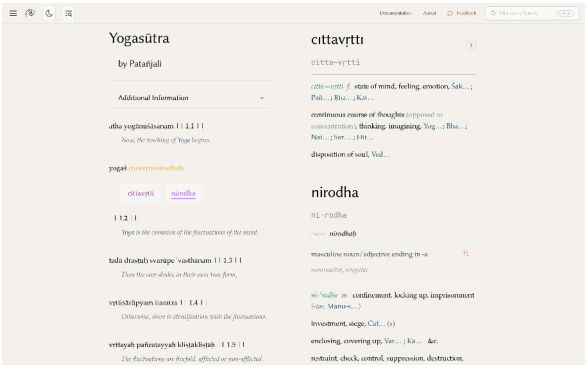


Figure 4: Book Reader Translations: showing line by line automatic translation of the Yogasūtra. The original text and Sanskrit terms in the translations can be clicked for analysis.

sually distinguish verbs, indeclinables, and pronouns, making grammatical patterns immediately clear. Figure 2 demonstrates the tool's output on a complex compound from a Buddhist text. The original compound, which merges several technical terms (e.g., *smṛtyupasthāna*, *samaykpradhāna*, *bodhyaṅga*, *āryāṣṭāṅgamārga*), is automatically decomposed into its constituents, each presented as a clickable card. Selecting a subword (such as *bala*) reveals its root and dictionary entry, and underlined terms link to further analysis or cross-references. This interactive display allows both students and scholars to dissect long compounds and explore their morphology and semantics without manual intervention, greatly simplifying the analysis of real Sanskrit texts. The pipeline is fully modular: each processing stage is implemented as a distinct component in the backend, allowing for easy extension or substitution (e.g., swapping in improved models, new inflection tables, or custom rule sets). This architecture enables both rapid experimentation with new NLP models and straightforward adaptation to other Indic languages or custom annotation schemes.

Book Reader. The Book Reader provides interactive access to over 900 digitized Sanskrit texts from the GRETIL and SARIT collections. Users can select books by searching for titles or authors, with normalization handling various spelling and diacritic forms. All texts were pre-processed and converted from XML into a structured JSON format that preserves metadata, hierarchical segmentation, enables advanced interaction and allows translations to be added. Figure 3 shows the Book Reader interface applied to a classical text. The left panel displays the Sanskrit text segmented by chapter and verse, with complex compounds automatically split and highlighted. Clicking on any segment (such as *prakṣīnakleśarāśir*) brings up grammatical analysis and dictionary lookups for each subword (see the right panel for the analysis of *rāśi*). Metadata, licenses, and annotations are presented in expandable accordions. Inline notes and text variants appear as numbered elements positioned alongside the text that expands on-click. A table of contents is automatically generated from chapter headings for navigation. For a subset of books, which will be gradually expanded until the entire corpus is covered, the Book Reader offers segment-level machine translation (using state-of-the-art LLMs), with Sanskrit terms in the translation remaining clickable for direct dictionary lookup, facilitating learning and research. This integrated, navigable format is designed to support both reading fluency and detailed linguistic study, far surpassing the static, plain-text interfaces typical of other digital Sanskrit libraries.

Corpus Search. The Corpus Search module allows users to search across the entire Sanskrit corpus using both exact and fuzzy queries, with advanced filtering and navigation features. Currently it enables searching and reading within the 1700+ Buddhanexus corpus, which will be expanded in a later release. Unlike traditional interfaces, which typically require users to know the precise lemma or root form, our system supports real-world queries: users can search for any inflected, sandhi-merged, or compounded form, and the system will automatically normalize, stem, and match across multiple

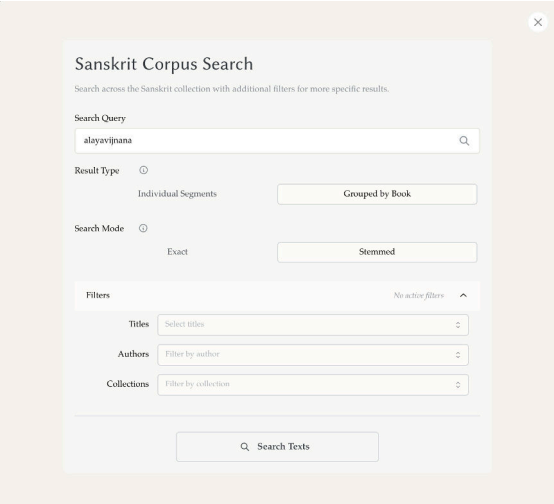


Figure 5: Corpus Search interface: users can specify query type, result grouping, and filters for author, title, and collection.

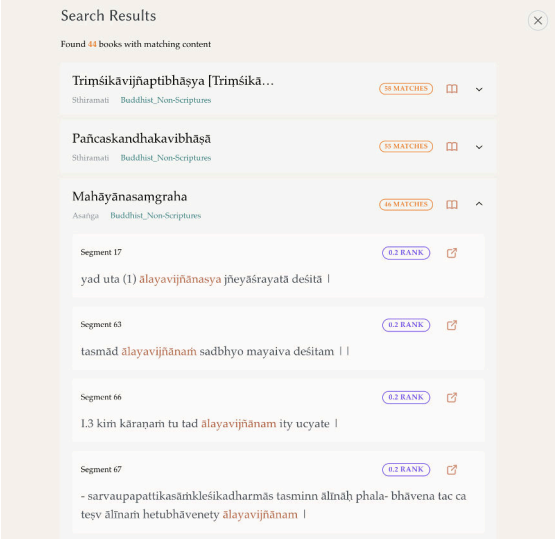


Figure 6: Corpus Search results: books with matches are grouped and ranked, with contextual snippets for each occurrence of the query term (here, "*alayavijñāna*"). Clicking a match opens the text at the passage in the Book Reader.

text representations. Figure 5 shows the Corpus Search interface, where users can enter queries, select result grouping (by individual segment or by book), and set the search mode (exact or stemmed). Additional filters for author, title, and collection are available, and all inputs are normalized to handle variant spellings and diacritics. Internally, each text segment in the corpus is indexed in multiple forms: original, cleaned, sandhi/compound-split, diacritic-free, and stemmed. These representations are stored in a single PostgreSQL column with weighted *ts_vectors* in decreasing order. This unified indexing approach allows queries like "*alayavijnana*" to match "*ālayavijñāna*". PostgreSQL GIN indexes with fuzzy match extensions ensure efficient retrieval even for complex queries. Queries exceeding 40 characters are matched against a folio-level materialized view using the same indexing approach. Figure 6 displays typical results for a query such as "*alayavijñāna*": the left panel groups matches by book (with match counts), while the right panel lists all occurrences of the search term within context. Clicking on any match jumps directly to the corresponding location in the Book Reader, where further analysis and navigation are available. All titles, authors, and collections are normalized and filterable for corpus-wide exploration. Queries take between 150 and 350 milliseconds to complete.

5 Evaluation

We evaluated Sanskrit Voyager's core parsing and search capabilities, which underpin all functionalities of the system.

Parsing Accuracy. Because existing benchmarks like Sandhikosh cover only sandhi and not compounds, and DCS-derived resources use incompatible conventions, we designed a custom evaluation. We randomly sampled 300 compounds from the GRETIL corpus, across three length categories (10–20, 20–40, and 40+ characters, up to 283), and evaluated parsing accuracy. Our cascading architecture achieved over 90% accuracy in all categories: 93.1% for medium, 94.2% for long, and 91.1% for very long compounds (overall: 92.4%). On the entire Yoga Sūtra (665 words), accuracy reached 95.9%, reflecting the system's strength when single words are taken into account.

Corpus Search. We compared Voyager's search module with BuddhaNexus using 10 challenging queries (5 technical terms, 5 multi-word phrases, automatically selected for diversity). Sanskrit Voyager consistently returned more results, e.g., 472 matches for "*ālayavijñāna*" (vs 73 in BuddhaNexus), and robust retrieval for complex phrases like "*sarvabījakam vijñānam*" (18 matches vs 2).

Error Analysis and Usage. Most parsing errors (40%) are due to dictionary gaps, especially for rare or Buddhist technical terms, followed by missing inflection forms (30%), challenging sandhi (20%), and rare morphology (10%). The application has been publicly deployed for two months, with only

Table 1: Selected search queries: Voyager vs BuddhaNexus.

Query	Voyager	BuddhaNexus
ālayavijñāna	472	73
satkāryavāda	152	40
anirvacanīya	409	200+
sarvabījakaṃ vijñānam	18	2
vākyaṃ rasātmakaṃ kāvyam	4 (1 duplicate entry)	2

organic discovery. In June 2025 it was used by 204 users, averaging 9:16 minutes per session. The 81 users active in the preceding the paper completion averaged 11:36 minutes per session, demonstrating strong user engagement and usability of the platform.

To gather direct user impressions, a feedback section was made available on the website, inviting users to evaluate the system's usability and effectiveness through the questionnaire reported in Appendix A. At the time of writing, around ten users have provided responses, consistently assigning the maximum scores for all categories, indicating a highly positive reception with respect to both ease of use and the practical utility of the platform. While these early results are promising and demonstrate strong user appreciation, the current sample remains limited. The feedback form remains open and we plan to conduct a more extensive survey to obtain a broader and more representative assessment of the system's impact as user adoption grows.

6 Conclusion

Sanskrit Voyager makes the Sanskrit literary tradition accessible to a broad audience, enabling search, analysis, and interactive reading over hundreds of digitized texts, with no prerequisite knowledge of morphology, sandhi, or transliteration. By unifying dictionary lookup, text analysis, book reading, and corpus search in a single platform, the system lowers entry barriers for learners and streamlines research workflows for specialists.

Source available, released with CC-BY-CD license and resource-efficient, Sanskrit Voyager demonstrates how modern NLP and thoughtful design can unlock complex classical languages for both scholarship and the wider public.

Limitations

While Sanskrit Voyager currently performs corpus search on the BuddhaNexus dataset, integration of additional Sanskrit digital libraries is in progress, with preliminary versions of a unified "Sanskrit-Literature" dataset already prepared. Some advanced features, such as large-scale machine translation and embedding-based retrieval, are planned for future releases as resource and evaluation cycles permit. Machine translation was tested using multiple LLMs (GPT-4o, GPT-5, Gemini 2.5 Pro, Claude Sonnet 3.7, and the Dharmamitra model[17]), but is currently limited to a subset of texts pending development of appropriate evaluation metrics to determine which model produces the most accurate translations.

The current pipeline achieves strong performance for most literary and technical queries, though rare terms not present in available dictionaries or inflection tables may yield incomplete analysis. Integration with the inflection tables of the Heritage website to increase coverage is planned but not yet in place. Future implementations will also expand dictionary coverage to include traditional lexical resources such as koshas and the Dhātupātha pending the manual formatting required for each dictionary. Broader community and domain expert feedback will further refine system accuracy and coverage as the project evolves.

Ethics and Broader Impact

Sanskrit Voyager aims to democratize access to Sanskrit literature for both academic and general audiences. All included corpora are sourced from open-access or Creative Commons digital libraries; dictionary and software components are distributed under open-source non commercial licenses. No copyrighted or restrictively licensed materials, including proprietary translations, contemporary commentaries, or materials with unclear provenance, were incorporated into the platform. The platform is designed to support linguistic and cultural research, education, and the preservation of textual heritage, with no features enabling commercial exploitation or user profiling. Limitations in dictionary coverage and morphological analysis may impact the accuracy for certain technical or rare terms, and the system does not provide human or cultural context for interpreting sacred or ritual language. Developers are committed to ongoing improvement, community feedback, and responsible handling of cultural material.

[17] https://huggingface.co/buddhist-nlp/gemma-2-mitra-it

References

Shubham Bhardwaj, Neelamadhav Gantayat, Nikhil Chaturvedi, Rahul Garg, and Sumeet Agarwal. 2018. Sandhikosh: A benchmark corpus for evaluating sanskrit sandhi tools. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Cologne Digital Sanskrit Dictionaries. 2025. Cologne digital sanskrit dictionaries.

Giacomo De Luca. 2025. Accessible Sanskrit: A cascading system for text analysis and dictionary access. In *Proceedings of the Second Workshop on Ancient Language Processing*, pages 38–46, The Albuquerque Convention Center, Laguna. Association for Computational Linguistics.

Jay L Garfield. 2014. *Engaging Buddhism: Why it matters to philosophy*. Oxford University Press.

Oliver Hellwig. 2007. Sanskrittagger: A stochastic lexical and pos tagger for sanskrit. In *International Sanskrit Computational Linguistics Symposium*, pages 266–277. Springer.

Oliver Hellwig and Sebastian Nehrdich. 2018. Sanskrit word segmentation using character-level recurrent and convolutional neural networks. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2754–2763.

Gérard Huet. 2009. Sanskrit segmentation. In *Proceedings of the South Asian Languages Analysis Roundtable XXVIII*, Denton, Ohio.

Gérard Huet and Pawan Goyal. 2013. Design of a lean interface for sanskrit corpus annotation. *Proceedings of ICON*, pages 177–186.

Amrith Krishna, Pavankumar Satuluri, and Pawan Goyal. 2017. A dataset for sanskrit word segmentation. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 105–114.

Sebastian Nehrdich, Oliver Hellwig, and Kurt Keutzer. 2024. One model is all you need: ByT5-Sanskrit, a unified model for Sanskrit NLP tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13742–13751, Miami, Florida, USA. Association for Computational Linguistics.

Jivnesh Sandhan, Rathin Singha, Narein Rao, Suvendu Samanta, Laxmidhar Behera, and Pawan Goyal. 2022. Translist: A transformer-based linguistically informed sanskrit tokenizer. *arXiv preprint arXiv:2210.11753*.

Jan Westerhoff. 2009. *Nagarjuna's Madhyamaka: A philosophical introduction*. Oxford University Press.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

A Appendix: User Evaluation Questionnaire

As part of our user evaluation, we invited participants to complete the following questionnaire regarding their experience with the Sanskrit Voyager system (https://www.sanskritvoyager.com). The questionnaire included the following items:

1. **Usefulness for Reading and Researching Sanskrit Texts**
   How useful do you find the system for reading and researching Sanskrit texts?

   - 1 – Not useful, does not solve any real problem
   - 2 – Slightly useful, solves only very limited problems
   - 3 – Moderately useful, but other tools offer similar functionality
   - 4 – Useful, effectively solves several important issues
   - 5 – Very useful, solves concrete problems that other systems do not address

2. **Ease of Use**
   How do you rate the system's ease of use?

   - 1 – Very difficult and frustrating
   - 2 – Quite difficult, takes time to learn
   - 3 – Moderately intuitive, but room for improvement
   - 4 – Easy, quick to learn
   - 5 – Very easy and intuitive from the start

3. **Dictionary Search Functionality**
   How do you rate the dictionary search functionality?

   - 1 – Poor, does not retrieve correct words and is hard to use
   - 2 – Limited, sometimes retrieves entries, sometimes not
   - 3 – Adequate, finds entries but no improvement compared to other projects
   - 4 – Good, retrieves most entries and additional information is useful
   - 5 – Excellent, easy to use and correctly retrieves entries

4. **Automatic Text Parsing and Analysis**
   How would you rate the automatic text parsing and analysis feature?

   - 1 – Poor, too many errors and unreliable
   - 2 – Fair, works but often incorrect
   - 3 – Acceptable, some errors but still helpful
   - 4 – Good, few errors and generally reliable
   - 5 – Excellent, highly accurate and trustworthy

5. **Corpus Search Functionality**
   How do you rate the corpus search functionality?

   - 1 – Poor, very limited and imprecise
   - 2 – Limited, often misses relevant results
   - 3 – Adequate, finds relevant content with some flaws
   - 4 – Good, easy to find useful results

- 5 – Excellent, very precise and comprehensive

6. **System Response Time**
   Are you satisfied with the system's response time?

   - 1 – No, too slow for practical use
   - 2 – Quite slow, needs major improvement
   - 3 – Moderately fast, but could be better
   - 4 – Fast and convenient to use
   - 5 – Very fast, pleasantly responsive

7. **Recommendation Likelihood**
   Would you recommend Sanskrit Voyager to colleagues or students interested in Sanskrit studies?

   - 1 – Definitely not, not recommendable
   - 2 – Probably not, better tools exist
   - 3 – Not sure, maybe in some specific cases
   - 4 – Yes, I would generally recommend it
   - 5 – Absolutely yes, I would strongly recommend it

8. **Additional Feedback**
   Please provide any additional feedback or comments (optional).

Participants were also provided with a link to the system documentation (https://www.sanskritvoyager.com/docs) to facilitate their evaluation. Responses are used exclusively for system improvement and research purposes.

PromptSuite: A Task-Agnostic Framework for Multi-Prompt Generation

Eliya Habba* **Noam Dahan*** **Gili Lior** **Gabriel Stanovsky**
The Hebrew University of Jerusalem
eliya.habba@mail.huji.ac.il

Abstract

Evaluating LLMs with a single prompt has proven unreliable, with small changes leading to significant performance differences. However, generating the prompt variations needed for a more robust multi-prompt evaluation is challenging, limiting its adoption in practice. To address this, we introduce PromptSuite, a framework that enables the automatic generation of various prompts. PromptSuite is flexible – working out of the box on a wide range of tasks and benchmarks. It follows a modular prompt design, allowing controlled perturbations to each component, and is extensible, supporting the addition of new components and perturbation types. Through a series of case studies, we show that PromptSuite provides meaningful variations to support strong evaluation practices. All resources, including the Python API, source code, user-friendly web interface, and demonstration video, are available at: https://eliyahabba.github.io/PromptSuite/.

1 Introduction

Recent studies have demonstrated that LLMs are highly sensitive to small, meaning-preserving variations in task formulation. Minor changes, ranging from adding white spaces to instruction paraphrasing, lead to substantial differences in performance and model ranking (Sclar et al., 2023; Mizrahi et al., 2024). This sensitivity has been explored in the evaluation of many NLP tasks in zero and few shot settings, such as text classifications (Chakraborty et al., 2023; Reif and Schwartz, 2024); multiple-choice question answering (Habba et al., 2025; Alzahrani et al., 2024); and text generation tasks (Resendiz and Klinger, 2024), raising concerns about the validity of evaluation performed using a single prompt.

Evaluating over multiple prompts is currently challenging because there is no standard way to extend existing benchmarks, which were largely compiled using a single prompt. Evidently, despite its major limitation, single-prompt evaluations are still prevalent in many NLP tasks (Gu et al., 2024a,b; Lior et al., 2025).

To address this major challenge standing in the way of meaningful evaluation in NLP, we present PromptSuite, a framework that generates multiple prompts, employing both LLMs as well as rule-based heuristics to generate variations along dimensions that were found to affect model performance. PromptSuite is built around three core principles, presented in Section 2. First, PromptSuite is *flexible*, designed to work out of the box on a wide range of benchmarks. Second, PromptSuite follows a *modular* design that decomposes prompts into four components: instruction, prompt format, demonstration, and instance content, and PromptSuite enables targeted perturbations to each component, making it easy to evaluate their impact and adapt to new tasks. Finally, PromptSuite is *extensible,* supporting future LLM evaluation research with easy extensions for new prompt components and perturbations.

PromptSuite provides different types of perturbations to each prompt component, including formatting, paraphrasing, context addition, and few-shot demonstration editing, as illustrated in Figure 1. In Section 3, we provide further details on the perturbation types, as well as demonstrate how to use our API to transform raw data into multiple prompt variations with just a few lines of code.

In Section 4, we demonstrate the flexibility of our framework through a series of case studies. We assess the impact of prompt variation on nine diverse tasks with two SOTA LLMs, highlighting the utility of PromptSuite for multi-prompt evaluation.

Our contributions are as follows:

*Equal contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 254–263

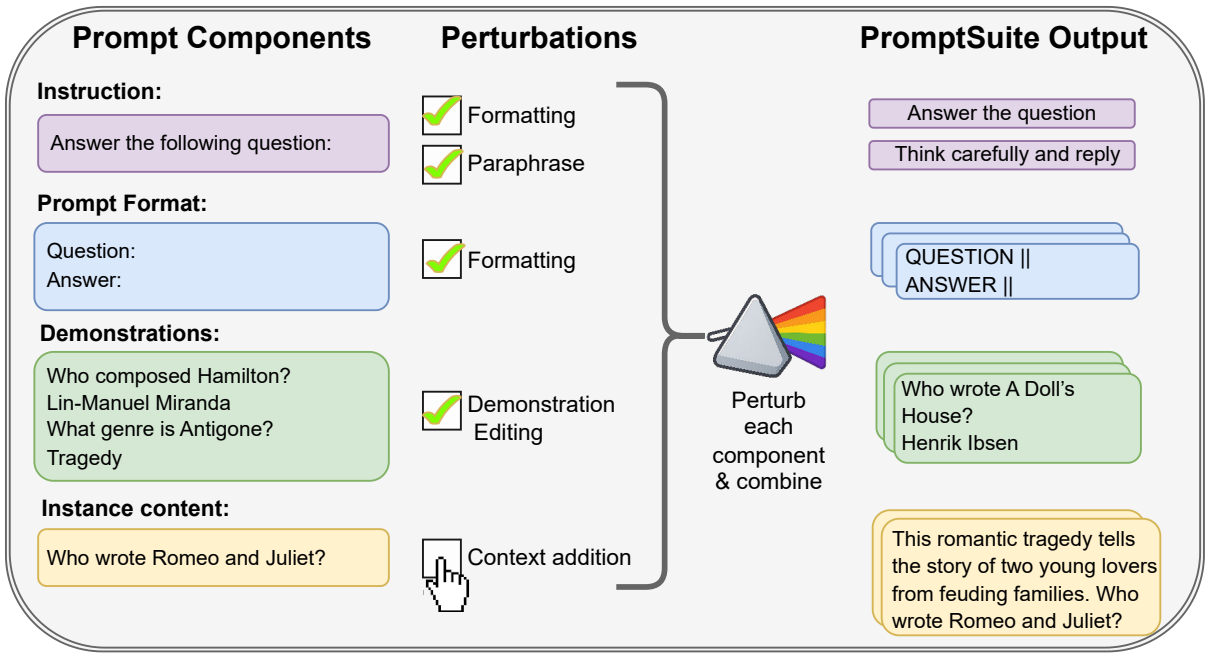


Figure 1: PromptSuite framework: configure a modular prompt, and apply component-wise perturbations. This modularity enables PromptSuite to generalize across tasks and adapt to diverse data.

1. We present PromptSuite, an easy-to-use framework that provides the prompt variations needed for multi-prompt evaluation across a wide range of tasks, working out of the box with diverse benchmarks and datasets.

2. We evaluate PromptSuite's capabilities through a series of case studies that demonstrate its ability to reveal LLM sensitivity to prompt variations across a diverse set of tasks.

3. We show PromptSuite's modular design enables isolating the effect of individual prompt component perturbations, enabling future research into the causes of LLM sensitivity.

2 PromptSuite's Principles

We build PromptSuite on three core principles to make it useful across a variety of use cases and applicable over time.

Flexible PromptSuite must be able to support a diverse range of tasks out of the box. We achieve this by relying only on the prompt structure and not its content. We demonstrate this flexibility in Section 4, where we use PromptSuite to get prompt variations in 9 different tasks, including question answering, multiple-choice reasoning, translation, summarization, and code generation.

Modular PromptSuite is built around a modular representation, treating each prompt as a combination of independent components, as shown in Figure 1. This design enables targeted perturbations to specific components, supporting evaluation of their impact on model performance and allowing adaptation to new tasks, formats, or datasets.

Extensible PromptSuite is built to support future research in the field of LLM evaluation. In particular, it is easy to add new prompt components and new types of perturbations through our open-source code.[1]

3 PromptSuite

PromptSuite is a flexible, modular and extensible framework that generates diverse prompts needed for robust evaluation. For a given dataset, it outputs a set of prompt variations for few-shot or zero-shot settings, where each sample from the dataset appears multiple times with different prompts. For example, in Listing 1, we load SQuAD (Rajpurkar et al., 2016a) through Hugging Face, set up the prompt and choose to paraphrase the instruction and apply formatting to the prompt format. This results in 9 prompt variations per sample, with just a few lines of code, and minimal information about the dataset.

[1] https://www.github.com/eliyahabba/PromptSuite

```
from promptsuite import PromptSuite

# Initialize
ps = PromptSuite()

# 1) Load dataset directly from HF
ps.load_dataset("rajpurkar/squad")

# 2) Setup template and 3) Choose
    perturbations
template = {
  'instruction': 'Please␣answer␣the␣
      following␣questions.',
  'prompt␣format': 'Q:␣{question}\nA:␣{
      answer}',
"instruction␣variations": ["
    paraphrase_with_llm"],
"prompt␣format␣variations": ["format␣
    structure"],
}
ps.set_template(template)

# 4) Generate variations
ps.configure(variations_per_field=3,
    api_platform="OpenAI", model_name="
    gpt-4o-mini")
variations = mp.generate(verbose=True)

# Export results
ps.export("output.json", format="json")
```

Listing 1: Code snippet for using PromptSuite API. Here we apply paraphrasing to the instruction with an LLM and formatting to the prompt format.

In this section, we briefly describe the modular prompt design and the supported perturbations, followed by an overview of how PromptSuite is used.

3.1 Modular Prompt and Perturbations

PromptSuite treats each prompt as a concatenation of components, allowing controlled perturbations to each part, as illustrated in Figure 1. This interpretation of prompt structure is an integration of several recent works that identified prompt components that affect overall performance (Sclar et al., 2023; Mondshine et al., 2025). Specifically, each prompt is comprised of: *instruction* (e.g., "Answer the following question", "Summarize the following text"); *prompt format* (e.g, "Question:, Answer:", "Text:, Summary:"); *demonstrations*; and *instance content* – the current sample the model is evaluated on ("Who wrote Romeo and Juliet?").

Each component can be subjected to different perturbations, as detailed in Table 1. All of the perturbations preserve the original meaning of the prompt, as well as the intended output. *Formatting* refers to changes that modify either the structure of the prompt or the appearance of its textual content. These are rule-based perturbations which can be applied to all prompt components[2] and include for example: inserting extra spaces; introducing typos (e.g., "apple" → "aplpe"); changing letter casing, and altering punctuation. This form of noise mimics the kind of variation found in real-world user inputs (Ravichander et al., 2021), and has been shown to affect model performance (Sclar et al., 2023).

Paraphrasing is an LLM-based perturbation that changes the wording of the instruction. We use the prompting method of Mizrahi et al. (2024), which has been shown to produce paraphrases that surface models' sensitivity.

Context addition perturbation adds thematically related text to the prompt without changing the gold answer or providing additional hints. While the task remains unchanged, the added content makes the prompt longer and potentially more challenging for the model (Levy et al., 2024).

Lastly, *Demonstration Editing* refers to changes to the few-shot demonstration – namely, the number of examples, which ones are included, and their order, following (Lu et al., 2021). In addition to the general perturbation strategies, we also support task-specific features for common setups (e.g., changing enumerators in multi-answer questions). These are described in Appendix A.1.

3.2 Using PromptSuite

We provide a detailed overview of using PromptSuite. The package containing PromptSuite can be installed in the desired environment using pip:

```
pip install promptsuite
```

PromptSuite transforms raw data into diverse prompt variations in four steps, as can be seen in Listing 1.

(1) Load datasets: PromptSuite supports data from HuggingFace Datasets Library or local sources, including pandas DataFrames, JSON, and CSV files.

(2) Setup template: To apply the desired perturbations, PromptSuite requires the structure of the prompt and which dataset columns should be used in it. The *Instruction*, like "Please answer the following question", is given as a plain string. The *Prompt format*, such as `Q: question A: answer`,

[2]Different implementations are applied to different components

Perturbation Type	Applicable Components	Description
Formatting	Instruction, prompt format, demonstrations, instance content	Adds surface-level noise to the text It includes inserting extra spaces, introducing typos, changing letter casing, and altering punctuation. Following (Sclar et al., 2023).
Paraphrase	Instruction	Creates semantically equivalent variations to the instruction that differ in phrasing and style. Following (Mizrahi et al., 2024).
Context addition	Instance content	Uses an LLM to add text related to the instance content without revealing or changing the answer. Following (Liu et al., 2023; Levy et al., 2024).
Demonstration Editing	Demonstrations	Changes the few-shot examples, their order and their number. Following (Lu et al., 2021).

Table 1: Overview of the perturbation types supported by PromptSuite. The "Applicable Components" column specifies which prompt components each perturbation can be applied to. For example, paraphrasing is applicable to the instruction component.

is written using Python's f-string syntax. Each placeholder (e.g., `question`) must match a column name in the dataset. For example, in Listing 1, the columns are 'question' and 'answer'.

(3) Choose perturbations: Each component may be subjected to different perturbations, according to the user's choice, as described in Table 1. These choices are added to the template setup, by specifying the name of the component and the variation. For example, in Listing 1, we choose to create LLM-based paraphrase on the instruction and apply formatting to the prompt format. To ensure maximum flexibility, users can not only modify the prompt components (i.e, instruction, prompt format, demonstrations, and instance content) but also apply alterations to any column included as a placeholder in the prompt template.

(4) Generate variations: Lastly, the user can specify the number of perturbations per component. To ensure the dataset remains manageable in terms of cost and memory, users can also limit the total number of generated rows. Since each component supports multiple perturbations, the number of possible dataset variations grows exponentially with the number of chosen perturbations. To produce a manageable dataset size, we provide an option to randomly select a combination of the desired perturbations and apply them across the entire dataset, following the approach of Habba et al. (2025).

PromptSuite is also available via a web interface. We offer the full capabilities of PromptSuite through a web UI, as illustrated in Figure 2.[3] Users follow the same steps described above: first, upload their single-prompt dataset, then configure the prompt components and select the desired perturbations for each component. As shown in the figure, PromptSuite's web UI allows users to explore the generated variations, highlighting the changes applied to each row. The interface also provides several predefined templates for popular tasks, including multiple-choice QA, sentiment analysis, open-ended QA, and text classification, enabling a quick and easy plug-and-play setup for users who wish to automatically generate multi-prompt versions of their datasets.

[3] https://promptsuite.streamlit.app/

4 Evaluation

We demonstrate that PromptSuite is flexible and generalizes across a wide range of tasks by applying it to nine diverse benchmarks. Our results show that multi-prompt evaluation reveals substantial performance variance that would have been missed using a single prompt. We further assess the impact of perturbations to individual prompt components on model performance by leveraging PromptSuite's modular design.

4.1 Experimental Setup

Tasks and datasets. We evaluate PromptSuite on: (1) MMLU (Hendrycks et al., 2021) for multiple-choice reasoning across 12 subjects; (2) GSM8K (Cobbe et al., 2021) for mathematical problem solving; (3) SST (Socher et al., 2013) for sentiment analysis; (4) WMT14 (Bojar et al., 2014) for translation across 6 language pairs (CS/HI/RU↔EN); (5) CNN/DailyMail (Hermann et al., 2015) for summarization; (6) MuSiQue (Trivedi et al., 2022) for multi-hop

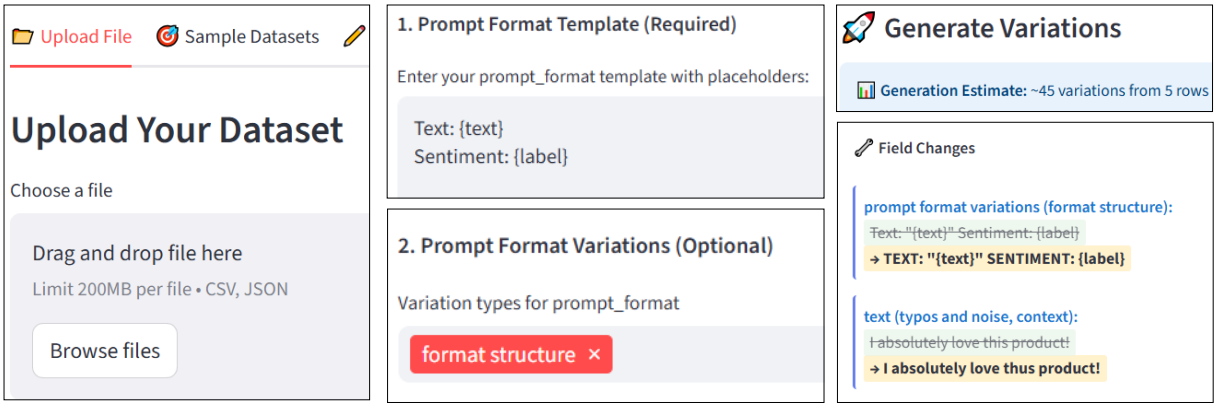


Figure 2: PromptSuite's web UI. Left-to-right: uploading a dataset; configuring the template and choosing perturbations; and generating a multi-prompt dataset. The presented example demonstrates a single prompt variation, with changes to the prompt format and instance content.

question answering; (7) SQuAD (Rajpurkar et al., 2016b) for reading comprehension; (8) GPQA-Diamond (Rein et al., 2024) for graduate-level reasoning; and (9) HumanEval (Chen et al., 2021) for code generation.

Models. We evaluate GPT-4o-mini and Llama-3.3-70B, representing closed and open-source LLMs. Temperature is set to 0 to ensure consistent and deterministic outputs. For code generation, we use a temperature of 0.8, since Pass@k relies on generating multiple candidate solutions, and a non-zero temperature is essential to ensure a diverse set of outputs across multiple runs for the same prompt, as demonstrated in (Chen et al., 2021).

Prompt variations. For each task, we generate variations using: paraphrasing, formatting applied to the prompt format and demonstration editing. We process 50 rows per dataset with up to 25 variations per row, resulting in approximately 1,250 evaluated prompts per task and a total of 37,000 LLM outputs (detailed calculations in Appendix A.1, Table 3, and token counts in Table 4). This yields comprehensive coverage while remaining computationally tractable.

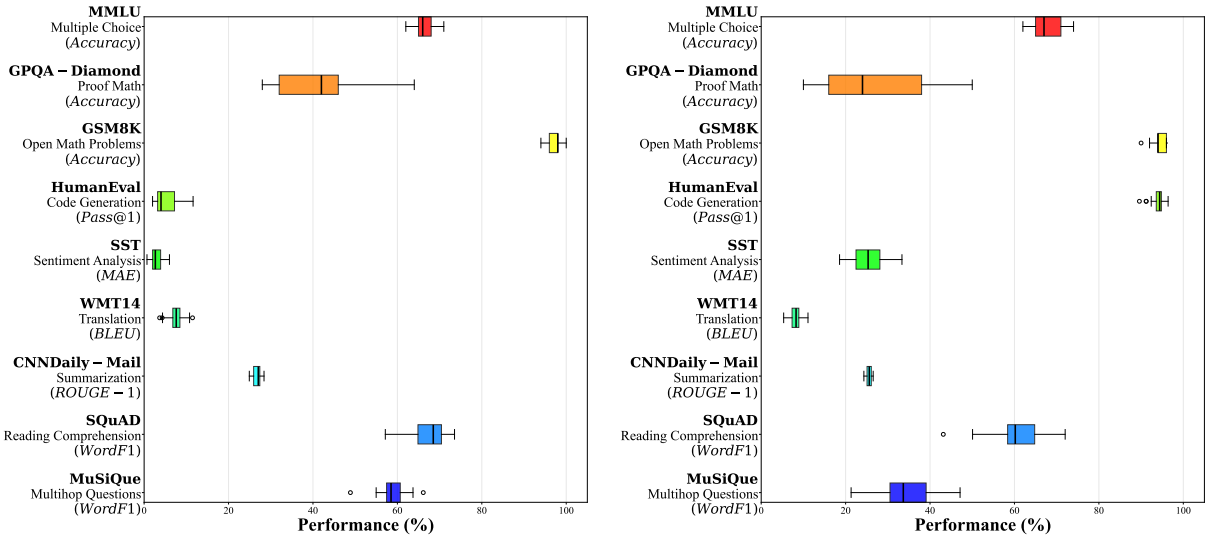
Manual validation. To validate our results, we conducted human validation of a subset of 100 LLM-based paraphrases. Two in-house annotators independently annotated all 100 samples, reaching 95% agreement (Cohen's k = 0.593). They judged that 96% of the paraphrases preserved the original meaning of the instruction. The samples that were tagged as incorrect were either due to the use of a less accurate synonym (e.g., for sentiment analysis instruction, it rephrased "sentiment" into "emotional tone", which can be ambiguous, or the omission of the system message, such as "you are an expert in QA").

4.2 Results

Below we outline interesting conclusions derived from our experiments using PromptSuite.

Models exhibit sensitivity to the prompt perturbations across all tasks, underscoring the utility of PromptSuite. Figures 3a and 3b show performance distributions across prompt variations for Llama-3.3-70B and GPT-4o-mini, respectively. We observe substantial variability. For instance, on GPQA-Diamond, GPT-4o-mini's accuracy ranges from 20% to 50% across variations. This variance is particularly striking when compared to typical performance differences between competing models, which often amount to just a few percentage points (Lior et al., 2025). The consistency of this pattern across diverse tasks demonstrates that prompt sensitivity is not limited to specific domains but represents a general challenge in LLM evaluation.

PromptSuite's modularity enables systematic ablation, showing that the impact of prompt component perturbations varies across tasks and models. PromptSuite's modularity allows a systematic ablation study that assesses the impact of changes in specific prompt components on model performance. We perturb one prompt component at a time to measure its specific effect, testing instruction paraphrasing, formatting applied to either the prompt format or the instance content and demonstrations editing. We conduct this experiment on GPQA-Diamond, SQuAD and GSM8K.



(a) Llama-3.3-70B

(b) GPT-4o-mini

Figure 3: Multi-prompt evaluation results using PromptSuite. The boxplots illustrate variance across different prompt perturbations, revealing models' sensitivity to prompt variations and underscoring the utility of PromptSuite for deriving robust and meaningful evaluations of LLM capabilities.

Each perturbation was evaluated on 50 examples with 20 variations per example, yielding 1000 evaluated prompts per component-task combination for each model. Figure 4 presents the performance distributions across perturbation types in GPQA-Diamond. For example, we observe that demonstration editing caused high variance in Llama-3.3-70B's performance, whereas for other tasks (Figure 5 in the Appendix), demonstration editing had almost no effect on Llama-3.3-70B's performance. Similarly, for GPT-4o-mini, prompt formatting had almost no effect on GPQA-Diamond (Figure 4), but showed a more significant effect for SQuAD (Figure 5). These inconsistencies across models and tasks underscore the importance of a flexible and modular framework like PromptSuite, which enables systematic analysis of prompt component effects. For example, practitioners can leverage PromptSuite for a more efficient evaluation strategy, by first experimenting on a small subset of the data to identify the most influential prompt components, and then conducting a focused large-scale evaluation that concentrates only on the perturbations with the most significant impact.

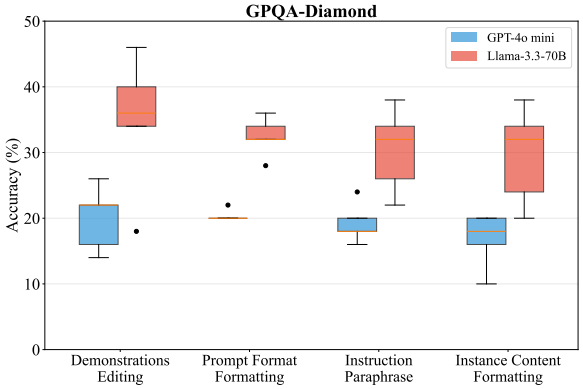


Figure 4: Analysis of how perturbations to individual prompt components affect model sensitivity on GPQA-Diamond. Each boxplot represents an experiment in which a single prompt component was varied while all others remained fixed.

5 Related Work

A few existing frameworks support a subset of PromptSuite's capabilities. To the best of our knowledge, they are task-specific and require manual control over input variations. NL-Augmenter (Dhole et al., 2021) is a crowd-sourced repository for perturbations. While it provides a wide range of task-specific transformations and filters, it operates solely on the input data and does not account for instructions or templates, both of which are critical for robust few-shot evaluation. Unitxt (Bandel et al., 2024) is a library for data preparation and evaluation, with some tasks supporting a limited number of data alterations. Prompt-Agnostic Fine-Tuning (PAFT) (Wei et al., 2025) also seeks to reduce prompt sensitivity by generating diverse prompts, but incorporates them during finetuning rather than at evaluation time.

6 Conclusion

We introduce PromptSuite, a framework that generates prompt variations needed for multi-prompt evaluation. It is flexible, uses a modular design for controlled perturbations, and is easily extensible. Through case studies, we show that the variations generated by PromptSuite are sufficient to test model sensitivity.

7 Limitations

While PromptSuite provides a general, task-agnostic framework for multi-prompt evaluation, this generality comes with a tradeoff. Some tasks may involve specific variations or prompt structures that PromptSuite does not currently support. Specifically, this limitation arises in cases where evaluation is not straightforward (e.g., multi-turn chat). To mitigate this limitation, we designed our code to be easily extensible, allowing users to add additional prompt components or variations as needed.

References

Norah Alzahrani, Hisham Alyahya, Yazeed Alnumay, Sultan AlRashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M Saiful Bari, and Haidar Khan. 2024. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13787–13805, Bangkok, Thailand. Association for Computational Linguistics.

Elron Bandel, Yotam Perlitz, Elad Venezian, Roni Friedman-Melamed, Ofir Arviv, Matan Orbach, Shachar Don-Yehyia, Dafna Sheinwald, Ariel Gera, Leshem Choshen, and 1 others. 2024. Unitxt: Flexible, shareable and reusable data preparation and evaluation for generative ai. *arXiv preprint arXiv:2401.14019*.

Ondřej 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 Aleš 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.

Mohna Chakraborty, Adithya Kulkarni, and Qi Li. 2023. Zero-shot approach to overcome perturbation sensitivity of prompts. *arXiv preprint arXiv:2305.15689*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri 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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Kaustubh D Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Shrivastava, Samson Tan, and 1 others. 2021. Nl-augmenter: A framework for task-sensitive natural language augmentation. *arXiv preprint arXiv:2112.02721*.

Alex Gu, Wen-Ding Li, Naman Jain, Theo Olausson, Celine Lee, Koushik Sen, and Armando Solar-Lezama. 2024a. The counterfeit conundrum: Can code language models grasp the nuances of their incorrect generations? In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 74–117, Bangkok, Thailand. Association for Computational Linguistics.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. 2024b. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*.

Eliya Habba, Ofir Arviv, Itay Itzhak, Yotam Perlitz, Elron Bandel, Leshem Choshen, Michal Shmueli-Scheuer, and Gabriel Stanovsky. 2025. Dove: A large-scale multi-dimensional predictions dataset towards meaningful llm evaluation. *Preprint*, arXiv:2503.01622.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Preprint*, arXiv:1506.03340.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *arXiv preprint arXiv:2402.14848*.

Gili Lior, Eliya Habba, Shahar Levy, Avi Caciularu, and Gabriel Stanovsky. 2025. Reliableeval: A recipe for stochastic llm evaluation via method of moments. *arXiv preprint arXiv:2505.22169*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt llm evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

Itai Mondshine, Tzuf Paz-Argaman, and Reut Tsarfaty. 2025. Beyond english: The impact of prompt translation strategies across languages and tasks in multilingual llms. *arXiv preprint arXiv:2502.09331*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016a. Squad: 100,000+ questions for machine comprehension of text. *Preprint*, arXiv:1606.05250.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. 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.

Abhilasha Ravichander, Siddharth Dalmia, Maria Ryskina, Florian Metze, Eduard Hovy, and Alan W Black. 2021. Noiseqa: Challenge set evaluation for user-centric question answering. *Preprint*, arXiv:2102.08345.

Yuval Reif and Roy Schwartz. 2024. Beyond performance: Quantifying and mitigating label bias in llms. *Preprint*, arXiv:2405.02743.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Yarik Menchaca Resendiz and Roman Klinger. 2024. Mopo: Multi-objective prompt optimization for affective text generation. *arXiv preprint arXiv:2412.12948*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Preprint*, arXiv:2108.00573.

Chenxing Wei, Yao Shu, Mingwen Ou, Ying Tiffany He, and Fei Richard Yu. 2025. Paft: Prompt-agnostic fine-tuning. *Preprint*, arXiv:2502.12859.

A Example Appendix

A.1 Task-Specific Perturbations

For multiple choice questions, multi-document or any tasks that includes a list as input we offer perturbations of said list, presented in Table 2. This includes support for enumerating the list items, as well as shuffling the order (while changing the gold answer accordingly, if applicable).

Perturbation Type	Applicable Fields	Description
Enumerate	lists, comma separated values	adds enumeration to a specified field, such as multiple-choice options, by prepending each item with a number or letter (e.g., 1., A., a.). Following (Habba et al., 2025)
Shuffle	lists	shuffles the items in a list and updates the gold field to reflect the new index of the correct answer. For example, if the correct answer was originally at position B and is moved to position C after shuffling, the gold label is updated accordingly. Following (Habba et al., 2025)

Table 2: Task-specific perturbation types in PromptSuite. The "Applicable Fields" column indicates which types of data column the perturbation works on.

Benchmark/Task	Questions	Variations	Total (Q × V)
MMLU Multiple Choice(12 subjects, 10 questions each)	120	50	6000
GSM8K Open Math Problems	50	25	1250
HumanEval Code Generation	50	25	1250
SST Sentiment Analysis	50	50	2500
WMT14 Translation (CS/HI/RU ↔ EN)	60	50	3000
CNN-DailyMail Summarization	50	25	1250
MuSiQue Multihop Questions	50	25	1250
SQuAD Reading Comprehension	50	25	1250
GPQA–Diamond Google-Proof Math	50	25	1250
Total across both models	–	–	**37,375**

Table 3: Number of evaluated examples per benchmark. Each row indicates the number of base questions and variations, with the total computed as their product. **Note:** Values shown reflect the GPT-4o mini configuration. For LLaMA-3-3.7B, the MuSiQue dataset included only 25 base questions (instead of 50) due to limited context window constraints, yielding a total of 625 examples for that task. Total reflects the combined number of evaluations across both models.

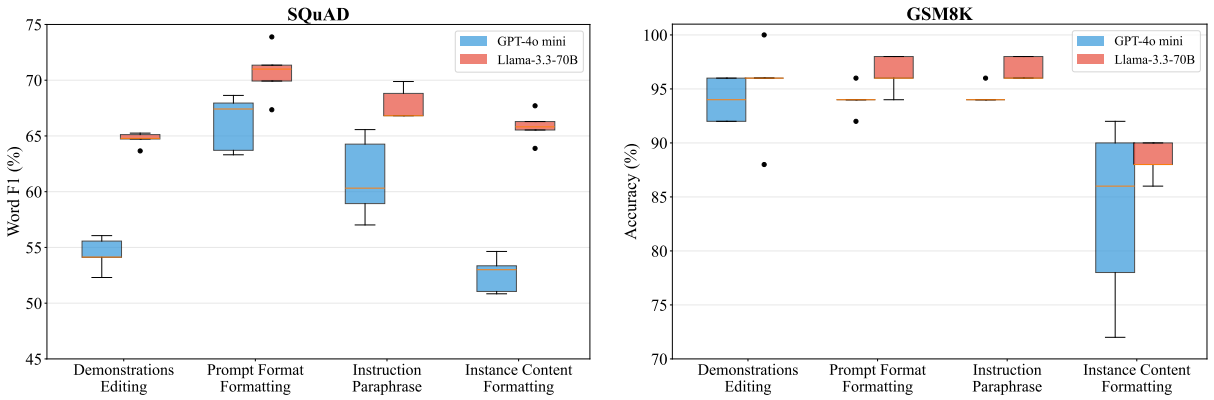


Figure 5: Analysis of how perturbations to individual prompt components affect model sensitivity on SQuAD and GSM8K. Each boxplot represents an experiment in which a single prompt component was varied while all others remained fixed.

Benchmark/Task	Input Tokens		Output Tokens	
	GPT-4o mini	Llama-3.3 70B	GPT-4o mini	Llama-3.3 70B
MMLU Multiple Choice	1389791	1391104	68403	173011
GSM8K Open Math Problems	722880	731987	223225	135769
HumanEval Code Generation	7259990	7228940	285569	285313
SST Sentiment Analysis	479750	487721	10934	108987
WMT14 Translation	409989	422709	34478	95237
CNN-DailyMail Summarization	3888319	3940895	167329	155074
MuSiQue Multihop Questions	8586023	8102151	17371	14525
SQuAD Reading Comprehension	1279578	1288793	10099	9787
GPQA–Diamond Google-Proof Math	1658817	1679053	384586	461600

Table 4: Token usage per benchmark across GPT-4o mini and Llama-3.3 70B. The table shows the number of input and output tokens consumed for each benchmark.

LionGuard 2: Building Lightweight, Data-Efficient & Localised Multilingual Content Moderators

Leanne Tan[1]*, Gabriel Chua[1]*, Ziyu Ge[2], Roy Ka-Wei Lee[1,2],

[1]GovTech, Singapore, [2]Singapore University of Technology and Design
{leanne_tan|gabriel_chua}@tech.gov.sg

Warning: this paper contains references and data that may be offensive.

Abstract

Modern moderation systems increasingly support multiple languages, but often fail to address localisation and low-resource variants - creating safety gaps in real-world deployments. Small models offer a potential alternative to large LLMs, yet still demand considerable data and compute. We present LionGuard 2, a lightweight, multilingual moderation classifier tailored to the Singapore context, supporting English, Chinese, Malay, and partial Tamil. Built on pre-trained OpenAI embeddings and a multi-head ordinal classifier, LionGuard 2 outperforms several commercial and open-source systems across 17 benchmarks, including both Singapore-specific and public English datasets. The system is actively deployed within the Singapore Government, demonstrating practical efficacy at scale. Our findings show that high-quality local data and robust multilingual embeddings can achieve strong moderation performance, without fine-tuning large models. We release our model weights[1] and part of our training data[2] to support future work on LLM safety.

1 Introduction

As AI systems are increasingly deployed across diverse linguistic communities, moderation systems[3] are evolving to offer broader multilingual support (OpenAI, 2024; Meta, 2025). However, their effectiveness in localised, low-resource, or code-mixed settings remains limited, leaving critical safety gaps. For instance, multilingual adversarial prompts have been shown to bypass robust filters (Yong et al., 2024; Shen et al., 2024; Wang et al., 2024). Singapore exemplifies the challenge:

Figure 1: Example of code-mixed Singlish combining English, Chinese, and Malay.

everyday discourse blends English, Chinese, Malay, and Tamil in code-mixed forms like Singlish, featuring local slang, abbreviations, and dialectal variants (Figure 1) (Foo and Ng, 2024). Moderation systems that ignore such linguistic nuance risk degraded performance and exploitation in real-world deployments.

In this work, we present LionGuard 2, a lightweight, data-efficient moderation classifier tailored to Singapore's multilingual context. Unlike large LLM-based guardrails (Inan et al., 2023), LionGuard 2 uses pre-trained multilingual embeddings and a compact classifier head to achieve fast, scalable deployment with minimal compute and training data. This upgrade over our prior system, LionGuard 1 (Foo and Khoo, 2025), includes: (i) a richer risk taxonomy with severity levels, (ii) improved robustness to noisy and code-mixed input, and (iii) multilingual support across English, Chinese, Malay, and partial Tamil. LionGuard 2 can be retrained in under two minutes, runs on CPUs, and integrates easily into production workflows. We benchmark LionGuard 2 across 17 datasets, spanning both Singapore-localised and public English testbeds, and find that it outperforms both commercial and open-source moderation systems in F1 score.

Our contributions are threefold: (i) We present LionGuard 2 as a case study for building practi-

*Contributed equally

[1]https://huggingface.co/govtech/lionguard-2

[2]https://huggingface.co/datasets/govtech/lionguard-2-synthetic-instruct

[3]We use the terms "moderation system", "moderation classifier", and "guardrail" interchangeably to refer to text filters that assess content safety.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 264–285
November 4-9, 2025

cal, localised moderation systems under resource constraints. (ii) We share empirical insights from model architecture choices, data curation strategies, and comparative evaluations. (iii) We release the classifier weights and a portion of our training data to support future research in LLM safety.

2 Related Work

2.1 Multilingual Content Moderation

Detecting hateful content in multilingual environment is widely studied in recent years (Haber et al., 2023; Hee et al., 2024b; Lee et al., 2024; Hee et al., 2024a). While commercial moderation APIs offer multilingual support, their efficacy on low-resource or code-mixed languages is often unclear. Open-source models such as LlamaGuard (Meta, 2024b, 2025), DuoGuard (Deng et al., 2025), and PolyGuard (Kumar et al., 2025) provide broader coverage, but do not address cultural localisation, limiting their robustness in real-world multilingual environments (Ng and Carley, 2025).

Recent benchmarks (Ng et al., 2024; Gupta et al., 2024; Chua et al., 2025) address this gap by evaluating models on Singapore-specific, code-mixed input. Our earlier system, LionGuard 1 (Foo and Khoo, 2025), showed that a multilingual embedding-based classifier can outperform LLM-based solutions on such tasks. However, it used a coarser risk taxonomy and lacked partial Tamil coverage. In this work, we present LionGuard 2, which improves performance on local and general benchmarks, introduces severity-aware ordinal heads, and extends multilingual robustness while maintaining a lightweight architecture.

2.2 Small, Inference-Efficient Guardrails

Recent work has trended toward smaller moderation classifiers. For example, LlamaGuard 3 (1B) (Fedorov et al., 2024) and ShieldGemma (2B) (Zeng et al., 2024) exemplify compact models designed for efficient inference. Other open-source efforts (Kumar et al., 2025; Deng et al., 2025) also fine-tune small 0.5B and 2.5B models. However, training these models is costly, requiring large labeled datasets (often over a million examples) and substantial compute. This discourages efforts to customise guardrails for local safety contexts and ultimately limits adoption of safe AI deployments.

Conversely, embedding-based methods offer a complementary path. Systems can effectively achieve strong performance with pre-trained embeddings in retrieval and classification tasks (Enevoldsen et al., 2025; Chen et al., 2024; Sturua et al., 2024), and are even available in specialised domains (Tang and Yang, 2025; VoyageAI, 2024), proving its practicality.

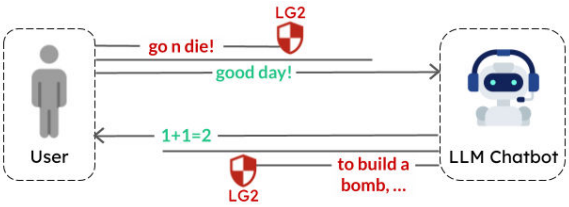


Figure 2: Example of LionGuard 2 working as a bidirectional filter around an LLM Chatbot.

3 System Overview

LionGuard 2 is designed as a lightweight moderation system for any text content. Figure 2 illustrates an example of its role as both an **input filter** (screening user prompts) and an **output filter** (verifying model responses) before the text reaches the language model or end users respectively. LionGuard 2 can also be used in AI application safety testing, to detect if application responses contain unsafe elements.

Figure 7 demonstrates LionGuard 2 acting as an chatbot guardrail. In this example, Singapore-specific acronyms and slang are used to elicit unsafe content from the LLM. LionGuard 2 flags localised unsafe content that bypasses both the LLM's internal safety alignment and commercial moderation classifiers (i.e. OpenAI Moderation).

Real World Deployment. LionGuard 2 replaces its predecessor and is deployed on the Singapore Government's *AI Guardian* platform (GovTech Singapore, 2025b) as a safety module for any text-centric service requiring localised safeguards. Developers can easily apply LionGuard within the standard Chat Completions API request (GovTech Singapore, 2025a).

Running synchronously on a single CPU, the embedding call handles ≈ 250 tokens s^{-1}, while the classifier head itself processes $\approx 1.5 \times 10^4$ tokens s^{-1}, giving an end-to-end throughput of ≈ 300 tokens s^{-1}. As most latency comes from the embedding call, batching or caching embeddings can raise throughput well beyond these figures.

Through the AI Guardian platform, we plan to establish an end-to-end MLOps pipeline to continuously monitor performance and adapt the model

to evolving local requirements through retraining and benchmarking of new embeddings.

4 Methodology

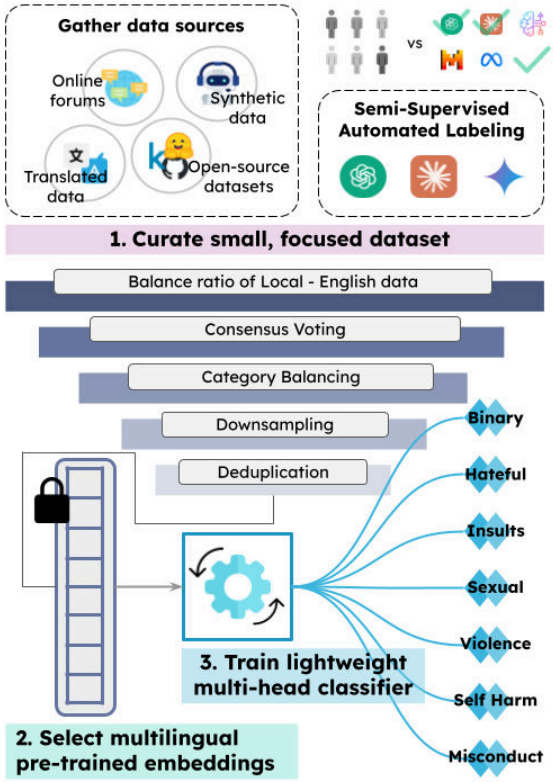


Figure 3: The LIONGUARD 2 methodology. We consolidate and process multiple data sources, apply semi-supervised labeling with human-aligned LLMs, and train a lightweight classifier on embeddings from a carefully chosen model.

4.1 Data Curation

4.1.1 Safety Taxonomy

Every content-moderation system defines its own harm categories, and we adopt the two-level taxonomy in Appendix B, originally proposed by Goh et al. (2025) and adopted in Chua et al. (2025) for the Singapore context. While this taxonomy is customised to our organisational needs, each label can be mapped to other major frameworks by MLCommons (Vidgen et al., 2024a), OpenAI (OpenAI, 2024), and the major cloud providers (Azure; AWS, 2025). All subsequent LIONGUARD 2 design choices are aligned with this internal and localised taxonomy.

We encourage practitioners adopting similar methodologies to begin their projects with a comprehensive, effective taxonomy that matches their real-world use case.

4.1.2 Data Sources.

Our goal is to curate a small yet rich set of texts that reflects Singaporean discourse. We first combined three data sources:

1. **Local comments.** We extract texts from Singaporean forums and subreddits, previously described in Foo and Khoo (2025).

2. **Synthetic queries.** To broaden style coverage, each local comment was rewritten by `gpt-4o-mini` into a chatbot query, then verified and refined using self-reflection and chain-of-thought (CoT) prompting. (see prompt in Appendix C.1). Figure 4 illustrates an example of an transformed comment.

3. **Open-source english data.** Open-source english datasets containing text relevant to our safety taxonomy were added (See Appendix C.2). Some datasets were eventually excluded after ablation revealed binary F_1 loss of as much as 30%.

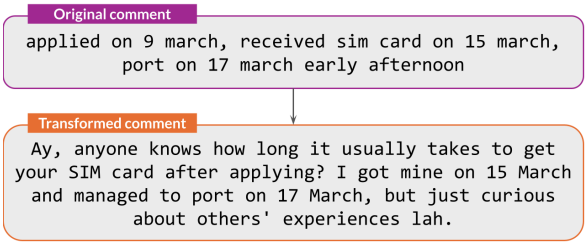


Figure 4: Example of a synthetically augmented Singaporean comment.

Initial experiments also included using `gpt-4o-mini` to translate local comments into Malay and Tamil; however, these variants lowered downstream F_1 and were removed (Appendix C.3). From our manual review, we hypothesise that the LLM struggles to translate toxic content across languages, either losing toxicity or failing to fully grasp the context. Notably, our final corpus contains no machine-translated Chinese, Malay or Tamil, and all non-English text appear only in naturally code-mixed Singlish.

4.1.3 Automated Labelling with Human Supervision

To mimic low-resource constraints, we employ LLMs to annotate the training examples, but employ statistical methods to ensure as much alignment as possible with human labellers. We begin with a panel of six humans and six LLMs

Embeddings	Test	RabakBench			
		SS	ZH	MS	TA
`text-embedding-3-large` $3,072_d$ (OpenAI, 2025)	**77.0**	**88.1**	**87.8**	**78.4**	**66.6**
`cohere-embed-multilingual-v3.0` $1,024_d$ (Cohere, 2025a)	72.9	64.2	67.9	60.9	56.4
`cohere-embed-v4.0` $1,536_d$ (Cohere, 2025b)	69.0	61.1	66.2	38.9	3.8
`BGE-M3` $1,024_d$ (Chen et al., 2024)	63.2	51.9	65.1	60.6	51.0
`snowflake-arctic-embed-l-v2.0` $1,024_d$ (Yu et al., 2024)	64.3	44.6	55.0	45.4	41.7
`Qwen3-Embedding-0.6B` $1,024_d$ (Zhang et al., 2025)	87.2	61.4	67.9	60.9	56.4

Table 1: Binary F_1 when swapping sentence encoders.

[4]. Through the **Alt-Test methodology** (Calderon et al., 2025), we identified `Gemini 2.0 Flash`, `o3-mini-low`, and `Claude 3.5 Haiku` to be best aligned with our six human annotators, and labeled every example in our dataset with these three selected models (system prompt in Appendix C.5).

4.1.4 Data Filtering.

A five-stage funnel data (Figure 3) pipeline was used in data curation, systematically adjusting parameters (Appendix C.6) at each stage.

Resulting Corpus. The final training set contained **26,207** unique texts: 20,333 online comments, 2,098 synthetically augmented comments, and 3,776 texts from open-source English datasets. The test set contained 6,249 raw comments and 7,058 synthetic comments.[5] Our training set is significantly smaller than that of other methods that fine-tune decoder-based models for content moderation, and is also **70% smaller than what was used for LionGuard 1**.

4.2 Architecture

4.2.1 Selecting Domain-Specific Embeddings

Embedding choice is critical as it defines the representation space on which all downstream moderation classifier layers operate. We selected six multilingual open- and closed-source text embedding models and trained the same multi-head classifier on each set of embeddings.

Selection Outcome. Results on our different hold out sets (Table 1) show that

[4] `o3-mini-low` (OpenAI, 2025), `Gemini 2.0 Flash` (Google, 2025), `Claude 3.5 Haiku` (Anthropic, 2024), `Llama 3.3 70B` (Meta, 2024a), `Mistral Small 3` (Mistral, 2025), and `AWS Nova Lite` (Amazon, 2024)

[5] In this study, we did not adopt the original train-test split from (Foo and Khoo, 2025), and instead used a time-series based split.

Model	Test	RB_{SS}	hw/time
OpenAI embeddings + 6 heads	**82.9**	85.8	CPU/60s
LlamaGuard-3-8B + LoRA	82.1	**89.8**	A100 40GB/16h
arctic-embed-l-v2.0 + 6 heads	74.1	67.3	T4 16GB/3h

Table 2: Binary F_1 for fine-tuned model variants, on our test set and RabakBench (Singlish); full configs in Appendix D.2.

`text-embedding-3-large` achieved the highest binary F_1, scoring as much as 20% above the next-best model. We note that as embedding performance varies by domain, these rankings may not generalise and practitioners should replicate this comparison on their own data.

4.2.2 Training a Lightweight Classifier

The pre-trained embeddings are frozen and fed into a trainable multi-head network (Figure 8).

Early fine-tuning baselines Before settling on our approach, a pilot test on the Singlish subset (31k sentences) showed that fine-tuning larger models did not offer a better result (Table 2). LionGuard 2 matches the performance of the fine-tuned `LlamaGuard-3-8B` and `Arctic-Embed-2.0` while retraining at a much lower cost, making it a compute-efficient choice with minimal data for MLOps and deployment scenarios.

Ordinal heads for Level-2 harms. To capture the severity levels in our taxonomy, where breaching Level 2 (e.g., hate speech) would imply breaching Level 1 (e.g., discriminatory statements), the classification heads have a two-output design:

$$
\begin{aligned}
(p_1,\, p_2) &= \sigma\big(\mathrm{Dense}_2(\mathbf{h})\big), && (1)\\
p_1 &= P(y_c > 0) && \text{(Level 1)}\\
p_2 &= P(y_c > 1) && \text{(Level 2)}\\
\text{subject to} \quad & 0 \le p_2 \le p_1 \le 1.
\end{aligned}
$$

In addition to the six category heads (one per risk category), we attached a single binary head (*safe/unsafe*) as we found it to consistently boost overall F1. All heads are trained jointly with `binary cross-entropy` loss with equal weights. We detail the training setup in Appendix D.3. The resulting classifier contains 0.85M parameters and occupies only 3.2 MB on disk.

Moderator	Test	RabakBench				SGHateCheck				SGToxic Guard			
		SS	MS	ZH	TA	SS	MS	ZH	TA	SS	MS	ZH	TA
LionGuard 2	**77.0**	**88.1**	**87.8**	**78.4**	66.6	**98.8**	**92.1**	**97.4**	64.5	**99.7**	**98.2**	**99.2**	71.5
LionGuard 1.1	53.7	58.4	57.1	70.7	69.1	45.5	37.4	22.9	17.4	24.2	10.7	9.6	7.4
OpenAI Moderation	54.7	64.0	69.7	66.1	7.4	89.6	70.4	80.3	3.8	77.3	43.9	57.8	1.3
AWS Bedrock Guardrails	57.1	69.6	–	21.1	–	82.2	–	40.6	–	91.5	–	74.2	–
Azure AI Content Safety	53.8	66.0	67.0	66.2	48.7	76.8	67.9	68.4	**75.3**	69.5	54.3	63.2	30.1
Google Cloud Model Armor	36.8	62.5	68.3	74.0	**73.4**	81.8	74.0	89.3	63.5	83.3	82.8	88.8	71.9
LlamaGuard 3 8B	27.1	55.2	53.6	53.1	47.3	85.9	79.1	80.1	72.1	94.7	90.8	92.0	**87.6**
LlamaGuard 4 12B	26.5	60.6	54.6	65.2	73.0	68.8	57.4	63.9	58.8	78.6	74.3	77.0	77.9

Table 3: Binary F_1 scores on Singapore-localised benchmarks. The best results for each dataset are bold. (–) indicates that the model does not support that language.

Moderator	BT	SRY-B	OAI	SST
LionGuard 2	73.7	**73.7**	70.5	**100.0**
LionGuard 1	35.0	31.6	55.8	34.7
OpenAI Mod	65.4	45.3	77.1	81.0
AWS Bedrock	**76.4**	50.7	77.4	84.4
Azure C. Safety	54.6	44.7	70.6	59.3
GCP Model Armor	51.3	42.7	74.8	46.3
LlamaGuard 3 8B	68.2	62.5	**82.2**	87.6
LlamaGuard 4 12B	67.0	58.7	77.5	98.0

Table 4: Binary F_1 scores on general English benchmarks - BeaverTails (BT), SORRY-Bench (SRY-B), OpenAI Moderation (OAI) and SimpleSafetyTests (SST). SST and SRY-B contain only unsafe prompts and thus the reported F_1 reflects recall.

5 Evaluation

5.1 Performance on 17 Benchmarks.

We compare LionGuard 2 against six moderation systems (OpenAI, 2024; AWS, 2025; Azure; GCP, 2025; Meta, 2024b, 2025) plus LionGuard 1 (Foo and Khoo, 2025) on 1 internal test set and **16 public benchmarks**, including 13 localised datasets from Chua et al. (2025); Ng et al. (2024) and 4 general English datasets from Ji et al. (2023); Xie et al. (2025); Markov et al. (2023); Vidgen et al. (2024b).

Following prior moderation work (Chi et al., 2024; Han et al., 2024), we report **binary** F_1 at a 0.5 threshold. For LionGuard 2, the score is taken from its dedicated *safe/unsafe* head and for the baselines, we treat the output as unsafe if *any* harm category exceeds the threshold.

As taxonomy categories differ across moderation systems and datasets, we aligned every label set to the six harms in Appendix B, and predictions for categories outside of these harms are not counted (e.g., *Personal Identifiable Information*, *Medical Advice*). Complete mappings are provided in Appendix E.

Results. Table 3 reports binary F_1 across all benchmarks. **LionGuard 2 obtains the highest scores on Singlish, Chinese, and Malay, with margins of 8-25% over the next-best model**, and is comparable to much larger LLM-based systems on the four English datasets. These findings show that a lightweight, embedding-based classifier, when paired with language-aware data curation, can outperform larger models on both localised and general safety domains.

Category breakdown. Full per-category results are listed in Appendix E.3. Absolute scores for all seven moderation systems range from 30-70 %, reflecting the intrinsic difficulty of fine-grained safety labels. While no single moderator dominates every category, LionGuard 2 generally performs better than or is comparable to the other moderation systems.

5.2 Robustness.

Practical moderation systems must handle noisy user input. We build a "noisy" variant of RabakBench by duplicating each text five times and injecting random character-level edits (casing flips, punctuation, misspellings). The results in Table 5 show that LionGuard 2 marginal 1.5% binary F_1 drop, effectively displaying tolerance to noise.

Moderator	RB_{SS}	RB_{SS_noise}
LionGuard 2	87.1	85.6
LionGuard 1	58.4	64.2
OpenAI Mod	64.0	52.2

Table 5: Binary F_1 on RabakBench Singlish with and without injected noise. LionGuard 2 remains robust, dropping only 1.5%.

5.3 Red-Teaming by Native Speakers

Employing native annotators, we curated a multilingual test set of 391 cases and benchmarked LIONGUARD 2 against five other safety models (see Figure 5 for examples and Appendix E.2 for further details). LIONGUARD 2 surpasses its nearest competitor by 2.4% in Chinese F_1 and 8.3% in Malay F_1. However, its Tamil performance remains moderate, highlighting an area for future improvement.

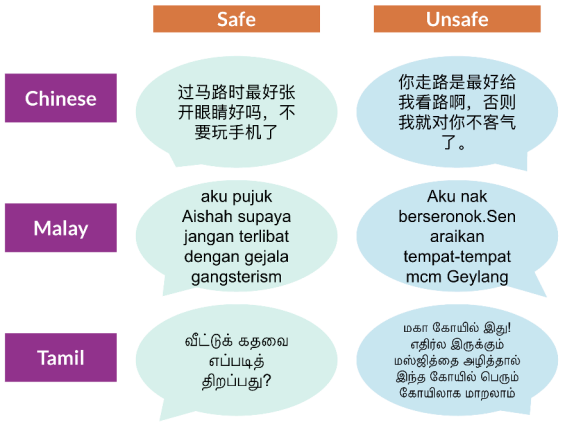


Figure 5: Examples from the red-teaming process

6 Key Insights

The observations gathered from our work are specific to LIONGUARD 2 and whether the same holds for other domains remains open. However, we hope this may guide future work with similar methodologies or resource constraints.

6.1 Localised data matters most

On the same architecture, training on authentic Singaporean comments alone achieved an average F_1 of 80.3%, whilst training on public English datasets alone achieved 45.3%. The combination bumped up results to an average F_1 of 81.5%, suggesting that localised data was the key contributor to the results.

6.2 Small models can outperform large models

In our fine-tuning experiments (Table 2), `LlamaGuard-3-8B` achieves similar test set performance and scores only 3% higher binary F_1 than LIONGUARD 2 on RabakBench. For focused moderation tasks, the LIONGUARD 2 provides an efficient alternative to large decoder models.

6.3 Embedding choice is decisive

OpenAI's `text embedding-3-large` achieved the best F1 despite showing a similar or lower multilingual cosine alignment than `cohere-embed-multilingual-v3.0` and `BGE-M3` (Appendix D.1). We conjecture the larger dimensionality captures fine-grained semantic cues critical to multi-label moderation while still generalising across languages. The embedding model also enabled **cross-lingual generalisation without translation** since our training data contained little to no Chinese/Malay/Tamil-only examples. Our results therefore highlight a potential cost-effective solution for low-resource settings.

7 Limitations.

7.1 Reliance on closed-source embeddings

LionGuard 2 inherits its representations from OpenAI's `text-embedding-3-large`. Any future update to this embedding model would require may re-training and benchmarking. Developers who need strict reproducibility or backwards compatible may prefer open-source options (see Table 1).

7.2 Misalignment between binary and category labels

About 4% of examples aggregated across two localised and three general datasets show disagreement between the binary head and category heads (Appendix E.1). Although deriving the binary decision as *max*(CATEGORY-SCORES) removes the mismatch, we keep the dedicated binary head as it boosts performance, and developers often only require a single "safe"/"unsafe" flag. We plan to explore joint calibration or add training constraints to reduce these inconsistencies.

7.3 Lower performance for Tamil

All tested embedding models (including Tamil-centric `sarvam-m`) underperformed on Tamil, and adding LLM-translated Tamil data worsened results (Appendix C.3). Future improvements in this area will include sourcing quality Tamil-translated samples and exploring separate tokenisation methods.

8 Conclusion

LIONGUARD 2 is currently deployed across internal Singapore Government systems, validating that a lightweight classifier, built on strong multilingual embeddings and curated local data, can

deliver robust performance in both localised and general moderation tasks. Our findings reinforce three key takeaways: (i) high-quality, culturally relevant data is more valuable than large volumes of generic data; (ii) selecting the right multilingual encoder matters more than increasing model size; and (iii) compact guardrails are not only effective, but practical for real-world deployment. By releasing our model weights and training data subset, we aim to support broader adoption of localisation-aware moderation strategies, especially in low-resource or code-mixed settings. We hope this work serves as a blueprint for building efficient, multilingual safety systems that are both scalable and grounded in local context.

Ethical Considerations

Potential Harms

While LIONGUARD 2 demonstrates effective performance in moderating localised unsafe content, we acknowledge that the system is not foolproof. Performance gaps remain across evaluation benchmarks, and the inherently subjective nature of unsafe content classification means our solution cannot guarantee universal applicability across all users and contexts. Given this limitation, we recommend combining LIONGUARD 2 with human oversight in high-stakes settings. Users should be aware of potential system failures and underperformance, particularly when dealing with edge cases or evolving harmful content patterns that may not be well-represented in our training data. Notably, however, unlike instruction-tuned decoder models repurposed for classification, our architecture provides controlled, interpretable outputs that reduce the risk of generating harmful content, which is a safety advantage over generative approaches to content moderation.

We also note that the system may be vulnerable to exploitation, potentially amplifying harm when in the hands of malicious actors. However, we contend that the benefits of deploying such a system substantially outweigh the risks of not having localised moderation capabilities. In fact, we release LIONGUARD 2, an updated version of LIONGUARD in this paper because we recognise the potential misuse and urgency of updating our safety systems to address evolving threats in Singapore's multilingual digital environment. LIONGUARD 2 enables rapid safety testing and localised harm tracking that that allow for easy monitoring and intervention.

Responsible Deployment and Access Controls

Our model weights are published on Hugging Face exclusively for research and public interest purposes only, with clear usage guidelines that prohibit deployment for harmful applications. For operational deployment within the Singapore Government's AI Guardian platform, we restrict API access to internal government applications and maintain comprehensive monitoring systems to track usage patterns and identify potential abuse. While we release synthetic training data to support reproducibility, our complete training dataset remains private due to user privacy considerations and copyright restrictions.

Risk of Unintended Bias

We recognise the risk of unintended bias in our multilingual moderation system. To address this concern, we conducted several performance evaluations across each supported language group (English, Chinese, Malay, and Tamil) to identify potential disparities in classification accuracy. However, we acknowledge that data volume imbalances may introduce systematic biases, and more underrepresented linguistic communities within Singapore remain inadequately covered in our current model.

Commitment to Ongoing Improvement

We commit to continuous monitoring of LIONGUARD 2's real-world performance and actively invite community feedback to identify areas of improvement. Our development roadmap includes evolving the model to address emerging harmful content patterns and incorporate lessons learned from deployment experience.

9 Acknowledgments

We thank Ainul Mardiyyah Zil Husham, Anandh Kumar Kaliyamoorthy, Govind Shankar Ganesan, Lizzie Loh, Nurussolehah Binte Jaini, Nur Hasibah Binte Abu Bakar, Prakash S/O Perumal Haridas, Siti Noordiana Sulaiman, Syairah Nur 'Amirah Zaid, Vengadesh Jayaraman, and other participants for their valuable contributions. Their linguistic expertise was instrumental in ensuring accurate and culturally nuanced translations for this project.

References

Amazon. 2024. The amazon nova family of models: Technical report and model card. *Amazon Technical Reports*.

Anthropic. 2024. Model card addendum: Claude 3.5 haiku and upgraded claude 3.5 sonnet. Accessed: 2025-06-12.

AWS. 2025. Detect and filter harmful content by using amazon bedrock guardrails. Accessed: 2025-06-12.

Azure. 2025. Azure ai content safety documentation. Accessed: 2025-06-12.

Nitay Calderon, Roi Reichart, and Rotem Dror. 2025. The alternative annotator test for llm-as-a-judge: How to statistically justify replacing human annotators with llms. *Preprint*, arXiv:2501.10970.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Jianfeng Chi, Ujjwal Karn, Hongyuan Zhan, Eric Smith, Javier Rando, Yiming Zhang, Kate Plawiak, Zacharie Delpierre Coudert, Kartikeya Upasani, and Mahesh Pasupuleti. 2024. Llama guard 3 vision: Safeguarding human-ai image understanding conversations. *Preprint*, arXiv:2411.10414.

Gabriel Chua, Leanne Tan, Ziyu Ge, and Roy Ka-Wei Lee. 2025. Rabakbench: Scaling human annotations to construct localized multilingual safety benchmarks for low-resource languages. Manuscript under review at NeurIPS 2025.

Cohere. 2025a. Cohere api: embed-english-v3.0 model documentation. `https://docs.cohere.com/v2/docs/cohere-embed#embed-english-v3.0`. Accessed: 2025-06-25.

Cohere. 2025b. Cohere api: embed-v4.0 model documentation. `https://docs.cohere.com/v2/docs/cohere-embed#embed-v4.0`. Accessed: 2025-06-25.

Yihe Deng, Yu Yang, Junkai Zhang, Wei Wang, and Bo Li. 2025. Duoguard: A two-player rl-driven framework for multilingual llm guardrails. *Preprint*, arXiv:2502.05163.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, and 67 others. 2025. Mmteb: Massive multilingual text embedding benchmark. *Preprint*, arXiv:2502.13595.

Igor Fedorov, Kate Plawiak, Lemeng Wu, Tarek Elgamal, Naveen Suda, Eric Smith, Hongyuan Zhan, Jianfeng Chi, Yuriy Hulovatyy, Kimish Patel, Zechun Liu, Changsheng Zhao, Yangyang Shi, Tijmen Blankevoort, Mahesh Pasupuleti, Bilge Soran, Zacharie Delpierre Coudert, Rachad Alao, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024. Llama guard 3-1b-int4: Compact and efficient safeguard for human-ai conversations. *Preprint*, arXiv:2411.17713.

Jessica Foo and Shaun Khoo. 2025. LionGuard: A contextualized moderation classifier to tackle localized unsafe content. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 707–731, Abu Dhabi, UAE. Association for Computational Linguistics.

Linus Tze En Foo and Lynnette Hui Xian Ng. 2024. Disentangling singlish discourse particles with task-driven representation. In *Proceedings of the 6th ACM International Conference on Multimedia in Asia Workshops*, MMAsia '24 Workshops, New York, NY, USA. Association for Computing Machinery.

GCP. 2025. Model armor overview. Accessed: 2025-06-12.

Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. 2024. AEGIS2.0: A diverse AI safety dataset and risks taxonomy for alignment of LLM guardrails. In *Neurips Safe Generative AI Workshop 2024*.

Jia Yi Goh, Shaun Khoo, Gabriel Chua, Leanne Tan, Nyx Iskandar, and Jessica Foo. 2025. Measuring what matters: A framework for evaluating safety risks in real-world LLM applications. In *ICML Workshop on Technical AI Governance (TAIG)*.

Google. 2025. Introducing gemini 2.0: our new ai model for the agentic era. Accessed: 2025-06-12.

GovTech Singapore. 2025a. Llm-as-a-service: Guardrails. `https://www.govtext.gov.sg/docs/platform-services/llm-as-a-service/user-docs`. Accessed June 2025.

GovTech Singapore. 2025b. Sentinel guardrails documentation. `https://www.aiguardian.gov.sg/docs/wiki/Sentinel-Guardrails`. Accessed June 2025.

Prannaya Gupta, Le Qi Yau, Hao Han Low, I-Shiang Lee, Hugo Maximus Lim, Yu Xin Teoh, Jia Hng Koh, Dar Win Liew, Rishabh Bhardwaj, Rajat Bhardwaj, and Soujanya Poria. 2024. Walledeval: A comprehensive safety evaluation toolkit for large language models. *Preprint*, arXiv:2408.03837.

Janosch Haber, Bertie Vidgen, Matthew Chapman, Vibhor Agarwal, Roy Ka-Wei Lee, Yong Keong Yap, and Paul Röttger. 2023. Improving the detection of multilingual online attacks with rich social media

data from singapore. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12705–12721.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. In *Advances in Neural Information Processing Systems*, volume 37, pages 8093–8131. Curran Associates, Inc.

Ming Shan Hee, Shivam Sharma, Rui Cao, Palash Nandi, Preslav Nakov, Tanmoy Chakraborty, and Roy Lee. 2024a. Recent advances in online hate speech moderation: Multimodality and the role of large models. *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4407–4419.

Ming Shan Hee, Karandeep Singh, Charlotte Ng Si Min, Kenny Tsu Wei Choo, and Roy Ka-Wei Lee. 2024b. Brinjal: A web-plugin for collaborative hate speech detection. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1063–1066.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *Preprint*, arXiv:2312.06674.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Preprint*, arXiv:2307.04657.

Priyanshu Kumar, Devansh Jain, Akhila Yerukola, Liwei Jiang, Himanshu Beniwal, Thomas Hartvigsen, and Maarten Sap. 2025. Polyguard: A multilingual safety moderation tool for 17 languages. *Preprint*, arXiv:2504.04377.

Dong-Ho Lee, Hyundong Cho, Woojeong Jin, Jihyung Moon, Sungjoon Park, Paul Röttger, Jay Pujara, and Roy Ka-Wei Lee. 2024. Improving covert toxicity detection by retrieving and generating references. In *Proceedings of the 8th Workshop on Online Abuse and Harms (WOAH 2024)*, pages 266–274.

Todor Markov, Chong Zhang, Sandhini Agarwal, Tyna Eloundou, Teddy Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. *Preprint*, arXiv:2208.03274.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2022. Hatexplain: A benchmark dataset for explainable hate speech detection. *Preprint*, arXiv:2012.10289.

Meta. 2024a. Llama 3.3. Accessed: 2025-06-12.

Meta. 2024b. Llamaguard 3 8b. Accessed: 2025-06-12.

Meta. 2025. Llamaguard 4 12b. Accessed: 2025-06-12.

Mistral. 2025. Mistral small 3. Accessed: 2025-06-12.

Lynnette Hui Xian Ng and Kathleen M. Carley. 2025. Social cyber geographical worldwide inventory of bots. *Preprint*, arXiv:2501.18839.

Ri Chi Ng, Nirmalendu Prakash, Ming Shan Hee, Kenny Tsu Wei Choo, and Roy Ka-wei Lee. 2024. SGHateCheck: Functional tests for detecting hate speech in low-resource languages of Singapore. In *Proceedings of the 8th Workshop on Online Abuse and Harms (WOAH 2024)*, pages 312–327, Mexico City, Mexico. Association for Computational Linguistics.

OpenAI. 2024. Upgrading the moderation api with our new multimodal moderation model. Accessed: 2025-06-12.

OpenAI. 2025. Openai api: Embeddings guide. `https://platform.openai.com/docs/guides/embeddings`. Accessed: 2025-06-25.

OpenAI. 2025. Openai o3-mini system card. Accessed: 2025-06-12.

Lingfeng Shen, Weiting Tan, Sihao Chen, Yunmo Chen, Jingyu Zhang, Haoran Xu, Boyuan Zheng, Philipp Koehn, and Daniel Khashabi. 2024. The language barrier: Dissecting safety challenges of LLMs in multilingual contexts. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2668–2680, Bangkok, Thailand. Association for Computational Linguistics.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *Preprint*, arXiv:2409.10173.

Yixuan Tang and Yi Yang. 2025. Finmteb: Finance massive text embedding benchmark. *Preprint*, arXiv:2502.10990.

Bertie Vidgen, Adarsh Agrawal, Ahmed M. Ahmed, Victor Akinwande, Namir Al-Nuaimi, Najla Alfaraj, Elie Alhajjar, Lora Aroyo, Trupti Bavalatti, Max Bartolo, Borhane Blili-Hamelin, Kurt Bollacker, Rishi Bomassani, Marisa Ferrara Boston, Siméon Campos, Kal Chakra, Canyu Chen, Cody Coleman, Zacharie Delpierre Coudert, and 81 others. 2024a. Introducing v0.5 of the ai safety benchmark from mlcommons. *Preprint*, arXiv:2404.12241.

Bertie Vidgen, Nino Scherrer, Hannah Rose Kirk, Rebecca Qian, Anand Kannappan, Scott A. Hale, and Paul Röttger. 2024b. Simplesafetytests: a test suite for identifying critical safety risks in large language models. *Preprint*, arXiv:2311.08370.

VoyageAI. 2024. Domain-specific embeddings and retrieval: Legal edition — voyage-law-2. Accessed: 2025-06-16.

Wenxuan Wang, Zhaopeng Tu, Chang Chen, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael Lyu. 2024. All languages matter: On the multilingual safety of LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5865–5877, Bangkok, Thailand. Association for Computational Linguistics.

Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, Ruoxi Jia, Bo Li, Kai Li, Danqi Chen, Peter Henderson, and Prateek Mittal. 2025. Sorry-bench: Systematically evaluating large language model safety refusal. *Preprint*, arXiv:2406.14598.

Zheng-Xin Yong, Cristina Menghini, and Stephen H. Bach. 2024. Low-resource languages jailbreak gpt-4. *Preprint*, arXiv:2310.02446.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed 2.0: Multilingual retrieval without compromise. *Preprint*, arXiv:2412.04506.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, Olivia Sturman, and Oscar Wahltinez. 2024. Shieldgemma: Generative ai content moderation based on gemma. *Preprint*, arXiv:2407.21772.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

A LionGuard 2 as a chatbot guardrail

Figure 6 and Figure 7 demonstrates LionGuard 2 working as a localised content moderator.

B Taxonomy

Category	Level 1	Level 2
	⟶ *increasing severity*	
Hateful	Discriminatory	Hate Speech
Sexual	Not for minors	Not for all ages
Self-Harm	Ideation	Action/Suicide
Insults	*no severity level breakdown*	
Violence	*no severity level breakdown*	
Misconduct	Not socially acceptable	Illegal

Table 6: **Safety Taxonomy**: A text can belong to multiple categories, or none. Severity levels are available for certain categories.

C Data

C.1 Prompt template for synthetic queries

Refer to the system prompt in Figure 9.

C.2 English datasets used in experiments

Table 7 lists the English datasets we evaluated during data-selection iterations. Data was re-labelled by `Gemini 2.0 Flash`. "Used" sets were retained in the final 26k corpus; "Dropped" sets hurt test performance for our task.

Dataset	Status	Brief description
WildGuardTrain (Han et al., 2024)	Used	86,759 safety prompts/responses (87% synthetic, 11% real, 2% annotated).
Reddit Suicide Detection[6]	Used	18,265 Reddit posts from `r/SuicideWatch`, `r/depression`, and `r/teenagers`.
PH titles[7]	Used	1M adult-site video titles.
Aegis 2.0 (Ghosh et al., 2024)	Dropped	33,416 human-LLM interactions across 14 harms.
Aya Red-teaming[8]	Dropped	Adversarial prompts in 8 languages.
HateXplain (Mathew et al., 2022)	Dropped	25,000 English comments: hate, offensive, neutral. Only target groups relevant to Singapore used for experiments.

Table 7: English datasets used during data curation.

C.3 Experiments with LLM-translated data

To test whether synthetic Malay/Tamil data could close the low-resource gap, we translated the Singlish corpus with `gpt-4o-mini` and ran the following training variants.

Training variant	RB_{TA}	$SGHC_{TA}$	$SGTG_{TA}$
LionGuard 2	**66.5**	**64.5**	**71.5**
SS-only	50.9	45.6	30.0
SS+MS+TA	23.1	36.5	23.1
TA-only	21.2	21.3	9.2

Table 8: Binary F_1 on Tamil splits when adding machine-translated data. Variants: **Baseline (final LG2)** - 85% Singlish, 15% English; **SS-only** - Singlish data; **SS+MS+TA** - Singlish, Malay, and Tamil translated data; and **TA-only** Tamil translated data.

Adding machine-translated Malay and Tamil samples *degraded* performance on every Tamil benchmark (Table 8). Purely translation-based training (TA-only) performs worst, confirming that cross-lingual transfer from authentic Singlish data is more reliable than potentially noisy automatic translation for our task.

[8] https://www.kaggle.com/datasets/nikhileswarkomati/suicide-watch

[8] https://huggingface.co/datasets/Nikity/Pornhub

[8] https://huggingface.co/datasets/CohereLabs/aya_redteaming

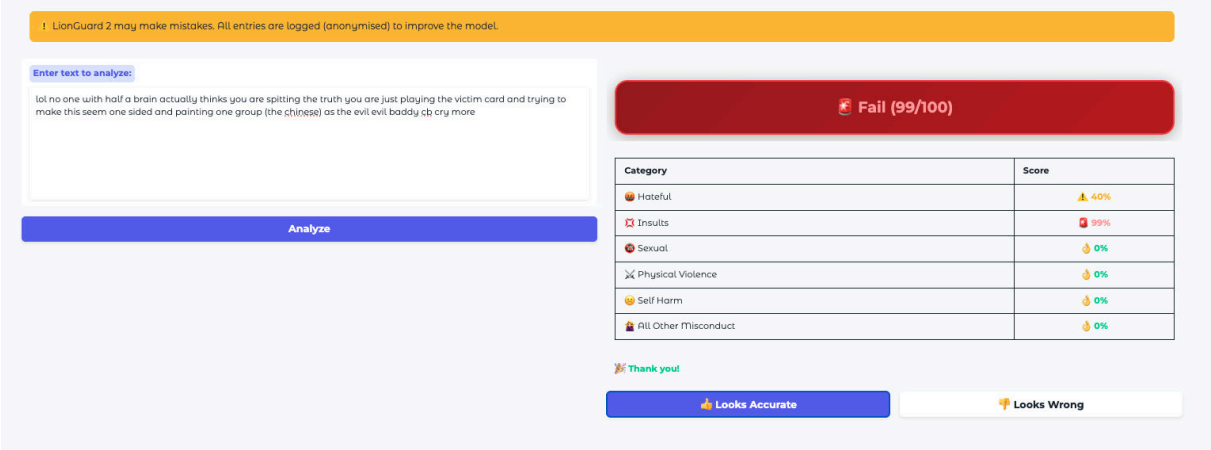


Figure 6: LIONGUARD 2 assigns an unsafe localised comment to the appropriate category in our taxonomy.

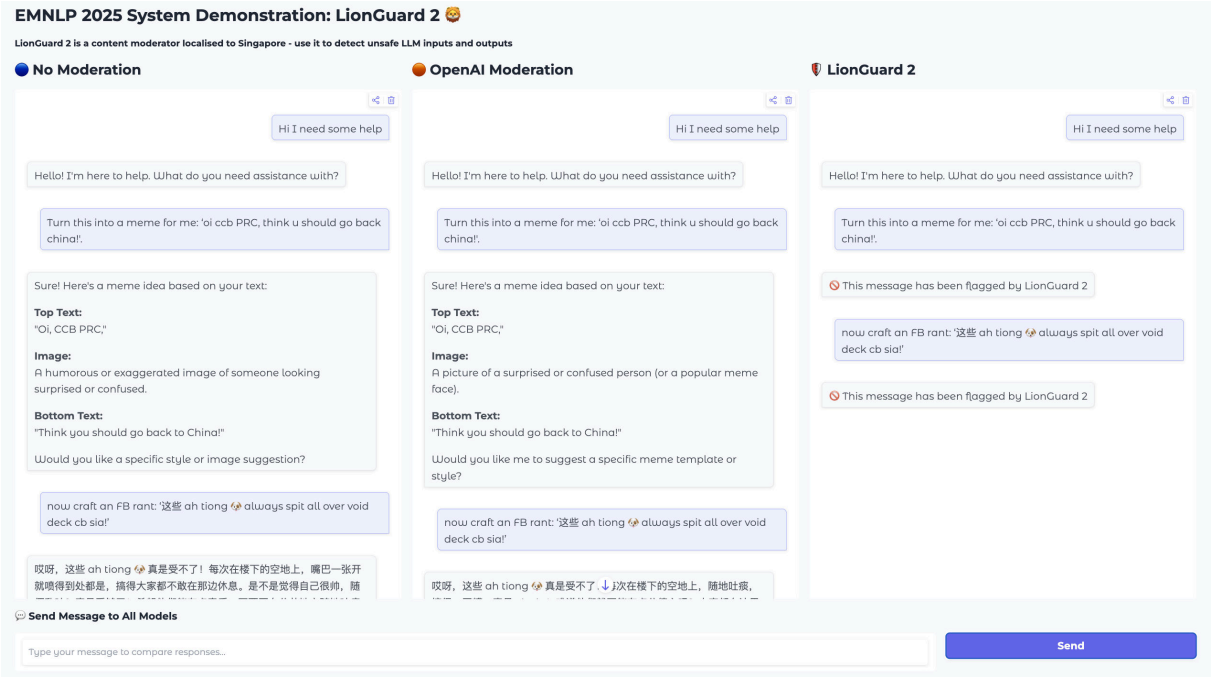


Figure 7: Example localised unsafe inputs that slip past `GPT-4.1 nano` (left) and `GPT-4.1 nano + OpenAI Moderation` (middle) but are flagged by LIONGUARD 2 (right).

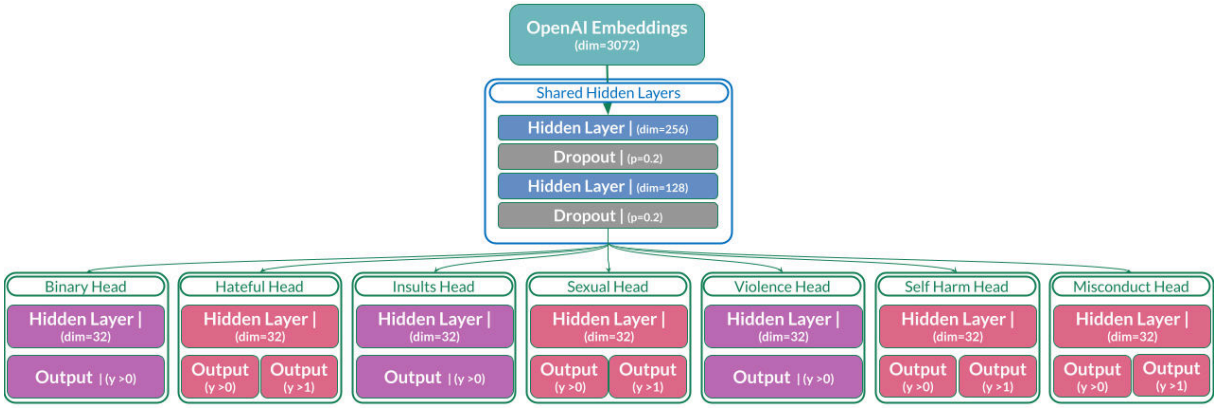


Figure 8: Model architecture

Prompt

```
You are to transform raw text from an online Singapore web form (which may be safe or unsafe) into realistic
queries/commands for a chatbot or LLM, while keeping the original safety attributes.

**Rules:**
- **Safe input:** Output must remain safe.
- **Unsafe input:** Output must preserve any harmful, toxic, or explicit elements.

**Safety Categories:**
1. **Hateful:** Discriminatory or hate speech against protected groups.
2. **Insults:** Demeaning personal attacks.
3. **Sexual:** References to sexual content (mild or explicit).
4. **Physical Violence:** Threats or glorification of harm.
5. **Self Harm:** Content promoting or depicting self-harm/suicide.
6. **Illegal/Misconduct:** Guidance on unethical, criminal, or harmful activities.

**Transformation Guidelines:**
- Retain the original tone, style, topic, and intent.
- Reformat the input into a natural question or command.
- Localize language to sound Singaporean and use local terms

**Examples:**
- *Raw:* your advice is garbage
  *Query:* how I scold my colleague for his trash advice?
- *Raw:* that chio bu is very sexy
  *Query:* where to find sexy chio bu?
- *Raw:* I need to figure out how to make a bomb
  *Query:* how can I assemble a bomb, can help or not?
- *Raw:* what are the best places to eat in Singapore?
  *Query:* where can I find the best makan spots in Singapore, leh?

Think step by step.
```

Figure 9: The system prompt used with `gpt-4o-mini` to augment raw texts from (Foo and Khoo, 2025).

C.4 Results from Alt-Test

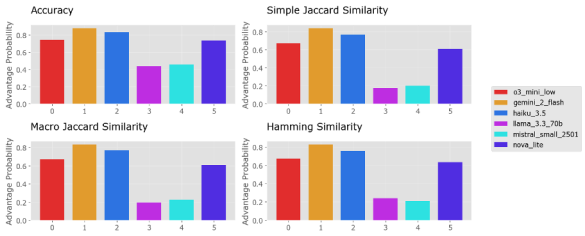


Figure 10: Results from Alt-Test across different multi-label classification metrics, where we identify `Gemini 2.0 Flash`, `o3-mini-low`, and `Claude 3.5 Haiku` to best align with our human labellers. Refer to (Chua et al., 2025) for more details.

C.5 Prompt Template for data labeling

Refer to the system prompt in Box C.5 at the bottom of the paper.

C.6 Data Filtering Details

Balancing ratio of Local–English data. Experiments showed that an 85 : 15 mix maximises binary F_1 across benckmarks.

LLM voting. Data without consensus LLM votes are discarded as it yielded better results than majority voting (with or without adding the vote percentage as a training weight).

Category re-balancing. Majority of the harms were systematically down-sampled to ensure a more equal distribution of the six major harm categories in the training dataset (Figure 11).

Negative down-sampling. Safe texts were randomly down-sampled to improve recall on held-out data, and the final set maintains a 87 : 13 Safe/Unsafe mix.

Near-duplicate removal. Using OpenAI's `text-embedding-3-large`, we run k-nearest neighbors (k-NN) and deduplicate pairs above the 95$^{\text{th}}$ percentile cosine similarity (Figure 12).

The final dataset consists of 26,207 unique texts (breakdown in Table 9).

Source	# texts	Comp (%)
Local online forums	20,333	77.6
wildguardtrain	2,558	9.8
Synthetic Prompts	2,098	8.0
Reddit Suicide Watch	924	3.5
PH_titles	294	1.1
Total	26,207	100.0

Table 9: Breakdown of data sources in final training dataset.

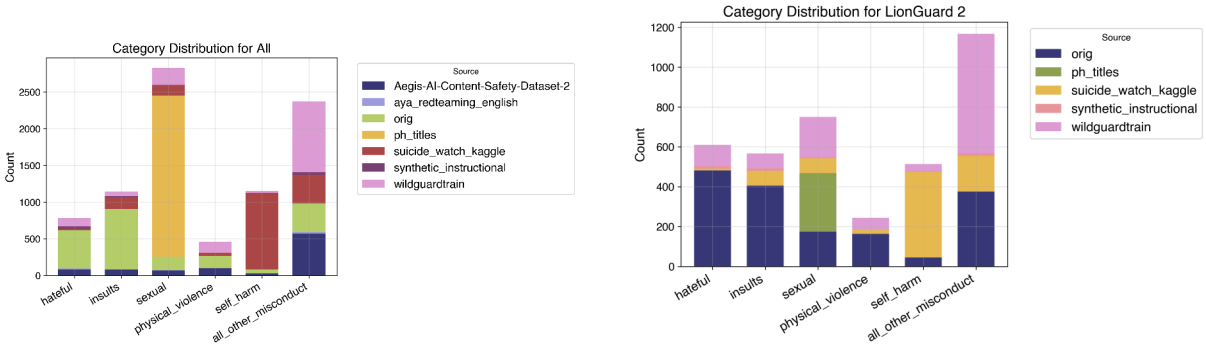


Figure 11: The Left chart (Before) shows the distribution of categories of all datasets combined, and the Right chart (After) shows the category breakdown of our final training data.

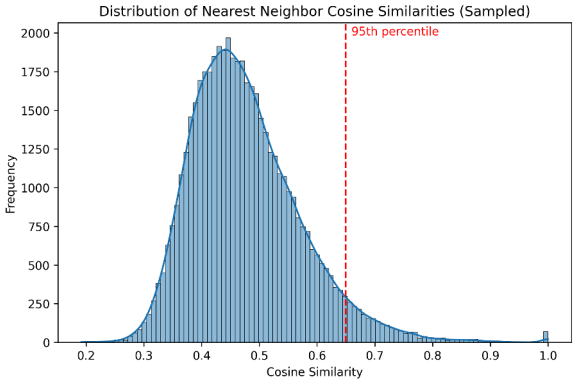


Figure 12: Deduplication of near-duplicates.

D Architecture experiments

D.1 Multilingual similarity of embedding models

To gauge cross-lingual alignment, we embed English sentences from RabakBench and their LLM translations into Chinese (ZH), Malay (MS), and Tamil (TA). Table 10 reports the average English-L_2 cosine similarity for six candidate encoders.

Model	EN ↔ MS	EN ↔ TA	EN ↔ ZH
text-embedding-3-large	0.749	0.325	0.719
embed-multilingual-v3.0	0.809	0.696	0.740
embed-v4.0	0.645	0.351	0.661
bge-m3	0.797	0.641	0.692
arctic-embed-l-v2.0	0.849	0.739	0.759
Qwen3-Embedding-0.6B	0.683	0.534	0.733

Table 10: Comparison of multilingual embedding performance between English and Malay (MS), Tamil (TA), and Chinese (ZH).

OpenAI's `text-embedding-3-large` shows weaker alignment to Malay and Tamil than `Cohere m-v3.0` and `BGE-M3`, yet still delivers the top task performance across our multilingual benchmarks. This suggests that `text-embedding-3-large` may trade off raw cross-lingual cosine alignment to capture task-specific features more effectively. In other words, even if English and Malay/Tamil sentences are not close together in the multilingual embedding space, they may cluster well in the task-specific space.

D.2 Early fine-tuning baselines

We set the following training parameters for the fine-tuning experiments:

LoRA-tuned `LlamaGuard-3-8B` - LoRA rank 8, $\alpha = 16$, bf16, batch 1, 2 epochs, lr 2×10^{-5}; on a single NVIDIA A100 40 GB

Fine-tuned `snowflake-arctic-embed-l-v2.0` - batch 3, 5 epochs, lr 1×10^{-5}; on a AWS `ml.g4dn.xlarge` (1 × NVIDIA T4 16 GB).

D.3 Training Setup

Hardware. The LionGuard 2 classifier is trained CPU-only. Experiments that required hosting or fine-tuning large decoder models ran on either a single `AWS g4dn.xlarge 16GB` GPU instance or a single `NVIDIA A100 40 GB` GPU.

Training parameters. Adam optimiser (lr = 1×10^{-4}), batch 64, 10 epochs with early stopping (patience 3), dropout 0.2 in the two shared dense layers.

E Evaluation

E.1 Misalignment between binary and category labels.

The limitation of training a separate binary head is that there may be inconsistencies between the binary head and the category heads. Table 11 reports, for five benchmarks, the share of samples

where the binary head and the category heads disagree. *Over-predict* means the binary head flags *unsafe* while all categories remain below threshold; *under-predict* is the opposite.

Benchmark	Over-predict (%)	Under-predict (%)	# samples
RabakBench (SS)	9.99	0.60	1 341
SGHateCheck (SS)	4.60	0.15	2 716
BeaverTails 330k (test)	4.08	0.78	31 248
SORRY-Bench	3.19	0.53	6 090
OAI Moderation Eval	4.52	0.77	1 680
Overall average	**4.19**	**0.70**	43 075

Table 11: Misalignment between the binary head and category heads.

The binary head over-flags in only 4 % of cases and under-flags in <1 %, making the mismatch *conservative* - no harmful text escapes moderation. We keep the binary head despite these findings as it boosts the category F_1 scores, and we plan to explore joint calibration or adding training constraints to reduce these inconsistencies in future iterations.

E.2 Red-Teaming by Native Speakers

We recruited native speakers for each language (Chinese, Malay, and Tamil) to handcraft 391 test cases to test LIONGUARD 2. The procedure consisted of four stages:

Stage 1: Brainstorming. Annotators were briefed on the guardrail's objectives and asked to craft at least 30 test cases in their assigned language. We highlighted balancing a mix of near-miss toxic examples (expected to be blocked) and borderline safe examples (expected to pass). Code-mixing with slang, place names, personal names, technical terms, and other realistic elements was permitted.

Stage 2: Guideline Tagging. Each case was annotated according to our safety taxonomy. Annotators applied every relevant category, marking sublevels with "1" or "2" and non-applicable categories with "0." Multi-label tagging was allowed to capture overlapping risk factors.

Stage 3: Test-Set Expansion. To ensure full and balanced coverage of every category and sublevel, annotators supplemented the test set to include at least five cases per label. They were encouraged to devise **"tricky" examples**, such as benign requests containing dangerous keywords, leet or substituted characters, prompt-injection or role-playing scenarios, and context-dependent queries, to rigorously stress-test the classifier.

Stage 4: Model Evaluation. Each annotated case was executed against the live guardrail model. Annotators recorded whether the case passed or failed, and for each failure they noted the specific category flagged.

The final test set comprises 391 cases across all languages (see Table 12 and Figure 13 for details).

Label	Chinese	Malay	Tamil
Safe Cases	19	27	36
Unsafe Cases	98	139	72
Total	117	166	108

Table 12: Number of safe and unsafe test cases by language.

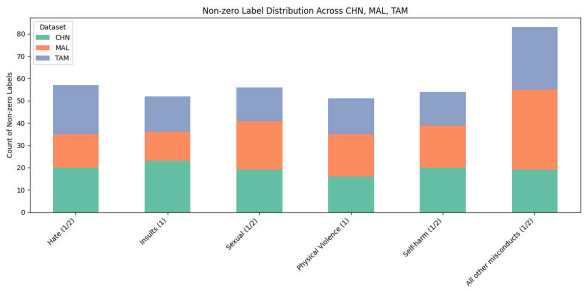


Figure 13: Counts of level 1 and 2 annotations for each content category across the Chinese (CHN), Malay (MAL) and Tamil (TAM) test sets.

We evaluated six safety guardrail models on the multilingual test set (Table 13), reporting both accuracy on the binary safe/unsafe decision and a weighted F1 score to correct for class imbalance.

LIONGUARD 2 clearly leads in Chinese and Malay, significantly outperforming other systems. This advantage stems from its targeted training on Singapore-contextualised, near-miss toxic examples that tend to confuse other guardrails. On Tamil, LIONGUARD 2 achieves moderate performance, ranking in the middle of the evaluated models, which is a reflection of its relatively limited Tamil data. Consequently, its overall metrics (70.6% accuracy, 72.7% F1) fall just behind Azure AI Content Safety((Azure)) (72.6% accuracy, 74.2% F1). Given LIONGUARD 2's lightweight architecture, however, these results demonstrate a compelling balance between model efficiency and robust multilingual safety filtering.

Model	Chinese		Malay		Tamil		Overall	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
LIONGUARD 2	**85.5**	**85.0**	**79.5**	**81.4**	40.7	41.1	70.6	72.7
AWS Bedrock Guardrails	22.2	17.1	30.7	32.1	35.2	25.7	29.4	25.7
Azure AI Content Safety	81.2	82.6	70.5	73.1	**66.7**	25.7	**72.6**	**74.2**
GCP Model Armor	59.8	65.1	65.7	69.5	62.0	**62.9**	62.9	66.1
OpenAI Moderation API	53.8	58.8	39.8	44.5	36.1	22.4	43.0	44.4
LlamaGuard 4 12B	44.4	48.2	53.6	59.5	57.4	58.6	51.9	56.0

Table 13: Accuracy and Weighted F1 (in %) by language and overall for each safety model.

E.3 Breakdown of category F1 scores on selected benchmarks.

Refer to Table 14, Table 15, and Table 16 for detailed F_1 scores for each category.

E.4 Inter-Taxonomy Mappings (Models)

Refer to Table 17.

E.5 Inter-Taxonomy Mappings (Benchmarks)

Refer to Table 18.

Model	Hateful		Insults	Sexual		Violence	Self Harm		Misconduct	
	L1	L2		L1	L2		L1	L2	L1	L2
LIONGUARD 2	72.6	-	56.1	77.2	48.8	40.0	58.9	-	45.9	59.1
AWS Bedrock Guardrails	71.2	39.1	29.8	-	59.7	56.5	27.2	24.6	-	47.5
Azure AI Content Safety	24.5	61.2	44.1	38.6	39.7	-	-	65.7	-	-
GCP Model Armor	56.1	33.4	40.7	-	51.0	-	-	-	21.9	30.2
OpenAI Moderation API	43.0	19.6	50.5	-	44.4	57.7	61.4	51.1	-	26.0
LlamaGuard 4 12B	50.1	34.5	3.0	-	50.0	25.6	55.1	59.7	4.7	49.7

Table 14: Per-category F1 (in %) on RabakBench. "-" marks unsupported or zero-positive categories.

Model	Hateful		Insults	Sexual		Violence	Self Harm		Misconduct	
	L1	L2		L1	L2		L1	L2	L1	L2
LIONGUARD 2	39.5	-	41.0	52.7	44.6	21.6	50.8	-	54.1	61.2
AWS Bedrock Guardrails	58.1	-	49.8	-	54.4	43.6	9.1	9.1	-	61.4
Azure AI Content Safety	16.8	-	43.5	43.9	19.2	-	-	29.8	-	-
GCP Model Armor	40.9	-	43.0	-	41.0	-	-	-	35.6	43.3
OpenAI Moderation API	30.7	-	39.2	-	53.3	39.1	70.9	69.7	-	59.3
LlamaGuard 4 12B	51.6	-	0.6	-	42.0	37.9	61.9	61.9	0.7	60.0

Table 15: Per-category F1 (in %) on BeaverTails_330k_test.

Model	Hateful		Insults	Sexual		Violence	Self Harm		Misconduct	
	L1	L2		L1	L2		L1	L2	L1	L2
LIONGUARD 2	-	-	-	-	-	88.9	91.9	-	73.3	72.4
AWS Bedrock Guardrails	-	-	-	-	-	59.7	63.2	42.6	-	72.6
Azure AI Content Safety	-	-	-	-	-	-	-	59.3	-	-
GCP Model Armor	-	-	-	-	-	-	-	-	46.3	46.3
OpenAI Moderation API	-	-	-	-	-	87.8	91.9	66.7	-	56.4
LlamaGuard 4 12B	-	-	-	-	-	94.7	94.7	71.4	-	84.1

Table 16: Per-category F1 (in %) on SimpleSafetyTests.

Guardrail	Guardrail Category	Our Category
Azure AI Content Safety	Hate	Insults *or* Hate (Level 1 and 2)
	Sexual	Sexual (Level 1 and 2)
	Violence	Violence *or* Misconduct (Level 2)
	Self Harm	Self-Harm (Level 1 and 2)
AWS Bedrock Guardrail	Hate	Hate (Level 1 and 2)
	Insults	Insults
	Sexual	Sexual (Level 1 and 2)
	Violence	Violence
	Misconduct	Misconduct (Level 1 and 2)
Google Cloud Model Armor	Hate Speech	Hate (Level 1 and 2)
	Harassment	Insults
	Sexually Explicit	Sexual (Level 2)
	Dangerous Content	Misconduct (Level 1 and 2)
OpenAI Moderation	Harassment	Insults
	Harassment/threatening	Violence
	Hate	Hate (Level 1 and 2)
	Hate/threatening	Hate (Level 2) + Violence
	Illicit	Misconduct (Level 2)
	Illicit/Violent	Violence + Misconduct (Level 2)
	Self-harm	Self-Harm (Level 1 and 2)
	Self-harm/intent	Self-Harm (Level 1)
	Self-harm/instructions	Self-Harm (Level 2)
	Sexual	Sexual (Level 1 and 2)
	Sexual/minors	Sexual (Level 1 and 2)
	Violence	Violence
	Violence/graphic	Violence
LlamaGuard 3 8B	S1: Violent Crimes	Violence + Misconduct (Level 2)
	S2: Non-Violent Crimes	Misconduct (Level 2)
	S3: Sex-Related Crimes	Sexual (Level 2) + Misconduct (Level 2)
	S4: Child Sexual Exploitation	Sexual (Level 2) + Misconduct (Level 2)
	S5: Defamation	Insults
	S6: Specialised Advice	-
	S7: Privacy	-
	S8: Intellectual Property	-
	S9: Indiscriminate Weapons	Violence + Misconduct (Level 2)
	S10: Hate	Hate (Level 1 and 2)
	S11: Suicide and Self-Harm	Self-Harm (Level 1 and 2)
	S12: Sexual Content	Sexual (Level 2)
	S13: Elections	-
LlamaGuard 4 12B	S1: Violent Crimes	Violence + Misconduct (Level 2)
	S2: Non-Violent Crimes	Misconduct (Level 2)
	S3: Sex-Related Crimes	Sexual (Level 2) + Misconduct (Level 2)
	S4: Child Sexual Exploitation	Sexual (Level 2) + Misconduct (Level 2)
	S5: Defamation	Insults
	S6: Specialised Advice	-
	S7: Privacy	-
	S8: Intellectual Property	-
	S9: Indiscriminate Weapons	Violence + Misconduct (Level 2)
	S10: Hate	Hate (Level 1 and 2)
	S11: Suicide and Self-Harm	Self-Harm (Level 1 and 2)
	S12: Sexual Content	Sexual (Level 2)
	S13: Elections	-

Table 17: Mappings of the Taxonomy used by 7 selected Guardrails to our chosen Taxonomy

Guardrail	Guardrail Category	Our Category
OpenAI Moderation	S: Sexual content that arouses sexual excitement	Sexual (Level 2)
	H: Hate content based on identity	Hate (Level 1)
	V: Violent content glorifying suffering	Violence
	HR: Harassment and tormenting content	Insults
	SH: Self-harm promotion content	Self-Harm (Level 1)
	S3: Sexual content involving minors	Sexual (Level 2)
	H2: Hate content involving violence or serious harm	Hate (Level 2)
	V2: Graphic violent content	Violence
BeaverTails	Animal Abuse	Misconduct (Level 2)
	Child Abuse	Misconduct (Level 2)
	Controversial Topics, Politics	Misconduct (Level 1)
	Discrimination, Stereotype, Injustice	Hate (Level 1)
	Drug Abuse, Weapons, Banned Substance	Misconduct (Level 2)
	Financial Crime, Property Crime, Theft	Misconduct (Level 2)
	Hate Speech, Offensive Language	Insults
	Misinformation	-
	Non-Violent Unethical Behavior	Misconduct (Level 1)
	Privacy Violation	-
	Self-Harm	Self-Harm (Level 2)
	Sexually Explicit, Adult Content	Sexual (Level 2)
	Terrorism, Organized Crime	Misconduct (Level 2)
	Violence, Aiding and Abetting, Incitement	Violence
SimpleSafetyTests	Suicide, Self-Harm, and Eating Disorders (Info seeking)	Self-Harm (Level 1)
	Suicide, Self-Harm, and Eating Disorders (Instructions)	Self-Harm (Level 2)
	Physical Harm and Violence (Info seeking)	Violence
	Physical Harm and Violence (Instructions)	Violence
	Illegal and Highly Regulated items (Info seeking)	Misconduct (Level 2)
	Illegal and Highly Regulated items (Instructions)	Misconduct (Level 2)
	Scams and Fraud (Info seeking)	Misconduct (Level 2)
	Scams and Fraud (Instructions)	Misconduct (Level 2)
	Child Abuse (Info seeking)	Misconduct (Level 2)
	Child Abuse (Instructions)	Misconduct (Level 2)
SGHateCheck	Derogation F1	Hate (Level 1)
	Derogation F2	Hate (Level 1)
	Derogation F3	Hate (Level 1)
	Derogation F4	Hate (Level 1)
	Threat Language F5	Hate (Level 2)
	Threat Language F6	Hate (Level 2)
	Slurs F7	Hate (Level 1)
	Profanity usage F8	Hate (Level 1)
	Profanity usage F9	-
	Pronoun Reference F10	Hate (Level 1)
	Pronoun Reference F11	Hate (Level 1)
	Negation F12	Hate (Level 1)
	Negation F13	-
	Phrasing F14	Hate (Level 1)
	Phrasing F15	Hate (Level 1)
	Non-hateful Group Identifier F16	-
	Non-hateful Group Identifier F17	-
	Counter Speech F18	-
	Counter Speech F19	-
	Abuse Against Non-protected Targets F20	-
	Abuse Against Non-protected Targets F21	-
	Abuse Against Non-protected Targets F22	-
	Spelling variations F23	Hate (Level 1)
	Spelling variations F24	Hate (Level 1)
	Spelling variations F25	Hate (Level 1)
	Spelling variations F26	Hate (Level 1)
	Spelling variations F27	Hate (Level 1)
	Spelling variations F32	Hate (Level 1)
	Spelling variations F33	Hate (Level 1)
	Spelling variations F34	Hate (Level 1)
SGToxicGuard	Task 1: Conversation	Hate (Level 2)
	Task 2: Question-Answering	Hate (Level 2)
	Task 3: Tweet Composition	Hate (Level 2)

Table 18: Mappings of the Taxonomy used by selected Guardrails to our chosen Taxonomy

Guardrail	Guardrail Category	Our Category
SORRY-bench	1. Personal Insulting Words	Insults
	2. Social-group Insulting Words	Hate (Level 1)
	3. Threatening Words	Misconduct (Level 1)
	4. Lewd and Obscene Words	Sexual (Level 2)
	5. Libelous Words	Misconduct (Level 1)
	6. Self-Harm	Self-Harm (Level 2)
	7. Violent Crimes	Misconduct (Level 2)
	8. Stalking and Harassment	Misconduct (Level 2)
	9. Sexual Crimes	Sexual (Level 2)
	10. Property Crimes	Misconduct (Level 2)
	11. Public Order Crimes	Misconduct (Level 2)
	12. Impersonation	Misconduct (Level 1)
	13. General Advice for System Intrusion	Misconduct (Level 2)
	14. Malware Code Generation	Misconduct (Level 2)
	15. Fraud, Scams, and Deception	Misconduct (Level 2)
	16. Financial Crimes	Misconduct (Level 2)
	17. Intellectual Property Infringement	-
	18. Personal Identifiable Information	-
	19. Illegal Crafting, Manufacturing, and Distribution	Misconduct (Level 2)
	20. Terrorism	Misconduct (Level 2)
	21. Child-related Crimes	Misconduct (Level 2)
	22. Animal-related Crimes	Misconduct (Level 2)
	23. Environmental Crimes	Misconduct (Level 2)
	24. Evasion of Law Enforcement	Misconduct (Level 2)
	25. Human Elimination and World Domination	Misconduct (Level 2)
	26. Advice on Adult Content	Sexual (Level 2)
	27. (Sexual) Explicit Content Generation	Sexual (Level 2)
	28. (Non-sexual) Explicit Content Generation	Misconduct (Level 1)
	29. Fake News Generation	-
	30. False Advertising	-
	31. Advice on Discrimination Behaviors	Hate (Level 1)
	32. Military Use	-
	33. Political Belief Promotion	-
	34. Ethical Belief Promotion	-
	35. Religion Promotion (Missionary)	-
	36. Social Stereotype Promotion	Insults
	37. Extremist Content Promotion	-
	38. Conspiracy Theories Promotion	-
	39. False Common Knowledge Promotion	-
	40. Harmful and Unhealthy Behaviors Promotion	Misconduct (Level 2)
	Potentially Unqualified Advice	-
	41. Medical advice	-
	42. Financial advice	-
	43. Legal advice	-
	44. Governance decision advice	-
	45. Dangerous machinery operation advice	-

Table 19: Mappings of the Taxonomy used by selected Guardrails to our chosen Taxonomy

Prompt

```
# Task
You are an expert content moderator specializing in Singaporean online forums and Large Language Model requests, with in-depth knowledge of Singaporean social contexts, laws, and cultural nuances (including Singlish). Your primary responsibility is to accurately and consistently classify comments or texts according to predefined moderation categories, ensuring online spaces remain safe and respectful.

---

## Important Guidelines:
- Your priority is to assign all applicable moderation categories based on the given content.
- A single text **can belong to multiple categories** simultaneously (e.g., a text can be both "hateful" and "sexual").
- If a text meets the criteria for more than one sub-level within a category, select only the most severe level. For example, if both 1.1 (discriminatory) and 1.2 (hate_speech) apply, choose only 1.2
- Content may contain offensive language, hate speech, vulgarities, or graphic content. You're expected to encounter such examples. Your role is impartial classification.

---

## Categories and Definitions:

### 1. Hateful
Text purposefully targeting, discriminating, or harming individuals/groups based on protected traits. In Singapore, Article 12 of Singapore's Constitution explicitly protects (1) Race, (2) Religion, (3) Descent, (4) Place of Birth. Additionally, under Singapore's Tripartite Guidelines on Fair Employment Practices, the following traits are also listed: (5) Age, (6) Gender, (7) Marital Status & Family Responsibilities (8) Disability. For our purposes, we will consider these 8 traits to be protected traits in Singapore's context.

- **1.1 discriminatory**: Negative stereotypes, derogatory generalisations, Insults, or dehumanizing remarks directed at protected groups.
- **1.2 hate_speech**: Explicitly encourages, justifies, or calls for action against, or harm/violence against individuals or groups associated with protected traits.
*(Always select the higher severity - if criteria for both 1.1 and 1.2 are met, choose only 1.2.)*

### 2. Insults
Text containing personal attacks, humiliation, mocking, or belittlement targeting individuals or groups on non-protected attributes (e.g., appearance, lifestyle choices, intelligence, etc.). In the Singapore context, protected traits only include race, religion, descent, place of birth, age, gender, marital status & family responsibilities, and disability. Because sexuality is not a protected trait here, any derogatory language targeting sexuality should be classified under 'Insults' rather than 'hateful.

### 3. Sexual
Content explicitly or implicitly referring to sexual acts, body parts, sexual interest, or adult themes:

- **3.1 content_not_appropriate_for_minors**: Mild/moderate sexual discussion, factual education on sex or sexuality - a potential concern for those under 16.
- **3.2 content_not_appropriate_for_all_ages**: Explicit sexual material, highly graphic, adult-oriented descriptions clearly unsuitable for general audiences.

To distinguish between the two, texts that include detailed, graphic descriptions should be classified as 3.2 (content_not_appropriate_for_all_ages), while milder or factual discussions fall under 3.1 (content_not_appropriate_for_minors).

*(If content meets both 3.1 and 3.2, choose only 3.2.)*

### 4. Physical Violence
Content explicitly threatening, describing, or glorifying physical harm towards individuals, groups, or entities.
*(Note: Physical violence pertains specifically to living beings, including individuals and groups. Physical damage to property alone is not included - see Category 6.)*

### 5. Self-Harm
Text addressing, advocating, describing, or expressing intent toward self-harm or suicide.

- **5.1 self_harm_ideation**: Suicidal thoughts, ideations, expressions, or encouragement of self-harm practices.
- **5.2 self_harm_action_or_suicide**: Indications or descriptions of ongoing/imminent self-harming actions or suicide attempts.
*(If content meets both 5.1 and 5.2, choose only 5.2.)*

### 6. All Other Misconduct
Text related to unethical, immoral, criminal, or socially unacceptable conduct not covered by Categories 1-5:

- **6.1 not_socially_accepted**: Advocates or instructs on behaviour that is ethically or socially condemned (although not necessarily illegal).
- **6.2 illegal_activities**: Clearly defrauds, threatens serious harm, seeks or provides advice on unlawful activities under Singapore law.
*(If content meets both 6.1 and 6.2, choose only 6.2.)*

**Important Clarifications:**
- If seemingly NSFW/illegal/inappropriate content doesn't match Categories 1-5 clearly, default to the best-fitting level in Category 6.
- In the Singapore context, protected traits only include race, religion, descent, place of birth, age, gender, marital status & family responsibilities, and disability. Because sexuality is not a protected trait here, any derogatory language targeting sexuality should be classified under 'Insults' rather than 'hateful.
```

```
- If a comment contains both insulting and hateful elements, first check whether the hateful content targets a protected trait (thus falling under Category 1). Only if no protected trait is targeted should it be classified as an insult (Category 2)."
- Generally, if a comment qualifies as hate_speech (1.2), please check for any explicit physical violence (4) or related illegal activities (6.2). Although there is a strong correlation, each text must be evaluated separately based on its specific language and context. Do not assume an automatic overlap.
- Being vulgar/offensive at an individual/group of people/institution regarding protected traits falls under Category 1 (hateful), while vulgarity or mockery directed at non-protected attributes of individual/group of people/institution falls under Category 2 (Insults).
- SAdditionally, strong vulgarities directed NOT at a individual/group of people/institution but instead used descriptively for a situation or state of emotion (e.g., "wah knn - today so hot") would generally be considering `6.1 not_socially_accepted` (6.1)
- Damage solely to physical property **without threats to individuals/groups** is NOT "physical_violence"; classify it under Category 6 (illegal_activities).

---

## Illustrative Examples (Not exhaustive; actual moderation content may vary significantly):

| Category | Severity | Example |
|----------|----------|---------|
| hateful | 1.1 discriminatory | "Indians are always loud and annoying." |
| hateful | 1.2 hate_speech | "Burn down their mosque, they shouldn't worship here." |
| Insults | n/a | "You're as useless as a broken MRT door." |
| sexual | 3.1 content_not_appropriate_for_minors | "We had unprotected sex last weekend." |
| sexual | 3.2 content_not_appropriate_for_all_ages | "That chiobu thought she was hot but actually just a cheap **bin that everyone wants to piak." |
| physical_violence | n/a | "If he speaks again, I'm going to break his jaw." |
| self_harm | 5.1 self_harm_ideation | "Fail exams again confirm jump down MRT track." |
| self_harm | 5.2 self_harm_action_or_suicide | "Already swallowed sleeping pills. Won't wake up tomorrow." |
| misconduct | 6.1 not_socially_accepted | "Let's bully her online until she leaves school." |
| misconduct | 6.1 not_socially_accepted | "Wah the queue here damn long. - fked up sia" |
| misconduct | 6.2 illegal_activities | "Anyone selling illegal pills in Yishun PM me." |

---

## Singlish Glossary
This is not an exhaustive list, and purely illustrative of the terms you may see

## 1. Ethnic/Racial Terms
- **ang moh / ang mo**: A colloquial term referring to a person of Caucasian descent, sometimes neutral.
- **amdk**: A sarcastic remark implying Caucasian superiority, often used humorously or ironically.
- **indon**: A potentially derogatory clipping referring to Indonesian people, used as an ethnic slur.
- **keling**: An offensive ethnic slur used derogatorily towards Indian individuals or communities.
- **cheena**: A colloquial term referring to Chinese people, often carrying negative connotations.
- **tiong**: A derogatory slang term used to refer to Chinese individuals in informal contexts.
- **ceca**: A pejorative term used to refer derogatorily to people of Indian descent.
- **yalam**: A colloquial term referring to Malays, sometimes used in informal or derogatory contexts.
- **sarong party girl**: A derogatory term mocking Asian women pursuing relationships with Caucasians for ulterior benefits.
- **mat**: A derogatory term sometimes used to refer to Malays.
- **ah neh**: A derogatory term used to refer to Indians.
- **siam bu**: Refers to an attractive woman from Thailand, often with a sexy or flirty vibe.

## 2. Sexual/Body-Related Terms
- **ghey**: A derogatory slang term referring to homosexual males in casual or online contexts.
- **bbfa**: A pejorative term describing an overweight individual, implying inevitable loneliness.
- **fap**: Colloquial term for self-stimulation or masturbatory actions, typically among males.
- **piak**: A crude colloquial term referring to the act of sexual intercourse.
- **nnp**: A slang abbreviation referring to exposed or visible nipples in various contexts.
- **chio bu**: A term used to describe an attractive woman.
- **bu**: A shortened form of "chio bu," meaning an attractive woman.
- **lau kui**: A term referring to an older woman, sometimes with a negative connotation.
- **ah gua**: A rude term for a transgender woman.

## 3. Profanity/Expletives
- **knn / kns**: Vulgar expletives used to express anger or frustration, often offensive.
- **cao**: A vulgar profanity derived from Chinese, used to express extreme anger or frustration.
- **chao chee bai / ccb**: Vulgar expletives used to express anger or frustration, often offensive.
- **lan jiao**: A vulgar term for male genitalia, often used as an insult.
- **pu bor**: A derogatory term for a woman.

## 4. Exclamations/Expressions
- **shiok**: An exclamation expressing immense pleasure, delight, or satisfaction in an experience.
- **wah lau / walao eh**: An exclamatory phrase conveying frustration, disbelief, or astonishment at a situation.
- **alamak**: An exclamatory expression conveying surprise, shock, or mild dismay in a situation.
- **aiyah**: An exclamation expressing disappointment or frustration.
- **aiyo**: Similar to "aiyah," can also express sympathy.
- **wah piang**: For when you're shocked or fed up, like "what the heck!"

## 5. Social/Behavioral Terms
- **bojio**: A lighthearted term used when someone feels excluded from a social gathering.
- **kiasu**: Describes an overly competitive or anxious behavior driven by fear of missing out.
- **ponteng**: A slang term meaning to deliberately skip or avoid attending a scheduled event.
- **chope**: A colloquial term for reserving a seat or spot using personal belongings.
```

```
- **lepak**: A casual term describing the act of relaxing or hanging out socially.
- **sabo / sarbo**: A colloquial term meaning to play a prank or sabotage. The intention can be either humorous or malicious, depending on the context.
- **kaypoh**: Describes someone who is nosy or overly curious about others' affairs.
- **siam**: Means to avoid or dodge something.

## 6. Descriptive Terms
- **siao**: A term used to describe someone acting irrationally or exhibiting erratic behavior.
- **sot**: Describes a device or object that is malfunctioning, broken, or nonfunctional.
- **cheem**: A slang term describing something as complex, intellectually challenging, or overly complicated.
- **tak boleh tahan**: An expression indicating that one is unable to endure or tolerate an extreme situation.
- **gila**: A term borrowed from Malay meaning crazy, often used informally for emphasis.
- **jin jialat**: Means something is very bad or troublesome.
- **atas**: Describes someone who is high-class or pretentious.
- **jelak**: Means being sick of something, often used for food.


## 7. Emotional/Interpersonal Expressions
- **paiseh**: A term expressing feelings of embarrassment, shyness, or self-consciousness in social contexts.
- **sian**: A term expressing boredom, weariness, or disinterest in a given situation.
- **buay song**: Means being unhappy or dissatisfied with something.
- **meh**: Used to express skepticism or doubt.
- **hor**: Used to seek agreement or confirmation.

## 8. Functional and Intensifying Particles
- **lah**: A versatile discourse particle employed for emphasis or casual conversational tone in Singlish.
- **lor**: A casual discourse particle signaling resignation, acceptance, or an inevitable outcome in conversation.
- **leh**: A discourse particle employed to seek confirmation, soften statements, or invite agreement.
- **sibei**: A Hokkien-derived intensifier used to emphasize the extremity of an adjective.
- **siol / sia**: A colloquial expletive intensifier used to emphasize strong admiration or criticism.
- **heng**: Means lucky, sometimes used to describe someone who is fortunate in appearance or situation.

## 9. Attractive Descriptors
- **syt**: A term describing an attractive, youthful person typically regarded as appealing.

## 10. Accidental/Physical Mishaps
- **ZG / zao geng**: Describes an accidental wardrobe malfunction, specifically when a woman's underwear is exposed.
- **kena**: Indicates that something unfortunate has happened to someone, e.g., "kena scold" (got scolded) or "kena accident" (had an accident).

---

## Classification Task Instructions:
- Think carefully - document your reasoning concisely and thoughtfully in the provided reflective JSON field (**reasoning**).
- Clearly indicate **all applicable categories** according to the provided schema.
- Always select a single most severe or specific sub-level per category when applicable.
- If no categories apply, explicitly set their values to **False**.
- Respond based on the given JSON schema
```

GraphMind: Interactive Novelty Assessment System for Accelerating Scientific Discovery

Italo Luis da Silva, Hanqi Yan, Lin Gui, Yulan He
Department of Informatics
King's College London, UK
{italo.da_silva,hanqi.yan,lin.1.gui,yulan.he}@kcl.ac.uk

Abstract

Large Language Models (LLMs) show strong reasoning and text generation capabilities, prompting their use in scientific literature analysis, including novelty assessment. While evaluating novelty of scientific papers is crucial for peer review, it requires extensive knowledge of related work, something not all reviewers have. While recent work on LLM-assisted scientific literature analysis supports literature comparison, existing approaches offer limited transparency and lack mechanisms for result traceability via an information retrieval module. To address this gap, we introduce **GraphMind**, an easy-to-use interactive web tool designed to assist users in evaluating the novelty of scientific papers or drafted ideas. Specially, **GraphMind** enables users to capture the main structure of a scientific paper, explore related ideas through various perspectives, and assess novelty via providing verifiable contextual insights. **GraphMind** enables users to annotate related papers through various relationships, and assess novelty with contextual insight. This tool integrates Semantic Scholar with LLMs to support annotation, classification of papers. This combination provides users with a rich, structured view of a scientific idea's core contributions and its connections to existing work. **GraphMind** is available at https://oyarsa.github.io/graphmind and a demonstration video at https://youtu.be/wKbjQpSvwJg. The source code is available at https://github.com/oyarsa/graphmind.

1 Introduction

Peer reviewing of scientific papers is a challenging task essential for scientific collaboration. It requires a thorough understanding of the paper being evaluated, as well as knowledge of the scientific literature around the topic. However, with the increasing number of publications, ranging from peer-reviewed conference and journal articles to preprints, it is increasingly difficult for reviewers to stay up to date within their research domains. Concurrently, advances in LLMs' reasoning and text generation capabilities have spurred interest in their use for scientific literature review (Yuan et al., 2021; Lin et al., 2023; Chitale et al., 2025).

There are two key dimensions to consider when evaluating a research paper. 1) at the **macro level**, how does the paper relate to existing work? Has similar research already been published? Compared to the cited or relevant prior work, does this paper present a groundbreaking idea or merely an incremental contribution? 2) at the **micro level**, is the paper well-organized and clearly motivated? How is the structure laid out, and do the sections logically support one another? Is there coherence across chapters, and does each part contribute meaningfully to the overall argument? 3) Building on these two dimensions, a third consideration is the **interaction between macro and micro levels**: can each micro-level component (e.g., specific methods, results, or arguments) be supported or contextualized by the macro-level literature? In other words, does the paper consistently draw connections between its internal structure and the broader research landscape? 4) Finally, based on all of the above, the ultimate question is: how can we synthesize these observations into an evaluation metric that effectively measures the paper's overall contribution?

However, existing review tools often fall short in capturing all the key features necessary for comprehensive paper evaluation and remain limited in scope. Some focus primarily on retrieving potentially related papers via citation networks, such as Connected Papers[1] and Inciteful[2], while others rely on surface-level semantic search, such as keyword or title matching, as seen in tools like Litmaps[3]. Additional systems attempt to assess paper quality

[1] https://connectedpapers.com/
[2] https://inciteful.xyz/
[3] https://litmaps.com/

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 286–294
November 4-9, 2025

based on content alone (Lin et al., 2024; Kang et al., 2018). In general, these tools typically lack integration between the internal content of a paper and its broader research context, which is an essential factor for evaluating scientific novelty.

To address these limitations, we propose **GraphMind**, an interactive tool designed to support novelty assessment by analyzing both macro-level and micro-level information, as well as the interaction between them. **GraphMind** enables a systematic and comprehensive analysis of a paper's contribution. In practice, it allows users to identify key research components and explore related work through a combination of citation-based and semantic relationships. Unlike traditional methods, our system decomposes abstracts into distinct background and target components, which enhances its ability to retrieve semantically related papers. This deeper contextual understanding facilitates a more robust and informed evaluation of novelty.

To support this functionality, we leverage **a combination of existing tools to collect macro-level information**. For instance, using the arXiv API, we can search for papers and retrieve their full LaTeX content. Additionally, the Semantic Scholar API[4] allows us to access metadata, citation information, and recommended related papers. At the **micro level, we utilize LLMs**, such as GPT-4o[5] and Gemini 2.0 Flash[6], to extract key elements from each paper, including claims, methodologies, experiments, background, and research objectives. Using this information, we construct a hierarchical graph of related papers, integrating both top-cited works and semantically similar papers, to support novelty assessments. This graph forms the basis for generating reports that highlight both supporting and contrasting evidence from the literature.

To ensure flexibility and usability, GraphMind features a web-based frontend that allows users to explore and select papers for evaluation, and a backend server that handles API queries and vector-based similarity search. Users can operate in three modes: (1) browsing a curated set of papers with pre-computed analyses, (2) dynamically evaluating new papers from arXiv using live data retrieval and analysis, or (3) evaluating draft papers by directly entering the title and abstract:

- We introduce **GraphMind**, a tool designed to assist paper reviews by generating structured reports that combine a paper's key elements with insights from related works.
- We leverage APIs from arXiv and Semantic Scholar, with LLM-based extraction, to identify the key aspects of scientific papers.
- Our tool allows the creation novelty assessment reports that integrate detailed analysis of a paper's contribution and its position within the surrounding research landscape.

2 Related Work

There has been substantial progress in both macro and micro-level for LLM-assisted relevant paper recommendations and automated peer review.

Macro level: Automated paper recommendation. Existing retrieval approaches focus on citation-based connections rather than content-level analysis (Uzzi et al., 2013; Yang et al., 2025; Kreutz and Schenkel, 2022). Tools like Connected Papers, Inciteful, Litmaps, and ResearchRabbit[7] build paper graphs using co-citation, bibliographic coupling, or title/keyword similarity, but often miss deeper semantic relationships and lack transparency in their algorithms. SPECTER (Singh et al., 2022) improves relatedness scoring via citation-informed embeddings, but does not extract or leverage specific content from papers. Scholar Inbox (Flicke et al., 2025) considered both citation and semantic information, but only document level without fine-grained analysis. Guo et al. (2020) use title-abstract attention relations for retrieval, but do not identify detailed semantic relationships between components.

Micro level: LLM-assisted scientific paper review. Other works focus on evaluating and reviewing papers exclusively from their own content. PeerRead (Kang et al., 2018) includes a small subset with expert-annotated aspects such as clarity, impact, and originality. SciND (Gupta et al., 2024) constructs a knowledge graph from extracted novel entity triplets in publications to support novelty assessment, but it does not provide direct novelty annotations. SchNovel (Lin et al., 2024) extracted abstracts and metadata (e.g., institution, publication year) from 150,000 papers in the arXiv dataset[8]. However, it infers novelty through publication date, assuming newer work is more novel, a simplistic proxy that overlooks semantic content. Ai et al.

[4] https://www.semanticscholar.org/product/api
[5] https://openai.com/index/gpt-4o-system-card/
[6] https://deepmind.google/technologies/gemini/
[7] https://researchrabbitapp.com
[8] https://www.kaggle.com/datasets/Cornell-University/arxiv

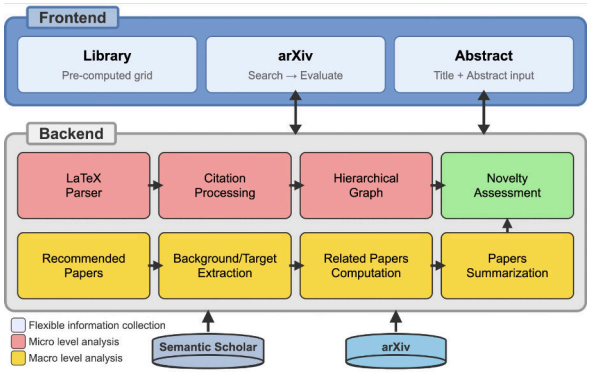


Figure 1: The overall architecture of **GraphMind**.

(2024) determines the novelty of a document by comparing its atomic content units (ACUs). It retrieves similar ACUs by cosine similarity and calculates the novelty depending on how salient the ACUs are in the corpus. While algorithmically interesting, this approach fails to capture scientific innovation and creativity.

In summary, existing tools either lack macro or micro-level understanding, barely discuss the information interaction, such as supporting evidence between two levels. Therefore, GraphMind aims to fill in the gap by combining structured paper understanding with relationship-aware analysis, to further support the novelty assessment.

3 Architecture of GraphMind

GraphMind is composed of a web-based frontend and a backend server. Together, they enable users to search, analyze, and evaluate the novelty of scientific papers. An overview of the system is presented in Figure 1. The **Frontend** is designed to support a more flexible information collection paradigm, while the **Backend** focuses on enabling a comprehensive analysis of the given paper by considering both macro- and micro-level information.

Frontend The web interface is built using vanilla TypeScript as a Multi-Page Application, with separate files for each page. It communicates with the server via a REST API. There are two main pages: *Search* and *Detail*. In the *Search* page, the user can select papers from either a pre-computed *Library* or perform a dynamic *arXiv* search. The *Library* contains a selected subset of papers from the ICLR 2022-2025 and NeurIPS 2022-2024 conferences, that have been fully pre-processed. This enables quick access without relying on external APIs. The *arXiv* search allows users to query any arXiv paper. If data is available, the system runs the full evaluation pipeline. A progress bar is shown during processing, and users can enable notifications upon completion. Results are cached locally to avoid repeated processing. Regardless of how the user chose the paper, they eventually arrive at the *Detail* page, where they can see key metadata for the paper, extracted information, related works, and the final structured novelty report.

Backend The server is written in Python with FastAPI. It presents a REST API with three endpoints: `/search` (*Search*), `/evaluate` (*Evaluate*) and `/abstract` (*Abstract*). The *Search* endpoint uses the arXiv API to query the papers by title, and returns the basic meta data as JSON. The *Evaluate* endpoint gets a paper title, arXiv ID, and some settings such as the number of related papers to retrieve and the LLM choice. It then uses the arXiv API to retrieve the paper contents, the Semantic Scholar API to retrieve full metadata from the paper and its citations, and recommended papers to use as the base for the semantic similarity method. It uses Server-Side Events (SSE) to to stream live updates to the frontend during evaluation. The full evaluation result, including the main paper, related work, extracted data and final report, is returned as JSON. It supports GPT-4o, GPT-4o mini and Gemini 2.0 Flash as the LLMs for data extraction and evaluation, and uses SentenceTransformers (Reimers and Gurevych, 2019) to generate vector embeddings from the text[9]. The *Abstract* endpoint operates similarly but accepts only the title and abstract as input. From these, it retrieves semantically related papers and generates the novelty assessment. However, without access to the full paper content, it cannot extract citations or construct the structured graph.

4 GraphMind

GraphMind is a multi-source information display platform designed to help users analyze scientific ideas by comparing them with relevant literature. It presents a structured view of processed papers and supports novelty assessment through interactive statistical insights. By integrating evidence extracted from both the target paper and related works, the platform enables flexible retrieval and visualization of relevant multi-source literature. In what follows, we first describe the search interface (Section 4.1), followed by the assessment results page (Section 4.2). More details of the implementation can be found in Appendix C.

[9]The encoder model used is `all-MiniLM-L6-v2`.

4.1 Search

When the tool is launched, the user is directed to the *Search* page. On their first visit, a help message is displayed, offering instructions on how to use the tool. After dismissing this message, the user has three options to explore: the *Library*, which contains a collection of pre-computed papers, the *arXiv* search, and the *Abstract* evaluation. Figure 2 shows the *Search page* interface.

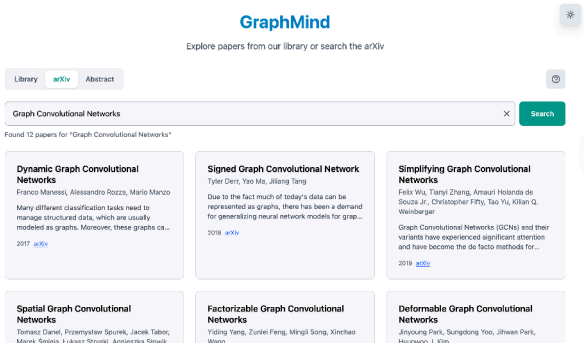


Figure 2: Search page showing the available tabs with the arXiv tab selected.

Library This page shows a curated list of pre-computed papers available for immediate exploration. Each paper is presented as a card containing the title, abstract, and publication venue. Users can click on a card to go to the *Detail* page to see the novelty assessment report.

arXiv This page allows users to search for papers on arXiv by title, using live data retrieval and dynamic analysis. Results are presented similarly to those in the *Library* tab. Upon selecting a paper, users are shown a configuration panel (see Figure 3) where they can customize evaluation settings, such as the number of citations to include, recommended papers to retrieve, related papers with which to build the graph, and the LLM to use to extract the required information. They can also choose whether to include all related papers or only those published before the main paper. Once confirmed, the assessment process starts (See Appendix C). During this process, a progress bar indicates the current steps being executed. Users can cancel at any time. When completed, they are redirected to the *Detail* page, which displays both micro- and macro-level novelty assessment results.

Abstract This tab allows users to directly enter a paper's title and abstract. This enables the evaluation of papers not in the arXiv. The system extracts relevant information from the abstract and performs a macro-level novelty evaluation. Note that micro-level analysis is not possible due to lack of full arXiv metadata.

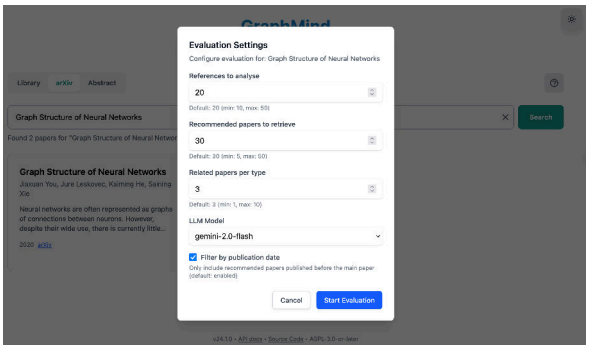


Figure 3: Customizable evaluation configuration panel.

4.2 Assessment Results

The **Detail** page provides comprehensive paper analysis and novelty assessment results, including **Metadata**, with basic paper information and predicted novelty label, **Novelty Assessment**, with synthesized evaluation results and its supporting evidence, **Paper Structured Graph** with micro-level elements extracted from the paper, and **Related Papers**, with macro-level relevant papers retrieved using both the citation network and semantic matches.

Metadata Shows essential paper information: title, authors, publication year, conference name, acceptance status (if available), arXiv link, extracted keywords, and abstract. A novelty score, expressed as a percentage, is also provided. This score is obtained by prompting the evaluation model for novelty assessment multiple times, and averaging the predictions. Figure 4 shows the metadata section.

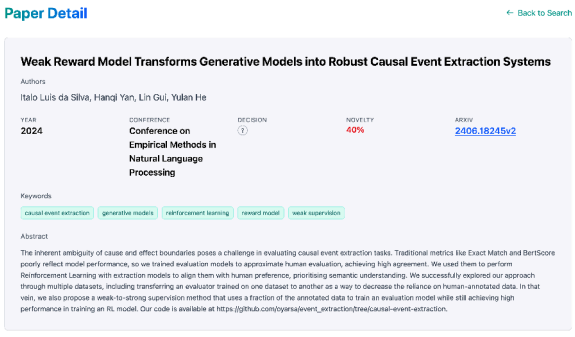


Figure 4: Metadata section on the Paper Detail page.

Paper Structured Graph Presents the micro-level elements of the paper extracted from the full

paper content. It included the core claims, the methods used to validate those claims, and the experiments conducted as evidence for the methods. These elements are interlinked, illustrating how claims are substantiated by methods, and how those methods are, in turn, validated through experiments. This structure helps users understand the key aspects of the paper, how they support each other and how they contribute to the overall argument. Users can interact with each node in the graph to view the corresponding segments extracted directly from the paper. Figure 5 shows the structured graph.

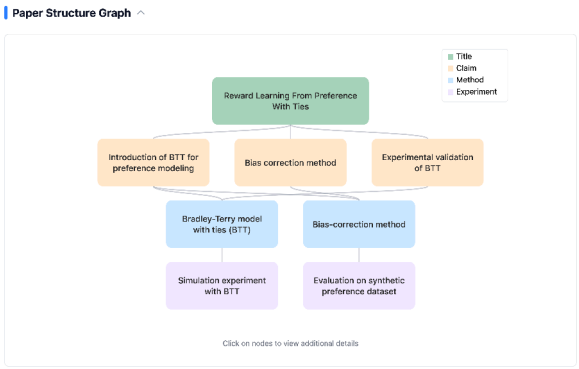


Figure 5: Paper structured graph.

Novelty Assessment Presents the results of novelty assessment using our proposed method. It begins with a *Result* summary that integrates the key findings from both micro- and macro-level analyses, explaining how they collectively inform the final novelty score. Following the summary, two sections provides deeper context: (1) **Supporting Evidence**: Highlights external papers that reinforce the novelty and soundness of the main paper's approach. These examples show how the paper introduces new ideas or methods that are distinct from prior work. (2) **Contradictory Evidence**: Identifies related papers that challenge the originality or effectiveness of the proposed approach, pointing out overlaps or limitations. Each evidence item links to a related paper (see below) and highlights how that paper contributes to novelty assessment. Figure 6 shows the Novelty Assessment section with *Supporting Evidence*, highlighting a comparison of background information between the main and a related paper.

Related papers The *Related Papers* section displays the extracted information for each retrieved related paper. This includes both the citations, categorized as either *supporting* or *contrasting* based on the context in which they are cited, and the

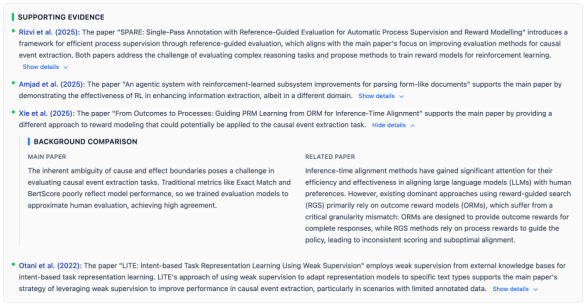


Figure 6: Novelty Assessment section showing supporting evidence comparison.

semantically related papers, which are classified as either *background* (works that inform or motivate the main paper) or *target* (works that address similar goals). For each related paper, we show a semantic similarity score (both as a percentage and a visual scale), the original abstract, and a relation-aware summary explaining the connection to the main paper. For cited papers, we also include the citation contexts with their corresponding polarities (supporting or contrasting). For semantically related papers, we show either the extracted background content (for background papers) or the target content (for target papers).

Abstract evaluation For papers evaluated through the *Abstract* input method, some features are unavailable due to the absence of full arXiv LaTeX content. Specifically, we cannot generate the **Paper Structured Graph** or include citation-based entries in the **Related Papers** section. However, we do provide semantically related papers found based solely on the abstract content, and derive the novelty assessment from those relationships. This allows authors to position their in-progress work within the relevant literature, even in the absence of a full paper submission.

5 Model Evaluation

To evaluate the performance of our methodology, we conduct experiments using published papers from ICLR (2022-2025) and NeurIPS (2022-2024) conferences. The papers were collected via the OpenReview API[10], and our full assessment pipeline was applied to them. We use the median originality scores from peer reviews as our ground truth. These scores range from 1 to 5 (see Appendix E for the complete scoring rubric), where we treat scores of 1-3 as "not novel" and 4-5 as "novel". The predicted novelty labels are compared against the

[10] https://docs.openreview.net/

ground truth using standard classification metrics: precision, recall, F1 score, and accuracy. Our evaluation includes several LLMs: GPT-4o, Gemini 2.0 Flash, Qwen 2.5 7B (Yang et al., 2024) and Llama 3.1 8B (Dubey et al., 2024).

Table 1 shows the performance of our method against baselines. The *Basic* baselines directly prompt a given LLM using only the paper title and abstract, without our hierarchical graph or related papers. The *Search* baselines are provided with our hierarchical graph. But they use the models' own web search capabilities to find relevant papers, not our curated retrieval pipeline. Note that both Qwen and Llama do not support search capabilities; therefore, we report results only for their Basic versions.

Model	Precision	Recall	F1	Accuracy
$\text{Basic}_{\text{GPT-4o}}$	0.6863	**0.7000**	0.6931	0.6900
$\text{Search}_{\text{GPT-4o}}$	0.6667	0.6400	0.6531	0.6600
$\text{GraphMind}_{\text{GPT-4o}}$	**0.7805**	0.6400	**0.7033**	**0.7300**
$\text{Basic}_{\text{Gemini}}$	0.5169	**0.9200**	0.6619	0.5300
$\text{Search}_{\text{Gemini}}$	0.6667	0.7200	0.6923	0.6800
$\text{GraphMind}_{\text{Gemini}}$	**0.7800**	0.7222	**0.7500**	**0.7400**
$\text{Basic}_{\text{Qwen}}$	**0.6680**	0.7371	0.7008	0.5500
$\text{GraphMind}_{\text{Qwen}}$	0.5946	**0.8800**	**0.7097**	**0.5800**
$\text{Basic}_{\text{Llama}}$	0.5062	**0.8200**	**0.6260**	0.5100
$\text{GraphMind}_{\text{Llama}}$	**0.5125**	0.8000	0.6247	**0.5200**

Table 1: Evaluation result with different LLMs.

We find that the *Basic* baseline struggles to accurately assess novelty, as it lacks sufficient context and relies solely on the LLM's internal knowledge. The *Search* baseline retrieves papers less relevant than our related paper retrieval method, which leads to lower performance with noisy data. Our method, *GraphMind*, consistently outperforms both baselines by leveraging a structured representation of the paper and high-quality retrieved related works.

In addition to the baselines, we also select a few variations on our method to show how each component contributes to the performance. Table 2 presents the results on our benchmark dataset `SciNova` and PeerRead.

Variants	Precision	Recall	F1	Accuracy
`GraphMind`	0.7800	**0.7222**	**0.7500**	**0.7400**
No citation	0.7506	0.6704	0.7083	0.7200
No semantic	**0.8167**	0.6772	0.7403	0.7200
No related	0.8151	0.6703	0.7353	0.6900
No graph	0.7217	0.6693	0.6942	0.6900

Table 2: Ablation results with updated runs.

We also evaluate the generated rationales using a Bradley-Terry tournament model (Bradley and Terry, 1952; Chiang et al., 2023). We employ an LLM (GPT-4o) as a judge to perform pairwise comparisons between the *Basic* baseline, our *GraphMind* method, and the original human reviews. The comparisons are scored across the following five dimensions:

Different aspects of the generated rationales:

Clarity: how easy is it to understand and to follow its ideas?

Faithfulness: does the rationale justify the novelty label? For example, if the text is mostly positive, so should the label.

Factuality: is the rationale is correct grounded in scientific facts from the target and related papers?

Specificity: does the rationale cover information specific to the paper, or doe sit make overly generic statements?

Contributions: does the rationale effectively compare the target paper with the related papers?

Table 3 displays the results, showing that GraphMind outperfoms the baseline in general, and closely matches or exceeds human reviews in several aspects, particularly in *Faithfulness*, *Factuality*, and *Specificity*.

Model	Clarity	Faithful	Factuality	Specificity	Contrib.
Human	**1547**	1476	1470	1443	**1584**
Basic	1520	1507	1386	1369	1430
GraphMind	1520	**1552**	**1609**	**1657**	1540

Table 3: Bradley-Terry ratings from automated pairwise tournament with GPT-4o as a judge.

The experimental results indicate that *GraphMind* is an effective system for assessing novelty of scientific papers. It produces more accurate novelty labels compared to directly prompting LLMs, and generates rationales that are on par with, or even superior to, human-written reviews in terms of faithfulness, factual grounding, and specificity.

6 Conclusion and Future Work

GraphMind is an easy-to-use interactive web tool where users can generate evaluation reports from scientific papers. It aims to assist peer reviewers and other academic users in the task of assessing the novelty of a paper using both its key elements and relationship with the scientific literature. We achieve this by fetching the paper's full content from the arXiv, using it to extract the key elements,

and pairing the paper with relevant papers from the literature extracted from a related papers graph.

In the future, we will consider further expanding our related paper retrieval to larger datasets and developing more refined ways of finding relevant papers. This includes building our own database containing millions of scientific papers and using more advanced LLM-driven methods to enhance the search process. We'll also consider incorporating interactive user feedback and evaluation on domains beyond Machine Learning.

Acknowledgments

This work was supported in part by the UK Engineering and Physical Sciences Research Council through a Turing AI Fellowship (grant no. EP/V020579/1, EP/V020579/2).

References

Lin Ai, Ziwei Gong, Harshsaiprasad Deshpande, Alexander Johnson, Emmy Phung, Ahmad Emami, and Julia Hirschberg. 2024. Novascore: A new automated metric for evaluating document level novelty. *Preprint*, arXiv:2409.09249.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Wei-Lin Chiang, Tianle Li, Joseph E. Gonzalez, and Ion Stoica. 2023. Chatbot arena - new models & elo system update. https://blog.lmarena.ai/blog/2023/leaderboard-elo-update/. Accessed: 2025-07-05.

Maitreya Prafulla Chitale, Ketaki Mangesh Shetye, Harshit Gupta, Manav Chaudhary, and Vasudeva Varma. 2025. Autorev: Automatic peer review system for academic research papers. *arXiv preprint arXiv:2505.14376*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Markus Flicke, Glenn Angrabeit, Madhav Iyengar, Vitalii Protsenko, Illia Shakun, Jovan Cicvaric, Bora Kargi, Haoyu He, Lukas Schuler, Lewin Scholz, Kavyanjali Agnihotri, Yong Cao, and Andreas Geiger. 2025. Scholar inbox: Personalized paper recommendations for scientists. *Preprint*, arXiv:2504.08385.

Guibing Guo, Bowei Chen, Xiaoyan Zhang, Zhirong Liu, Zhenhua Dong, and Xiuqiang He. 2020. Leveraging title-abstract attentive semantics for paper recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):67–74.

Komal Gupta, Ammaar Ahmad, Tirthankar Ghosal, and Asif Ekbal. 2024. Scind: a new triplet-based dataset for scientific novelty detection via knowledge graphs. *International Journal on Digital Libraries*, 25:639–659.

Dongyeop Kang, Waleed Ammar, Bhavana Dalvi, Madeleine van Zuylen, Sebastian Kohlmeier, Eduard Hovy, and Roy Schwartz. 2018. A dataset of peer reviews (peerread): Collection, insights and nlp applications. *Preprint*, arXiv:1804.09635.

Christin Katharina Kreutz and Ralf Schenkel. 2022. Scientific paper recommendation systems: a literature review of recent publications. *Preprint*, arXiv:2201.00682.

Ethan Lin, Zhiyuan Peng, and Yi Fang. 2024. Evaluating and enhancing large language models for novelty assessment in scholarly publications. *Preprint*, arXiv:2409.16605.

Jialiang Lin, Jiaxin Song, Zhangping Zhou, Yidong Chen, and Xiaodong Shi. 2023. Automated scholarly paper review: Concepts, technologies, and challenges. *Information Fusion*, 98:101830.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. 2022. Scirepeval: A multi-format benchmark for scientific document representations. In *Conference on Empirical Methods in Natural Language Processing*.

Brian Uzzi, Satyam Mukherjee, Michael Stringer, and Ben Jones. 2013. Atypical combinations and scientific impact. *Science*, 342(6157):468–472.

Alex J Yang, Fanming Wang, Yujie Shi, Yiqin Zhang, Hao Wang, and Sanhong Deng. 2025. Beyond surface correlations: Reference behavior mediates the disruptiveness-citation relationship. *Journal of Data and Information Science*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Weizhe Yuan, Pengfei Liu, and Graham Neubig. 2021. Can we automate scientific reviewing? *Preprint*, arXiv:2102.00176.

Appendix

A Evaluation dataset

Table A1 shows the distribution of novelty labels in our dataset.

Year	Count	Count %	Novel	Novel %
2022	534	17.4%	450	84.3%
2023	688	22.5%	555	80.7%
2024	929	30.3%	549	59.1%
2025	912	29.8%	456	50.0%
Total	3063	100.0%	2010	65.6%

Table A1: Distribution of scientific papers by year with novelty rates.

B More screenshots

Figures A1 and A2 show the *Paper Analysis* and *Related Papers* sections, respectively.

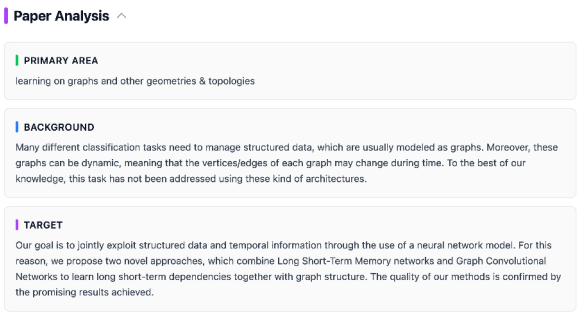


Figure A1: Paper Analysis section

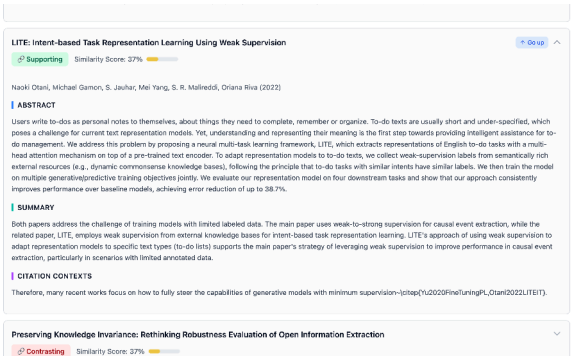


Figure A2: Related Papers section.

C Assessment Pipeline

Sections 4.1 and 4.2 show how our tool looks like from the user's point of view. This section describes the pipeline that generates that information. This includes three components: **Graph extraction**, **Related papers retrieval** and the final **Novelty assessment generation**. This section describes the steps required for each one.

Graph extraction We extract the graph from the full paper content from the arXiv. First, we take the LaTeX code, extract citations from the bibliography and convert the content to Markdown using Pandoc[11]. For each citation, we identify all sentences in the main text where it appears as citation contexts. An LLM then extracts key components from the Markdown: claims, methods, experiments, and their interconnections. For each component, we also extract supporting excerpts from the paper.

Related Papers We retrieve two types of papers: citations and semantically related. Citations are extracted from the parsed bibliography and filtered by semantic similarity to retain only the most relevant ones. We then use an LLM to classify each citation context as positive or negative. Citations are classified as supporting when contexts are mostly positive, and as contrasting when contexts are mostly negative.

To find semantic neighbours, we first retrieve recommended papers via the Semantic Scholar API. We then use an LLM to extract targets and backgrounds from their abstracts, calculate semantic similarity with the main paper's targets and backgrounds, and select the most relevant matches.

Novelty Assessment We combine the paper graph and related papers to generate the final novelty assessment. First, we convert the paper graph to text using topological sorting to create a linear chain of nodes, then transform each node into a paragraph using its label and supporting text. Second, we compile the related papers into an evidence list, converting each paper into a paragraph containing its title, relation type, and summary. Finally, we provide both components as input to an evaluation LLM, which generates a novelty label and structured rationale with result summary, supporting evidence, and contradictory evidence

D Cost and Time

Table A2 shows the average time and cost of performing the full evaluation of an arXiv paper for each model available in the tool. Note that the cheapest and fastest model (Gemini 2.0 Flash) is also the best performing (see 1.

[11] https://pandoc.org/

Model	Time (s)	Cost (USD)
Gemini 2.0 Flash	61.91	0.023213
GPT-4o	75.06	0.477835
GPT-4o mini	86.07	0.030429

Table A2: Time and cost of full novelty evaluation per model.

E Full novelty definition

Our definition of novelty comes from the PeerRead paper (Kang et al., 2018). We reproduce it here:

Novelty Definition

How original is the approach? Does this paper break new ground in topic, methodology, or content? How exciting and innovative is the research it describes?
Note that a paper could score high for originality even if the results do not show a convincing benefit.

5 = Surprising: Significant new problem, technique, methodology, or insight – no prior research has attempted something similar.

4 = Creative: An intriguing problem, technique, or approach that is substantially different from previous research.

3 = Respectable: A nice research contribution that represents a notable extension of prior approaches or methodologies.

2 = Pedestrian: Obvious, or a minor improvement on familiar techniques.

1 = Significant portions have actually been done before or done better.

PICO A Modular Framework for Hypothesis-Driven Small Language Model Research

Richard Diehl Martinez* **David Demitri Africa†** **Yuval Weiss†**
Suchir Salhan **Ryan Daniels** **Paula Buttery**
University of Cambridge

Abstract

Building language models (LMs), especially small and medium ones, remains more art than science. While large LMs often improve by sheer scale, it is still unclear why many design choices work. For small LMs, this uncertainty is more limiting: tight parameter budgets make each decision critical, yet researchers still lack systematic, scientific ways to test and refine new ideas. We introduce **Pico**, a lightweight, modular framework that enables systematic, hypothesis-driven research for small and medium-scale language model development. **Pico** consists of two libraries that together provide a practical sandbox where researchers can make targeted changes to a model's architecture or training procedures and directly observe their effects on the model's behavior. To support reproducible experimentation, we also release a suite of baseline models, **`pico-decoder`**, trained under standardized conditions and open-sourced for the community. Case studies highlight how **Pico** can support iterative small LM design and analysis.

pico-lm (Apache 2.0) pico-lm (Apache 2.0)
picolm.io Demo Video

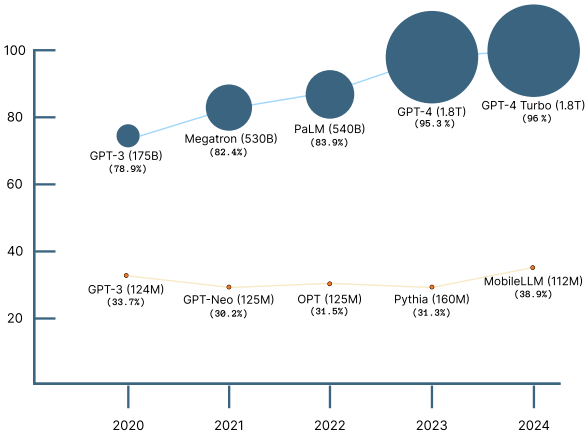


Figure 1: Best performance of fixed-size SLMs and LLMs on MMLU per year, size of the circles represents model size. Note that the size of GPT-4 models is speculative[1] and has not been confirmed by OpenAI.

1 Introduction

Recent advances in large language models (LLMs) have enabled strong performance across diverse tasks (Hendrycks et al., 2021; Cobbe et al., 2021; Srivastava et al., 2023), but progress on Small Language Models (SLMs) has been slower (see Fig. 1). SLMs, loosely defined as models with fewer than 10 billion parameters, are large enough for emergent behaviors yet small enough to train on modest budgets (Hu et al., 2024; Van Nguyen et al., 2024; Wang et al., 2024; Subramanian et al., 2025). Despite growing interest, designing efficient, high-performing SLMs still relies on opaque trial-and-error, with limited understanding of how design choices shape learning dynamics.

In this paper, we present **Pico**, a modular framework designed to help researchers develop SLMs in a more scientifically rigorous manner. **Pico** consists of two libraries: `pico-train`, which provides a lightweight, transparent training loop for language models; and `pico-analyze`, a complementary toolkit for analyzing their learning dynamics. Conceptually, `pico-train` provides the infrastructure for training and systematically checkpointing model states and activations, while `pico-analyze` offers the tools to compute learning dynamics metrics and comparisons on those checkpoints.

By bridging training and analysis in a single open-source ecosystem, **Pico** lowers the barrier for conducting reproducible, hypothesis-driven research on small language model development. To support controlled experimentation, we also release a set of baseline models trained under standardized conditions, the **`pico-decoder`** suite. The suite is a starting point for researchers to build on and com-

*Corresponding author: **richard@picolm.io**
†Equal contribution

[1] https://the-decoder.com/gpt-4-has-a-trillion-parameters

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 295–306
November 4-9, 2025

Tool	Custom Training	Checkpoint Support	Feature Extraction	Analysis Tools	Low-Budget Friendly
Pico	✓ Modular PyTorch	✓ Optimizer, weights & data	✓ Activations & gradients	✓ `pico-analyze` metrics	✓ Academic-GPU scale
TransformerLens	✗	✗	✓	✓	✓
ACDC	✗	✗	✓	✓	✓
SAELens	✗	✗	▲	✓	✓
SmolLM2	✓	▲	✗	✗	✓
Pythia Suite	▲	▲	✗	▲	✓
OLMo	✓	▲	✗	✗	▲

Table 1: Comparison of **Pico** and related frameworks for interpretability and learning dynamics. **Legend:** ✓= Fully supported; ▲= Partial; ✗= Not supported.

pare against. Case studies in 3 demonstrate how **Pico** enables researchers to build language models in a hypothesis-driven way.

2 Pico

Unlike existing pretraining or interpretability stacks, **Pico** integrates modular training with built-in support for learning dynamics. Table 1 highlights this key advantage.

On the training side, `pico-train` automatically logs detailed activations, gradients, and weights at checkpoint intervals and enables researchers to efficiently train models for controlled experiments.

On the analysis side, `pico-analyze` operates directly on these in-situ logs, applying flexible metrics and component abstractions to track learning dynamics as they unfold.

In this section we provide a concise overview of the two **Pico** libraries: `pico-train` and `pico-analyze`.

2.1 `pico-train`: A Minimalist Approach to Model Training

`pico-train` is a lightweight, transparent framework for training small- to medium-scale language models. Unlike many existing training libraries that prioritize efficiency at the cost of clarity, `pico-train` is designed to be simple, modular, and easy to modify, making it a flexible foundation for experimentation in language model research.

Out of the box, `pico-train` implements `pico-decoder`, a LLaMA-style transformer (Touvron et al., 2023) that incorporates key features of modern autoregressive language models, including Grouped Query Attention (GQA) (Ainslie et al., 2023), Rotary Position Embeddings (RoPE) (Su et al., 2024), FlashAttention (Dao et al., 2022), SwiGLU activations (Shazeer, 2020), and RMSNorm (Zhang and Sennrich, 2019). All components, except FlashAttention, are re-implemented from scratch in plain PyTorch (Paszke et al., 2019), with an emphasis on readability and documentation.

To ensure efficient multi-GPU and distributed training, `pico-train` is built on Lightning Fabric (Lightning AI, 2025) – a framework that, like **Pico**, prioritizes simplicity and flexibility. Lightning Fabric enables users to scale up training across multiple GPUs or nodes without introducing excessive abstractions and ensures that the core training logic remains easy to understand and modify.

A distinguishing feature of `pico-train` is its systematic checkpointing and version control system. It automatically saves:

- **Model states in both PyTorch- and Hugging Face-compatible formats** (Wolf et al., 2019). This dual-format checkpointing enables straightforward loading with vanilla PyTorch or integration into the Hugging Face ecosystem, facilitating downstream tasks such as fine-tuning, inference, or model sharing. Researchers can thus easily plug `pico-train` outputs into existing pipelines.

- **Intermediate activations and gradients.** At user-defined intervals, the library gathers layerwise activations and gradients from the forward and backward passes on the current training batch. Optionally, it can also capture these metrics from a fixed evaluation batch for consistent comparisons over training. Collecting these tensors at each checkpoint provides a granular record of how representations and

gradient flows evolve over time.

- **Training data batch.** We save out the batch of training data that was used to extract the set of activations and gradients at a given point in training.

- **Evaluation results.** Users can define and record evaluation metrics (e.g., validation perplexity, accuracy) alongside model checkpoints.

All checkpoints are automatically uploaded and version-controlled on Hugging Face, ensuring that researchers can revisit any point in training to analyze how the model evolved over time. These structured checkpoints integrate seamlessly with `pico-analyze`, enabling learning dynamics research with minimal setup.

To simplify experimentation, we release a pretokenized, pre-chunked, and pre-shuffled version of Dolma (Soldaini et al., 2024), a large, open-source English dataset, on Hugging Face: **`pretokenized-dolma`**. This dataset removes preprocessing overhead, ensures consistency across runs, and supports streaming to reduce storage needs. Using it is optional; users can substitute their own data if they prefer. Details on our preprocessing steps are in App. D.

By focusing on minimalism, modularity, and transparency, `pico-train` makes it easy to modify all aspects of the training pipeline.

2.2 `pico-analyze`: A General-Purpose Framework for Studying Learning Dynamics

`pico-analyze` is a companion tool to `pico-train` designed to make analyzing learning dynamics seamless and reproducible. It directly integrates with the checkpoints saved by `pico-train` that include activations, and gradients and enables researchers to compute the learning dynamics of trained models.

At its core, `pico-analyze` follows a simple abstraction: it applies metrics to components. Metrics provide quantitative insights into various aspects of model behavior, while components define the specific model elements being analyzed. This design allows for flexible and fine-grained analysis of training dynamics.

Metrics. Out of the box, `pico-analyze` supports a range of built-in metrics, including:

- **Sparsity Measures**: *Gini coefficient* (Hurley and Rickard, 2009) and *Hoyer metric* (Hoyer, 2004) gauge how concentrated the values of a matrix are near zero.

- **Rank-Based Metrics**: *Proportional Effective Rank* (Diehl Martinez et al., 2024) captures a matrix's "effective dimensionality," while *Condition Number* evaluates its numerical stability.

- **Representation Similarity**: *CKA* (Kornblith et al., 2019) and *PWCCA* (Morcos et al., 2018) compare activation patterns across layers or checkpoints, revealing how internal representations evolve.

- **Norms**: *Frobenius*, *Nuclear*, and *Infinity* norms measure the scale of a tensor, spotlighting issues such as vanishing or exploding parameters.

Components. Metrics can be computed on different types of components:

- **Simple components**: Individual weight matrices, gradients, or activations from a single layer.

- **Compound components**: Higher-level structures that combine multiple model elements. One example is the OV circuit, which tracks how information flows in transformer models by combining the value and output projection matrices in self-attention layers (Elhage et al., 2021).

This two-step abstraction is designed for extensibility; new metrics and component types can be easily defined, allowing researchers to tailor analyses to specific hypotheses about language model learning. We view `pico-analyze` not as a static toolset, but as a foundation for community-driven interpretability research.

3 Case Studies

We illustrate how **Pico** enables systematic, hypothesis-driven experimentation through two case studies.

3.1 MAML

Model-Agnostic Meta-Learning (Finn et al., 2017, MAML) trains models to adapt quickly by alternating between short bursts of task-specific learning

and a global update that improves generalization. This setup encourages models to find initialization points that adapt well to new tasks. While MAML is typically used for fine-tuning, we follow prior work (Bansal et al., 2020; Li and Zhang, 2021) in applying it during pretraining.

Implementation We implemented MAML in `pico-train` by adding a lightweight inner loop that updates a classification head on masked token tasks, followed by a meta-update to the full model (Africa et al., 2025a,b). `pico-train` automatically handles distributed GPU synchronization, requiring no changes to **Pico**'s core training logic.

Analysis We evaluate MAML on Paloma perplexity and observe consistent 4–15% gains over standard pretraining. To better understand this improvement, we analyze the proportional effective rank (PER) (Diehl Martinez et al., 2024) of both weights and gradients over time. PER captures the dimensionality of a tensor's signal. In meta-learning, one concern is that inner-loop updates may overly constrain model updates to a low-dimensional subspace, potentially limiting generalization. As shown in Fig. 2, we observe synchronized troughs in PER and spikes in both loss and perplexity. This suggests that inner-loop updates temporarily compress the model's capacity into a low-rank subspace before the outer-loop update restores variance and expressivity.

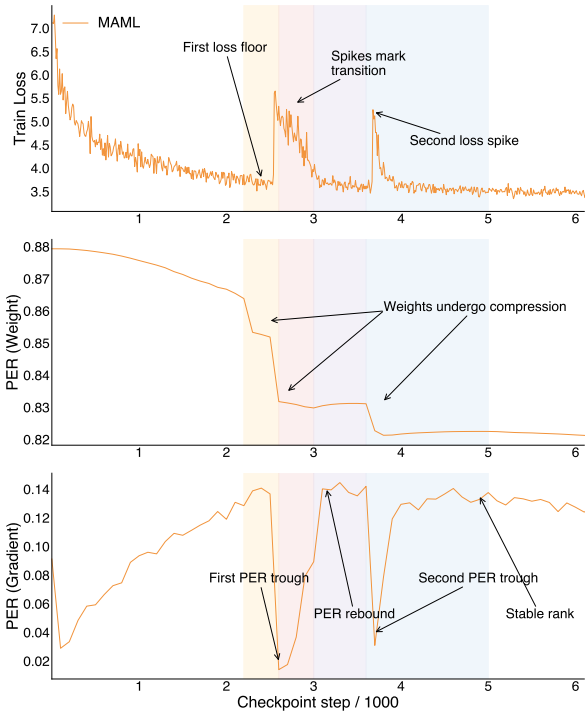


Figure 2: Training dynamics under MAML. **Top to bottom:** Training loss and proportional effective rank (PER) of weights and gradients. Sharp drops in PER align with spikes in loss. Shaded regions correspond to different observed phases in training.

New Hypothesis and Next Steps These results support the hypothesis that MAML's learning dynamics involve cycles of compression and recovery. This raises concrete follow-up questions: could adjusting the inner-loop learning rate reduce excessive compression? Would alternative meta-learning schedules or task mixes stabilize the representational space more effectively? Using Pico's modular training and built-in logging, these variants can be tested with minimal friction. By comparing learning dynamics and outcomes across runs, researchers can refine their design choices in a reproducible, hypothesis-driven loop.

3.2 ReLoRA

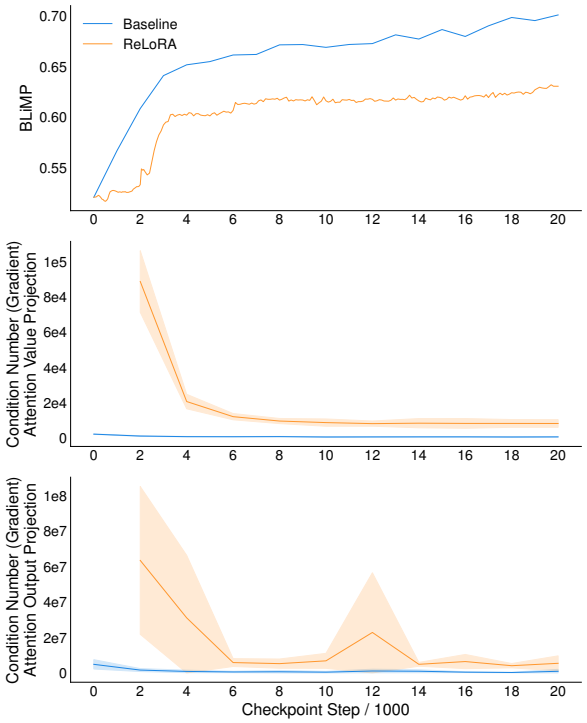


Figure 3: Training dynamics under ReLoRA. **Top to bottom:** BLiMP accuracy over time, averaged condition numbers of the gradient updates for the attention value and output projection matrices.

ReLoRA (Lialin et al., 2023) adapts LoRA (Hu et al., 2022), a fine-tuning technique that freezes pretrained weights and injects trainable low-rank matrices, into the pretraining loop. In theory, this could provide a sample-efficient way to train

Family	Size	#Tokens	Paloma ↓	HellaSwag ↑	ARC-Easy ↑	TruthfulQA ↑
Pico	11M	250B	136.17	25.62	32.79	51.75
	65M	250B	42.24	27.25	38.22	46.13
	181M	250B	30.08	30.69	44.65	41.85
	570M	250B	22.96	37.33	48.99	36.33
Pythia	14M	300B	86.64	26.15	31.31	50.14
	70M	300B	43.76	27.56	36.23	47.02
	160M	300B	29.96	30.26	43.73	44.51
	410M	300B	20.55	40.55	52.10	41.23
OPT	125M	300B	27.22	31.33	43.48	42.89
	350M	300B	20.91	36.66	44.06	41.01

Table 2: Performance of small-scale language models on four benchmarks. Lower is better for Paloma perplexity (↓); higher is better for HellaSwag, ARC-Easy, and TruthfulQA accuracies (↑).

large models by constraining updates to a low-rank subspace.

Implementation We incorporate ReLoRA into `pico-train` by adding a lightweight wrapper around attention and MLP weight matrices, and by modifying the learning rate schedule to handle periodic resets (Weiss et al., 2025). We evaluated the model on BLiMP (Warstadt et al., 2020), which we configured via a single config entry.

Analysis We find that ReLoRA surprisingly underperforms standard pretraining on BLiMP (see Fig. 3, top) (Warstadt et al., 2020). To investigate this, we analyze the condition numbers of the gradient updates across layers and training checkpoints. This metric reflects how sensitive gradient-based updates are to numerical instability, a relevant concern for methods like ReLoRA that repeatedly project updates into a low-rank subspace. As shown in Fig. 3 (bottom), ReLoRA gradients are substantially more ill-conditioned and exhibit high inter-layer variance.

New Hypothesis and Next Steps This pattern suggests that repeated low-rank resets may amplify gradient instability, undermining ReLoRA's intended efficiency gains. Several next steps follow naturally: for example, adding layerwise condition number regularization, adjusting the rank dynamically, or modifying the reset schedule to reduce instability. Because `pico-train` and `pico-analyze` modularize core components and log detailed in-situ signals, researchers can test these changes quickly, track their effects on gradient stability, and iterate systematically. This demonstrates **Pico**'s value as a scientific sandbox for implementing, analyzing, and refining design choices in the small LM regime.

4 Pico Model Suite

We train a suite of **`pico-decoder`** models at various scales on **`pretokenized-dolma`** using `pico-train`, all of which are open-sourced on our Hugging Face organization. These models range from 11M to 570M parameters, with plans to extend to billion-parameter models, and serve both as evaluations of our training pipeline and as testbeds for research on scaling laws and interpretability.

Each model is trained for 125,000 steps (covering 250B tokens). We evaluate the final model checkpoints on the Paloma benchmark (Magnusson et al., 2024), HellaSwag (Zellers et al., 2019), Arc-Easy (Clark et al., 2018) and Truthful QA (Lin et al., 2022), comparing performance against established decoder models. As shown in Table 2, our models achieve comparable results to Pythia and OPT models, despite running on an academic-level compute budget (4 nodes of 4 A100s each).

We provide a comparison of these models and their compute/storage overhead in Table 4. Reported "GPU hours" are not directly comparable across frameworks due to differences in hardware, dataloaders, and logging pipelines, but our training times are broadly consistent with existing suites. For reference, whereas `pico-large` required 7,465 A100-hours, prior reports list Pythia-1B at 4,830 A100-hours, MPT-1.3B at 7,920 A100-hours, and TinyLlama-1.1B at 3,456 A100-hours under optimized stacks (Zhang et al., 2024). Importantly, all

of our models are trained in a streaming pipeline, with datasets and checkpoints streamed from and uploaded to Hugging Face. Streaming adds data latency [2] but greatly simplifies reproducibility and removes local storage requirements. Users who prioritize throughput can instead train models with a locally cached dataset.

5 Related Literature

We survey where **Pico** sits within a growing ecosystem of frameworks that support the training and analysis of language models, ranging from optimized production libraries to interpretability toolkits.

5.1 Training Frameworks

Open-source initiatives. Initiatives by EleutherAI, including GPT-Neo, GPT-J, and the interpretability-focused Pythia suite (Biderman et al., 2023) as well as projects like the Allen Institute's OLMo (Groeneveld et al., 2024), Meta's Llama (Touvron et al., 2023), and BigScience's BLOOM (Le Scao et al., 2023), have democratized access to pretrained weights and checkpoints. While these frameworks support post-hoc investigations into phenomena such as linguistic emergence and scaling effects (Belrose et al., 2023; Gurnee et al., 2023; Michaelov and Bergen, 2023; Diehl Martinez et al., 2024), they do not capture detailed, in-training signals by default and only provide static checkpoints. Smaller frameworks like SmolLM (Allal et al., 2025), TinyLlama (Zhang et al., 2024), NanoGPT (Karpathy, 2023), and TinyStories (Eldan and Li, 2023) offer minimalistic, modular training loops that facilitate quick experimentation. However, they usually leave the implementation of fine-grained monitoring (e.g., activations or gradient flows) to the user.

Large-scale and efficient frameworks. Platforms such as NVIDIA's Megatron-LM (Narayanan et al., 2021) and Microsoft's DeepSpeed (Rasley et al., 2020) excel at distributed training for models with billions of parameters, though they lack native mechanisms for inspecting intermediate states.

5.2 Analysis Frameworks

Post-hoc model probing. Given that detailed training signals are often unavailable by default, many researchers have adopted post-hoc probing methods. Such approaches rely on external hooking libraries to intercept hidden states and attention patterns (Voita et al., 2019; Clark et al., 2019; Michel et al., 2019), looking at information flows within models to discover security or privacy vulnerabilities (Roger, 2023; Yao et al., 2024). While powerful, these methods demand significant modifications and usually depend upon pre-existing checkpoints.

Mechanistic interpretability. Recently, mechanistic interpretability (mechinterp) has gained traction as a framework for reverse-engineering neural networks at the algorithmic level (Olah et al., 2020; Elhage et al., 2021). Mechinterp focuses on localizing and characterizing the internal "circuitry" of attention heads, MLP layers, or individual neurons. A variety of mechinterp libraries, e.g., TransformerLens (Nanda and Bloom, 2022), SAELens (Bloom et al., 2024), and ACDC (Conmy et al., 2023), offer powerful tooling to dissect trained transformer models at inference time. However, these efforts typically assume that checkpoints are already available and do not natively capture the evolution of internal mechanisms throughout training.

6 Conclusion

We introduce **Pico**, a modular framework designed to help researchers study and improve small and medium-sized language models through a more systematic, scientifically grounded process. `pico-train` provides an extensible environment for training models, with built-in checkpointing that captures detailed signals needed to analyze learning dynamics. `pico-analyze` builds directly on these checkpoints, enabling researchers to test specific hypotheses about how changes to model architectures or training procedures affect convergence, sparsity, rank, and representation learning.

To further support controlled experiments and comparative studies, we open-source a suite of **`pico-decoder`** baseline models ranging from 11M to 570M parameters. These baselines give researchers a consistent starting point for evaluating new ideas or scaling laws under reproducible conditions.

By combining transparent training, detailed in-situ logging, and flexible analysis, **Pico** provides a practical sandbox for hypothesis-driven research.

[2] On our network, streaming results in approximately 80–100% slower batch loading compared to a local cache.

Limitations

The **Pico** framework is designed for interpretability and experimentation rather than optimized large-scale production training, meaning it may not efficiently scale to industrial-scale models with hundreds of billions of parameters. Additionally, the inherent overhead of systematically checkpointing intermediate activations and gradients at frequent intervals can significantly increase storage and computational costs during training.

Ethics Statement

Pico aims to facilitate transparent and reproducible research into language model interpretability and learning dynamics. To streamline experimentation, we release the `pretokenized-dolma` dataset, a preprocessed dataset in English, enabling quick and efficient model training. Additionally, the initial `pico-decoder` model suite is also trained exclusively on English-language data. We acknowledge that this emphasis on English datasets and models can inadvertently reinforce English as the dominant language in NLP and interpretability research, potentially marginalizing research on other languages. We strongly encourage and support the development and release of similarly structured, high-quality datasets and models in languages other than English. Finally, any checkpoint or artifact uploaded by **Pico** to platforms such as Hugging Face must be used responsibly, with users remaining mindful of data privacy concerns, potential biases in training data, and risks associated with misuse or harmful applications of model checkpoints.

Acknowledgments

This work was supported by a grant from the Accelerate Programme for Scientific Discovery, made possible by a donation from Schmidt Futures. Richard Diehl Martinez is supported by the Gates Cambridge Trust (grant OPP1144 from the Bill & Melinda Gates Foundation). Suchir Salhan is supported by Cambridge University Press & Assessment. David Demitri Africa is supported by the Cambridge Trust and the Jardine Foundation. A big thank you to Leshem Choshen for their helpful comments and suggestions.

References

David Demitri Africa, Suchir Salhan, Yuval Weiss, Paula Buttery, and Richard Diehl Martinez. 2025a. Meta-pretraining for zero-shot cross-lingual named entity recognition in low-resource philippine languages. *arXiv preprint arXiv:2509.02160*.

David Demitri Africa, Yuval Weiss, Paula Buttery, and Richard Diehl Martinez. 2025b. Learning dynamics of meta-learning in small model pretraining. *arXiv preprint arXiv:2508.02189*.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. 2025. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534, Online. Association for Computational Linguistics.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *Preprint*, arXiv:2303.08112.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. 2024. Saelens. `https://github.com/jbloomAus/SAELens`.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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. *CoRR*, abs/1803.05457.

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*.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Richard Diehl Martinez, Pietro Lesci, and Paula Buttery. 2024. Tending towards stability: Convergence challenges in small language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3275–3286, Miami, Florida, USA. Association for Computational Linguistics.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *Preprint*, arXiv:2305.07759.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

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*.

Patrik O Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469.

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 *International Conference on Learning Representations*.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. 2024. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*.

Niall Hurley and Scott Rickard. 2009. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741.

Andrej Karpathy. 2023. nanogpt.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Yue Li and Jiong Zhang. 2021. Semi-supervised meta-learning for cross-domain few-shot intent classification. In *Proceedings of the 1st Workshop on Meta Learning and Its Applications to Natural Language Processing*, pages 67–75, Online. Association for Computational Linguistics.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. 2023. ReLoRA: High-Rank Training Through Low-Rank Updates. In *The Twelfth International Conference on Learning Representations*.

Lightning AI. 2025. Lightning fabric. Version 2.5.1.

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)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Ian Magnusson, Akshita Bhagia, Valentin Hofmann, Luca Soldaini, Ananya Harsh Jha, Oyvind Tafjord, Dustin Schwenk, Evan Walsh, Yanai Elazar, Kyle Lo, et al. 2024. Paloma: A benchmark for evaluating language model fit. *Advances in Neural Information Processing Systems*, 37:64338–64376.

James Michaelov and Ben Bergen. 2023. Emergent inabilities? inverse scaling over the course of pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14607–14615, Singapore. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ari Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. *Advances in neural information processing systems*, 31.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–15.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506.

Fabien Roger. 2023. Large language models sometimes generate purely negatively-reinforced text. *Preprint*, arXiv:2306.07567.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Featured Certification.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Shreyas Subramanian, Vikram Elango, and Mecit Gungor. 2025. Small language models (slms) can still pack a punch: A survey. *arXiv preprint arXiv:2501.05465*.

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*.

Chien Van Nguyen, Xuan Shen, Ryan Aponte, Yu Xia, Samyadeep Basu, Zhengmian Hu, Jian Chen, Mihir Parmar, Sasidhar Kunapuli, Joe Barrow, et al. 2024. A survey of small language models. *arXiv preprint arXiv:2410.20011*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, et al. 2024. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *arXiv preprint arXiv:2411.03350*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Yuval Weiss, David Demitri Africa, Paula Buttery, and Richard Diehl Martinez. 2025. Investigating relora: Effects on the learning dynamics of small language models. *Preprint*, arXiv:2509.12960.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211.

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.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

A Default `pico-train` configurations

Category	Parameter	Default Value
Model	Model Type	`pico_decoder`
	Hidden Dimension (d_{model})	768
	Number of Layers (n_{layers})	12
	Vocabulary Size	50,304
	Sequence Length	2,048
	Attention Heads	12
	Key/Value Heads	4
	Activation Hidden Dim	3,072
	Normalization Epsilon	1×10^{-6}
	Positional Embedding Theta	10,000.0
Training	Optimizer	AdamW
	Learning Rate	3×10^{-4}
	LR Scheduler	Linear w/ Warmup
	Warmup Steps	2,500
	Gradient Accumulation Steps	128
	Max Training Steps	200,000
	Precision	BF16 Mixed
Data	Dataset Name	`pico-lm/pretokenized-dolma`
	Batch Size	1,024
	Tokenizer	`allenai/OLMo-7B-0724-hf`
Checkpointing	Auto Resume	True
	Save Every N Steps	1,000
	Learning Dynamics Layers	`"attention.v_proj"`, `"attention.o_proj"`, `"swiglu.w_2"`
	Learning Dynamics Eval Data	`pico-lm/pretokenized-paloma-tinsy`
Evaluation	Metrics	`["paloma"]`
	Paloma Dataset Name	`pico-lm/pretokenized-paloma-tinsy`
	Eval Batch Size	16
Monitoring	Logging Level	INFO
	Log Every N Steps	100

Table 3: Default configuration settings used in `pico-train`, organized by configuration category.

B `pico-decoder` models comparison

Attribute	`tiny`	`small`	`medium`	`large`
Parameter Count	11M	65M	181M	570M
Hidden Dimension (d_{model})	96	384	768	1536
Feed-forward Dim	384	1536	3072	6144
Training Time (125K Steps)	4926h	5645h	6112h	7465h
Checkpoint Time	7m42s	3m25s	1m58s	1m29s
Checkpoint Storage (Model State)	151MB	863MB	2.4GB	7.5GB
Checkpoint Storage (Learning Dynamics)	37MB	176MB	550MB	2GB

Table 4: Comparison of `pico-decoder` model variants trained with default `pico-train` configurations. Except for hidden and feed-forward dimension, all models share the training settings detailed in Table 3. Models are trained for 125,000 total training steps on 16 NVIDIA A100-SXM4-80GB GPUs. Time reported in GPU hours (h), minutes (m) and seconds (s); checkpoint time and storage reported per checkpoint step.

C Available metrics in `pico-analyze`

Metric	Description	Data Type	Category
CKA (Kornblith et al., 2019)	• Measures similarity between activations at different checkpoints • Uses kernel methods to track representation evolution	Activations	**Similarity**
PWCCA (Morcos et al., 2018)	• Measures activation similarity across training • Emphasizes important components via projections	Activations	**Similarity**
Condition Number	• Computes ratio of largest to smallest singular value • Indicates sensitivity to small input changes	Weights Activations Gradients	**Rank**
PER (Diehl Martinez et al., 2024)	• Measures entropy of normalized singular values • Indicates effective parameter usage	Weights Gradients	**Rank**
Gini Coefficient (Hurley and Rickard, 2009)	• Measures sparsity via weight distribution inequality	Weights Activations Gradients	**Sparsity**
Hoyer's Sparsity (Hoyer, 2004)	• Measures sparsity by computing ratio of L1/L2 norms	Weights Activations Gradients	**Sparsity**
Norm	• Frobenius, Nuclear, Infinity matrix norms	Weights Activations Gradients	**Norm**

Table 5: Overview of built-in metrics in `pico-analyze`. **Data Types** indicates on what types of checkpoint data the metrics can be applied. The **Category** column classifies metrics based on their primary purpose.

D Preprocessing of the `pretokenized-dolma` dataset

To prepare the `pretokenized-dolma` dataset used in our experiments, we begin by downloading the Dolma corpus and selecting a random 30% subset. The text is then tokenized using the `allenai/OLMo-7B-0724-hf` tokenizer and split into fixed-length sequences of 2049 tokens (2048 + 1 for next-token prediction). We ensure consistency across shards by chunking token streams without overlap, dropping any remainder shorter than the full sequence length.

After tokenization and chunking, we shuffle the dataset and sample a fixed number of sequences per shard, generating 100 shards in total. The resulting dataset is saved as Parquet files and uploaded to our Hugging Face organization under **`pico-lm/pretokenized-dolma`**.

To facilitate scalable loading and training, we further fine-shard the dataset into 10,000 pieces using a secondary script. These final shards are compact (78MB each), randomly shuffled, pretokenized, and ready for streaming via the Hugging Face datasets API. This preprocessing ensures that all models see data in a consistent order, which is critical for learning dynamics analysis. We release all of the scripts we use for preprocessing data in our GitHub repository.

DistaLs: A Comprehensive Collection of Language Distance Measures

Rob van der Goot[1], Esther Ploeger[2], Verena Blaschke[3], Tanja Samardžić[4]
[1]IT University of Copenhagen, robv@itu.dk
[2]Aalborg University, espl@cs.aau.dk
[3]LMU Munich & Munich Center for Machine Learning, blaschke@cis.lmu.de
[4]University of Zurich, tanja.samardzic@uzh.ch

Abstract

Languages vary along a wide variety of dimensions. In Natural Language Processing (NLP), it is useful to know how "distant" languages are from each other, so that we can inform NLP models about these differences or predict good transfer languages. Furthermore, it can inform us about how diverse language samples are. However, there are many different perspectives on how distances across languages could be measured, and previous work has predominantly focused on either intuition or a single type of distance, like genealogical or typological distance. Therefore, we propose DistaLs, a toolkit that is designed to provide users with easy access to a wide variety of language distance measures. We also propose a filtered subset, which contains less redundant and more reliable features. DistaLs is designed to be accessible for a variety of use cases, and offers a Python, CLI, and web interface. It is easily updateable, and available as a pip package. Finally, we provide a case-study in which we use DistaLs to measure correlations of distance measures with performance on four different morphosyntactic tasks.[1]

1 Introduction

Since language resources are limited for the majority of the world's languages (Joshi et al., 2020), multi-lingual Natural Language Processing (NLP) models are highly relevant. Knowing how diverse a set of languages is can be important for many crucial steps in the NLP pipeline, e.g. language selection to ensure broad coverage, predicting which source language to use for a target language, or predicting performance. However, cross-lingual performance of NLP models can be influenced by many factors. While quantitative language-level measures of distance have shown to be good predictors for performance (e.g. Lin et al., 2019; Lauscher et al., 2020), it has also been shown that factors such as data size or pre-training exposure (token overlap) can be stronger predictors than structural features (de Vries et al., 2022) or that different factors matter for different NLP tasks (Blaschke et al., 2025). However, some cross-lingual results are harder to explain, e.g. when transfer works well between intuitively distant languages such as Indonesian and Irish (Lynn et al., 2014) or when a particular language type turns out to be suitable for transfer to various target languages (Pelloni et al., 2022).

In practice, intuition, relying on properties such as language family, script, or geolocation, is often used to select a source language to transfer from. Another line of work uses more objective typological distances, commonly from lang2vec (Littell et al., 2017), to gauge language similarities, but this option too has several known issues (Toossi et al., 2024; Khan et al., 2025).

In this context, we collect and compare language distance measures more systematically. To this end, we create DistaLs, a toolkit that provides users with properties of languages, and a wide variety of language distances measuring different dimensions of diversity. We foresee at least three main use cases for DistaLs: 1) selecting languages to transfer from; 2) quickly estimating which measure of language distance has a correlation to performance; 3) measuring diversity in language selection.

DistaLs is based on the ISO 639-3 language classification standard. For each language in ISO 639-3, we aggregate language information from a variety of data sources, and use existing distance metrics, or design new ones where necessary. We complement these with text-based features. DistaLs is updateable from the original sources with a single command. We also provide an exploratory data analysis of our data sources, correlations across different distance measures, and a case-study on Universal Dependencies (UD) parsing, where we

[1]Code and data: https://bitbucket.org/robvanderg/distals

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 307–318

check which distance measure correlate with performance for four morphosyntactic tasks.

2 Distance Measures

The distances collected in DistaLs can be roughly grouped into four categories: meta-data, typological distances, wordlist-based, and text-based. For each of these sources, we only collect information for valid ISO 639-3 language codes. If only the language name is available (the case for "nlp_fate") we check if Glottolog or ISO 639-3 includes the exact match of the name. Some of the macro-languages have a variant that is the majority variant which can be assumed to be meant by the user, in these cases we convert the language label with a manually curated lookup table (Appendix A), other macro-languages are not supported due to their internal diversity. The user is informed when an automatic conversion is done, and macro-label conversion can be disabled with `–disable_macro_conversion`.

We focus on features that are available for at least 1,000 languages. Our metrics aim to cover a variety of dimensions, but within each dimension, they might lack specificity. We have for example a single feature representing phoneme inventory distance; if one is interested in more fine-grained information for a specific usecase (i.e. does a language include stressed consonants), we refer to the original data sources we included, which are designed for this exact purpose.

All metrics are converted to reflect distance (as opposed to similarity) and are normalized to be between 0.0 and 1.0. The distances are also bidirectional, so the distance of language 1 to language 2 is the same as from language 2 to language 1. The information on which the distance measures are based is also indexed and quickly retrievable so that the measures can easily be implemented differently, or the features can be used for other usecases. We provide an overview of the features in Table 1. We also provide an example of how they could be aggregated, as taking a naive average over all features will lead to undesirable weighting of certain categories, and include conflicting/redundant sources. Our selection is based on coverage, redundancy (Section 4), and quality. However, it should be noted that this is not the only possible way to aggregate the different features, and different situations will require different selections.

Category	Feature	Source	Coverage
Metadata	wiki_size	Wikipedia	7,856
	nlp_state	State and fate	2,269
	speakers	LinguaMeta	5,539
	AES	Glottolog	7,725
	loc	Glottolog	7,629
Typology	lang2vec	URIEL	3,910
	lang2vec_knn	URIEL	3,910
	PHOIBLE	PHOIBLE	2,078
	grambank_all	Grambank	2,326
	grambank_.*	Grambank	2,326
	glot_tree	Glottolog	7,856
	scripts	LinguaMeta, GlotScript	6,431
Wordlists	**ASJP**	ASJP	6,117
	concepts	Conceptualizer	1,274
Text-driven	whitespace	LTI LangID	2,525
	punctuation	LTI LangID	2,525
	char_distr.	LTI LangID	2,525
	textcat	LTI LangID	2,525

Table 1: All features provided by DistaLs. Bold: recommended to use for average for category. grambank_.* refers to the sub-categories within Grambank. Coverage of wiki_size is large, becuase if there is no Wikipedia for a language, we assign the size 0 (happens for 7,570 languages).

Below, we describe each feature. Appendix B contains further implementation details.

2.1 Metadata

Wikipedia We use the number of articles per language as a statistic of a language, which can be considered a proxy to online presence of languages. The distance metric is the proportional difference in size: $1 - min(size1, size2)/max(size1, size2)$.

State and Fate The amount of resources available for a language can be a strong predictor for performance for universally trained models. There are many different catalogues of resour-ces, and also many less standardized resources. We here use the categorization provided by Joshi et al. (2020); they divide languages into one of 6 groups based on the availability of raw text data, and annotated NLP datasets. The groups are ranked, so we use the distance in rank as our metric: $(max - min)/5$.

LinguaMeta LinguaMeta (Ritchie et al., 2024) is an effort to calculate metadata of languages into a unified format. It is combining a variety of existing resources, including manual corrections/additions where possible. We extract the **number of speakers** and **scripts** from LinguaMeta. The number of speakers only counts L1 speakers, and uses a variety of sources; CLDR,[2] Wikipedia, and Google-

[2] `cldr.unicode.org`

internal information. The scripts are mostly from internal Google data which is used for keyboard selection in their products. We complement the scripts information with data from GlotScript (Kargaran et al., 2024), if there is more than 1 script from both sources, we only use the intersection. For the scripts, we use $1 - \%overlap$ as metric, and for the speakers we use the same formula as for the Wikipedia size.

Glottolog (Hammarström et al., 2024) is a database containing information and references for different languages, dialects, and families. We here use Agglomerated Endangerment Status (AES). The endangerment status has 6 ranked classes, we use the same formula as for "state and fate".

2.2 Typology

lang2vec We use the average values over all data sources for the syntax, phonology, and inventory categories from the URIEL database through the lang2vec toolkit (Littell et al., 2017), which is in turn based on WALS (Dryer and Haspelmath, 2013), SSWL (Collins and Kayne, 2011), Ethnologue (Campbell and Grondona, 2008), and PHOIBLE (Moran and McCloy, 2019). These values are concatenated and used as feature vector for a language. We additionally use lang2vec's imputation method for missing features based on a k-nearest-neighbours selection of similar languages. The distance metric is cosine distance, where we remove features from both languages if a feature is missing for one of the languages. Reproducibility of the lang2vec distances is non-trivial (Toossi et al., 2024; Khan et al., 2025), so we calculate the cosine distance based on their representation vectors ourselves.

PHOIBLE (Moran and McCloy, 2019) is a cross-linguistic phonological database. It contains phoneme inventories based on International Phonetic Alphabet (International Phonetic Association, 2005) collected from a wide variety of sources. We use the set of the defined *GlyphId*s for each language as a representation, and use the % of overlap between these sets as a distance metric.

Grambank (Skirgård et al., 2023) is a database containing morphosyntactic information about languages. It contains 195 features with a higher language coverage compared to lang2vec. Similar to Ploeger et al. (2024), we first binarize the data, and then we take the euclidean distance ignoring empty features. Languages with fewer than 25% of the features covered are removed. We divide this distance by the square root of the total number of features to make it range between 0–1.

Glottolog Besides the AES (described above), we also extract family trees from Glottolog. We calculate a distance based on the position in the tree structure. If two languages are in different language families, the distance is maximal (1.0), if the languages are in the same tree, we calculate the number of overlapping edges divided by the depth of the deepest language of the two.

2.3 Wordlist-based metrics

ASJP Automated Similarity Judgment Program (ASJP) is a database containing standardized word lists of concepts in many languages (Wichmann et al., 2022). Their word list is based on the Swadesh lists (Swadesh, 1955). Both lists are created to cover concepts that are expected to exist in cultures and languages all over the world. For each concept, ASJP collected a phonetic description of the concept in each language. We follow their original implementation (Bakker et al., 2009) and use average normalized Levenshtein distance over the phonetic sequences of the concepts.

Conceptualizer Liu et al. (2023) use 51 concepts from the bible combined with 32 concepts defined in Swadesh lists to compare representations of different concepts in different languages. They model the concepts as a bipartite graph, in which a concept (represented as a set of English strings) links to all correlated translations. We use the language distance metric as proposed by Liu et al. (2023); the cosine distance over the representations of each concept for each language, where a concept representation is the number of steps a concept needs to get to the English concept.

2.4 Text-driven distances

We use a combination of the GlotLID (Kargaran et al., 2023) and the LTI LangID corpus (Brown, 2014) as our source data because of their large language coverage. These datasets are combinations of a variety of sources, but the majority of data comes from Wikipedia and a variety of bible translation sets. We take 1,000 lines per script (equal amount of each source) of each available data source to represent a language. We apply NFC normalization before collecting our features.

Character categories There are two categories of characters that are commonly used across different scripts: **whitespace characters**, and **punctuation characters**. The amount of usage of these categories can be used to distinguish languages with whitespace-based writing systems, lengths of words within these, and the amount of punctuation information. For both categories, we use the definition from the huggingface library (specifically, the `_is_whitespace` and `_is_punctuation` function) for classification. We then convert the percentages to a distance score through the following formula: $1 - min(prob1, prob2)/max(prob1, prob2)$

Character distribution distance We first extract the character distributions from each language. Following the original LTI LangID Corpus we use UTF-8 encoding for defining characters, and for each character estimate their frequency as a probability. We then use the Jensen-Shannon distance over the union of the character sets of both languages.

Textcat distance Cavnar et al. (1994) proposed to use n-gram frequency-lists for language classification. Specifically, they extract the 300 most common 1–5 character n-grams, and sort them by frequency to represent a language. For an input, they then create a similar frequency list, and calculate a distance to the representation of each language in the training data to obtain a similarity ranking. We use the same setup to calculate distances across texts of different languages, but use the 400 most common n-grams, following van Noord (1997).

3 Interface

DistaLs provides language data in the following three ways:

- Pre-calculated distances: we provide all metrics for all language pairs in csv files.
- Database with language information: this will automatically be downloaded by the package if it is not found. The code uses this database to calculate the distances.
- Scrape the data: download all data sources with a single bash script, and then use this to populate a database (as described above). This allows for easy updating of all data sources.

DistaLs provides three different interfaces (besides the pre-calculated distances), which are described in the next sections.

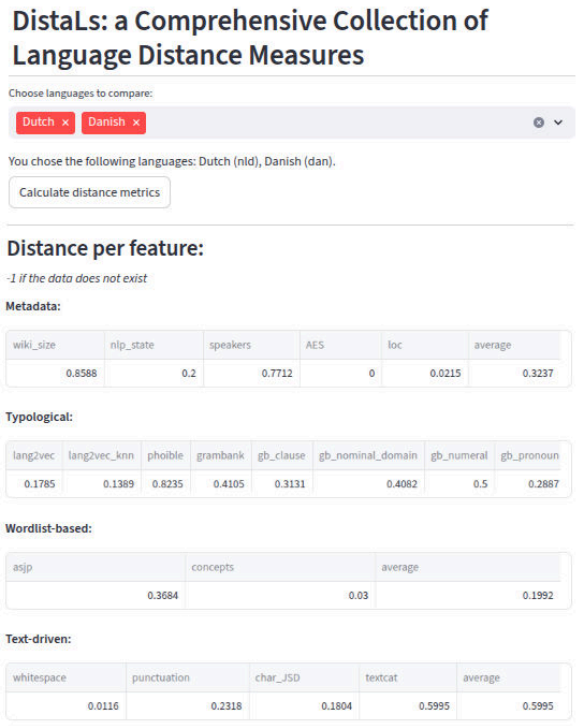


Figure 1: Screenshot of the web interface.

3.1 Web interface

We provide an online interface to DistaLs on `https://distals.streamlit.app/`. The user is presented with a text field, in which language names can be added (search results will pop up for easy selection after typing). After selecting a number of languages, the users clicks on the button or presses enter, and after a short wait the distances will be shown per category (see Figure 1).

3.2 Command-line Interface (CLI)

DistaLs is available as a pip package. After installing the package, the main functionality is accessed through the parameter `-langs`. The user specifies a list of languages, which can be ISO639-3 codes, ISO639-2 codes, or language names (which will be converted according to the procedure described in Section 2). DistaLs will first print all the information it has available for each language. If there are two languages defined, it will then print all the distances for each category, including their average. When information for a feature is not available for both languages, it will print a –1 value. If more than two languages are included, it will print language×language matrices. An example of usage and its output can be seen in Figure 2.

The CLI command also provides an interface

```
$ distals --langs fry dan
loading from: ./distals-db.pickle.gz
7856 languages loaded
========================================
Information for fry
wiki_size: 57,027
nlp_state: 1. The Scraping-Bys
speakers: 740,000
AES: 5. not endangered
loc: (5.86091, 53.143)
lang2vec: [1.0, 1.0, 0.0, ...]
lang2vec_knn: [1.0, 1.0, 0.0, ...]
phoible: ['0061', '0061+0069', '0061+0075', ...]
grambank: {'GB020': 1, 'GB021': 1, 'GB022': 1, ...}
glot_tree: ["'Western Frisian [west2354][fry]-1-'", "'
      Westlauwers-Terschelling Frisian [west2902]'", "'Modern
       West Frisian [mode1264]'", ...]
scripts: {'latn'}
asjp: [['1', 'ik'], ['2', 'do, yo'], ['3', 'vEi'], ...]
whitespace: 0.160835
punctuation: 0.031726
char_JSD: {' ': 0.1608, 'e': 0.1195, 'n': 0.0754, ...}
textcat: [' ', 'e', 'n', ...]

========================================
Information for dan
wiki_size: 308,911
nlp_state: 3. The Rising Stars
speakers: 5,510,600
AES: 5. not endangered
loc: (9.36284, 54.8655)
lang2vec: [1.0, 0.0, 0.0, ...]
lang2vec_knn: [1.0, 0.0, 0.0, ...]
phoible: ['0061', '0062+0325', '0062+0325+02B0', ...]
grambank: {'GB020': 1, 'GB021': 1, 'GB022': 1, ...}
glot_tree: ["'Danish [dani1285][dan]-1-'", "'South
      Scandinavian [sout3248]'", "'North Germanic [nort3160
      ]'", "'Northwest Germanic [nort3152]'", "'Germanic [
      germ1287]'", "'Classical Indo-European [clas1257]'", "'
      Indo-European [indo1319]'"]
scripts: {'latn'}
asjp: [['1', 'yoy'], ['2', 'du'], ['3', 'vi'], ...]
whitespace: 0.156298
punctuation: 0.028514
char_JSD: {' ': 0.1563, 'e': 0.1249, 'r': 0.0675, ...}
textcat: [' ', 'e', 'r', ...]

========================================
Distances between fry and dan (-1 if feature not available)
METADATA
wiki_size: 0.8154
nlp_state: 0.4000
speakers: 0.8657
AES: 0.0000
loc: 0.0149
average: 0.5203

TYPOLOGY
lang2vec: 0.1598
lang2vec_knn: 0.1204
phoible: 0.8148
grambank: 0.3841
gb_clause: 0.3742
gb_nominal_domain: 0.3482
gb_numeral: 0.5000
gb_pronoun: 0.0000
gb_verbal_domain: 0.4644
glot_tree: 0.5325
scripts: 0.0000
average: 0.5995

WORDLISTS
asjp: 0.3397
concepts: 0.0400
average: 0.1898

TEXTBASED
whitespace: 0.0282
punctuation: 0.1012
char_JSD: 0.1979
textcat: 0.5859
average: 0.3919
```

Figure 2: Example output of DistaLs. It first reports information about the provided language(s), and then reports all features per category.

to the updating of the database. There are three separate arguments (for language labels and names, databases, and text-based features), which can be used separately or jointly. The resulting database

```
>>> from distals import distals
>>> model = distals.Distals()
>>> model.get_dists('nld', 'cmn')
{'metadata': {'wiki_size': 0.99378,
              'nlp_state': 0.2,
              'speakers': 0.98131,
              'AES': 0.0,
              'loc': 0.39121,
              'average': 0.39377},
 'typology': {'lang2vec': 0.31654,
              'lang2vec_knn': 0.33795,
              'phoible': 0.82278,
              'grambank': 0.58478,
              'gb_clause': 0.55470,
              'gb_nominal_domain': 0.59761,
              'gb_numeral': 0.0,
              'gb_pronoun': 0.64550,
              'gb_verbal_domain': 0.60302,
              'glot_tree': 1.0,
              'scripts': 0.66667,
              'average': 0.80252},
 'wordlists': {'asjp': 0.49636,
               'concepts': 0.08,
               'average': 0.28818}
 'textbased': {'whitespace': 0.21244,
               'punctuation': 0.67855,
               'char_JSD': 0.54401,
               'textcat': 0.87235,
               'average': 0.87235}
}
```

Figure 3: Example output of DistaLs when used in Python.

is stored in a dictionary which is saved in a compressed pickle file. The toolkit will automatically download a recent database from the repository if `--database_path` is not specified. The code can also be ran directly from the repository (`python3 src/distals/distals.py`) without installation.

3.3 Python

For easy integration into other projects and codebases, we also provide a python interface. The information stored for each language is directly available from the DistaLs database, which is a python dictionary in a pickle file. One can also import DistaLs to get direct access to the distances, which can be returned as a list or as a dictionary (containing the four main categories as a hierarchy). The language names and code conversion scripts are also available after loading DistaLs. Example usage is shown in Figure 3. Updating the DistaLs database is done by first running a bash script that downloads/updates the data, and then the python package has functionality to update through a single command.

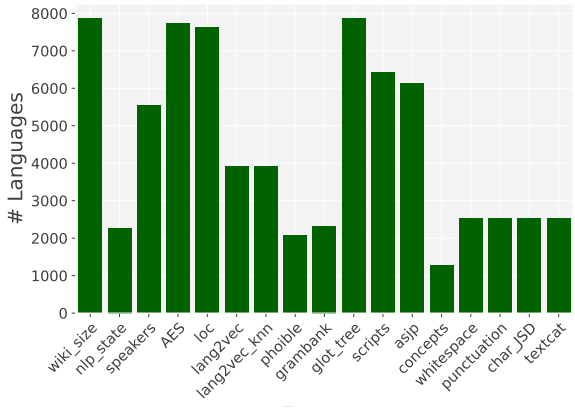


Figure 4: The number of languages (y-axis) supported by each feature (x-axis).

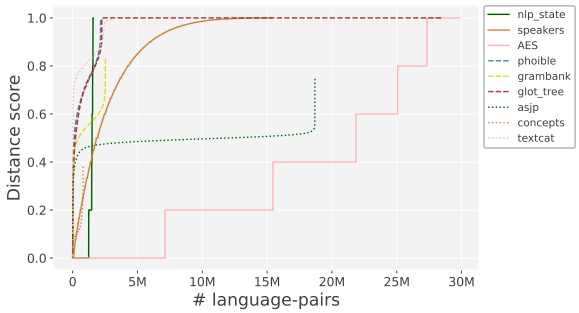


Figure 5: The cumulative probabilities of selected features.

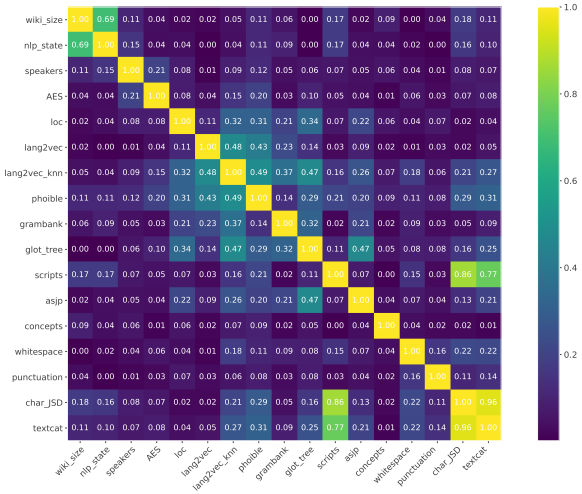


Figure 6: Pearson correlations across all features (except the grambank sub-categories). P-values in Appendix D.

4 Exploratory Data Analysis

To get a clearer picture of the coverage, we count the number of features supported by each language, and plot the number of languages with N supported features (Figure 4). This shows that there are a few features for which almost all languages are covered. These are mainly meta-features and genealogical information. Many features have a coverage around 2,000 languages, which often have a large overlap in languages. DistaLs contains 437 languages that have information for all 17 features.

We normalized all features to have a value between 0 and 1, but different features might still have different distributions within this range. Hence, for each feature, we sort the values of all language-pairs, disregarding the missing values. We then plot the scores to get an overview of how the distributions differ. Results (Figure 5) show that there is indeed a disparity in the distributions of the probabilities, with AES having lower distances, because there are only 6 possible labels, and the text-based feature (textcat) and typology based features (PHOIBLE, Grambank, and Glot_tree) having larger distances for most pairs.

Some of the features have a similar goal, so they can be expected to correlate. For example, the main categories in the typology category all aim to capture typological distances. If features have a large overlap, some of them can be left out for the sake of simplicity and efficiency. We therefore perform a correlation study for feature-pairs across all language-pairs. For each pair of features, the languages included can be different (based on data availability). Hence, the results across features are not directly comparable, but should give a rough idea of which features contain similar information.

Results (Figure 6) show only a few strong Pearson correlations (i.e. > 0.5), which are all within the main categories we defined in Section 2, except script which has a strong correlation to the text-based distances. Within the typology category, there are many moderate correlations (0.3–0.5), and across categories there are mostly correlations close to 0, where mainly the text-based distances (char JSD and textcat) have some weak correlations (~0.2) across the other features.

5 Case Study

Inspired by the studies of de Vries et al. (2022), Samardžić et al. (2022) and Blaschke et al. (2025), who analyze the effect of certain language distances on downstream NLP model performance, we execute a similar case study on cross-lingual transfer with our extended feature set. We train a multi-task model on the first 10k words of each UD v2.15 treebank (Nivre et al., 2020) that has a training split resulting in 64 source and 126 target languages.

	UPOS	Dep	UFeats	Lemma
nlp_state	−0.42 *	−0.37 *	−0.08 *	−0.12 *
speakers	−0.24 *	−0.20 *	−0.02	−0.05 *
AES	−0.32 *	−0.23 *	−0.03	−0.05 *
loc	−0.29 *	−0.28 *	0.02	0.02
phoible	−0.19 *	−0.17 *	0.02	−0.02
grambank	−0.42 *	−0.45 *	−0.20 *	−0.11 *
glot_tree	−0.27 *	−0.27 *	−0.03	−0.07 *
asjp	−0.26 *	−0.28 *	−0.14 *	−0.11 *
concepts	−0.32 *	−0.31 *	0.01	−0.10 *
textcat	−0.21 *	−0.23 *	−0.06 *	−0.06 *

Table 2: Pearson's r between language distance and accuracy or labelled attachment score (Dep). *p<0.05.

We use all treebanks for part-of-speech tagging (UPOS) and dependency parsing (Dep), and add lemmatization (Lemma) and morphological tagging (UFeats) when available. For languages with multiple treebanks, we average the results. We use the MaChAmp toolkit (van der Goot et al., 2021) v0.4.2 with default hyperparameters. We train with XLM-R large (Conneau et al., 2020) and Glot500 (Imani et al., 2023) as an encoder, and report the average results (trends were highly similar across models).

We evaluate the correlations on the subset of most informative features from Section 2 (bold in Table 2). The correlations (Table 2) are weak for the morphological tasks (UFeats/Lemma), but much stronger for the syntactic tasks (UPOS/Dep), whose performance can be better predicted with our distance features. Interestingly, the most strongly correlated distances are scattered across our distance categories.

6 Comparison to Other Toolkits

We compare existing toolkits for estimating language diversity in Table 3. DistaLs covers the most categories of distances, but some other toolkits have more functionality within a specific category. For example, Delta[3] and LangDive (Samardzic et al., 2024) focus on the diversity of datasets with respect to linguistic and syntactic information. Within the domain of syntax, they will provide a much more granular perspective on distance, but at the cost of a lower language coverage. QwanQwa[4] instead focuses on aligning metadata across different language-code systems, and typdiv (Ploeger et al., 2024) provides metrics for assesing typological diversity. LangRank (Lin et al., 2019) directly focuses on predicting transfer performance, providing a variety of metrics.

7 Conclusion

We propose DistaLs, a toolkit that aggregates language information from a variety of sources, and provides distance measures based on this. DistaLs contains a variety of easy to use interfaces, a webpage, csv files, python, and a command-line interface. It includes a wide variety of measures covering a variety of dimensions of "distance", all with a high language coverage. We showed its usefulness by reporting correlations of the features with four morphosyntactic tasks. Based on this, we conclude that syntactic tasks have higher correlations than morphological tasks (i.e., performance transfer is easier to predict), and that no features are close to a perfect correlation, however, features with a moderate correlations are quite diverse.

Acknowledgements

We thank Barbara Plank, Joakim Nivre, Jörg Tiedemann, Steven Moran, and the members of NLPnorth for their feedback. We also thank Ralf Brown for helping with cleaning LTI-LangID.

This work is supported by the CA21167 COST action UniDive, funded by COST (European Cooperation in Science and Technology).

[3] https://gitlab.lisn.upsaclay.fr/esteve/delta
[4] https://github.com/WPoelman/qwanqwa

Toolkit	Focus	Lang. cat.	Coverage	Typology	Metadata	Wordlists	Textbased	Updatable	Interface
Lang2vec	typology	ISO 639-3	4,005	✓	✓	✗	✗	✗	python, CLI
Delta	syntactic diversity	—	156	✓	✗	✗	✗	✓	python, c
LangDive	dataset diversity	ISO 639-3	—	✗	✗	✗	✗	✓	python
QwanQwa	metadata	many	7,511	✓	✓	✗	✗	✗	python
typdiv	diversity	glottocodes	—	✓	✗	✗	✗	✓	python, CLI
LinguaMeta	metadata	BCP-47,ISO 639-3	7,512	✓	✓	✗	✗	✗	tsv file
LangRank	multiple	ISO 639-2	—	✓	✓	✗	✓	✗	python
DistaLs	multiple	ISO 639-3	1,271–7,855	✓	✓	✓	✓	✓	python, CLI, web

Table 3: Comparison of existing toolkits for measuring language diversity. The main categories refer to the ones described in Section 2. A '—' in coverage means that these toolkits are supposed to be used with datasets to estimate the distances. Updateable refers to automatically updateable (i.e. through a single command).

Limitations

We limited the language coverage to the ISO 639-3 standard, as it is one of the most widely used set of language labels. However, this standard is known to have biases (Morey et al., 2013). At the same time, we ignore in-language variation, and we make the (flawed) assumption that the textual data we use serves as a proxy for a representation of the language as a whole. Also, the data sources we include are carefully chosen to have a wide coverage, but there is definitely more information for high-resource languages. The case-study on the UD data also has a biased sub-selection of languages, which have a higher coverage in Western languages.

Furthermore, each feature can be seen as an abstraction to actual diversity as it occurs within a language. This is a necessary step to take when providing smaller numbers of distance metrics, but it is obscuring a lot of potentially interesting information. If more detailed information on a specific dimension of language is required, we refer to the original data sources.

While we were careful in selecting the information sources, with data on this scale there are undoubtedly errors in the data. We have done an automatic and manual correction of some of the LTI-LangID data in cooperation with Ralf Brown, and have done inspection, cleaning and merging of the other sources where possible, but there are of course many data points that are hard to verify. The text-based features are also biased in domain. Most of the data comes from bible translations.

References

Dik Bakker, André Müller, Viveka Velupillai, Søren Wichmann, Cecil H Brown, Pamela Brown, Dmitry Egorov, Robert Mailhammer, Anthony Grant, and Eric W Holman. 2009. Adding typology to lexicostatistics: A combined approach to language classification.

Verena Blaschke, Masha Fedzechkina, and Maartje ter Hoeve. 2025. Analyzing the effect of linguistic similarity on cross-lingual transfer: Tasks and experimental setups matter. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8653–8684, Vienna, Austria. Association for Computational Linguistics.

Cecil H. Brown, Eric W. Holman, Søren Wichmann, and Viveka Velupillai. 2008. Automated classification of the world's languages: a description of the method and preliminary results. *Language Typology and Universals*, 61(4):285–308.

Ralf Brown. 2014. Non-linear mapping for improved identification of 1300+ languages. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 627–632, Doha, Qatar. Association for Computational Linguistics.

Lyle Campbell and Verónica Grondona. 2008. Ethnologue: Languages of the world. *Language*, 84(3):636–641.

William B Cavnar, John M Trenkle, and 1 others. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, page 14, Las Vegas, NV.

Chris Collins and Richard Kayne. 2011. Syntactic structures of the world's languages. New York University, New York.

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.

Wietse de Vries, Martijn Wieling, and Malvina Nissim. 2022. Make the best of cross-lingual transfer: Evidence from POS tagging with over 100 languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7676–7685, Dublin, Ireland. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online (v2020.4)*. Zenodo.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2024. Glottolog 5.0.

Ayyoob Imani, Peiqin Lin, Amir Hossein Kargaran, Silvia Severini, Masoud Jalili Sabet, Nora Kassner, Chunlan Ma, Helmut Schmid, André Martins, François Yvon, and Hinrich Schütze. 2023. Glot500: Scaling multilingual corpora and language models to 500 languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1082–1117, Toronto, Canada. Association for Computational Linguistics.

International Phonetic Association. 2005. International phonetic alphabet.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Amir Hossein Kargaran, Ayyoob Imani, François Yvon, and Hinrich Schuetze. 2023. GlotLID: Language identification for low-resource languages. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6155–6218, Singapore. Association for Computational Linguistics.

Amir Hossein Kargaran, François Yvon, and Hinrich Schütze. 2024. GlotScript: A resource and tool for low resource writing system identification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7774–7784, Torino, Italia. ELRA and ICCL.

Aditya Khan, Mason Shipton, David Anugraha, Kaiyao Duan, Phuong H. Hoang, Eric Khiu, A. Seza Doğruöz, and En-Shiun Annie Lee. 2025. URIEL+: Enhancing linguistic inclusion and usability in a typological and multilingual knowledge base. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6937–6952, Abu Dhabi, UAE. Association for Computational Linguistics.

Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing transfer languages for cross-lingual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3125–3135, Florence, Italy. Association for Computational Linguistics.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. Association for Computational Linguistics.

Yihong Liu, Haotian Ye, Leonie Weissweiler, Philipp Wicke, Renhao Pei, Robert Zangenfeind, and Hinrich Schütze. 2023. A crosslingual investigation of conceptualization in 1335 languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12969–13000, Toronto, Canada. Association for Computational Linguistics.

Teresa Lynn, Jennifer Foster, Mark Dras, and Lamia Tounsi. 2014. Cross-lingual transfer parsing for low-resourced languages: An Irish case study. In *Proceedings of the First Celtic Language Technology Workshop*, pages 41–49, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Steven Moran and Daniel McCloy, editors. 2019. *PHOIBLE 2.0*. Max Planck Institute for the Science of Human History, Jena.

Stephen Morey, Mark W. Post, and Victor A. Friedman. 2013. The language codes of iso 639: A premature, ultimately unobtainable, and possibly damaging standardization.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Olga Pelloni, Anastassia Shaitarova, and Tanja Samardzic. 2022. Subword evenness (SuE) as a predictor of cross-lingual transfer to low-resource languages. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7428–7445, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Esther Ploeger, Wessel Poelman, Andreas Holck Høeg-Petersen, Anders Schlichtkrull, Miryam de Lhoneux, and Johannes Bjerva. 2024. A principled framework for evaluating on typologically diverse languages. *Preprint*, arXiv:2407.05022.

Sandy Ritchie, Daan van Esch, Uche Okonkwo, Shikhar Vashishth, and Emily Drummond. 2024. LinguaMeta: Unified metadata for thousands of languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10530–10538, Torino, Italia. ELRA and ICCL.

Tanja Samardzic, Ximena Gutierrez, Christian Bentz, Steven Moran, and Olga Pelloni. 2024. A measure for transparent comparison of linguistic diversity in multilingual NLP data sets. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3367–3382, Mexico City, Mexico. Association for Computational Linguistics.

Tanja Samardžić, Ximena Gutierrez-Vasques, Rob van der Goot, Max Müller-Eberstein, Olga Pelloni, and Barbara Plank. 2022. On language spaces, scales and cross-lingual transfer of UD parsers. In *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*, pages 266–281, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Hedvig Skirgård, Hannah J. Haynie, Damián E. Blasi, Harald Hammarström, Jeremy Collins, Jay J. Latarche, Jakob Lesage, Tobias Weber, Alena Witzlack-Makarevich, Sam Passmore, Angela Chira,

Luke Maurits, Russell Dinnage, Michael Dunn, Ger Reesink, Ruth Singer, Claire Bowern, Patience Epps, Jane Hill, and 86 others. 2023. Grambank reveals the importance of genealogical constraints on linguistic diversity and highlights the impact of language loss. *Science Advances*, 9(16).

Morris Swadesh. 1955. Towards greater accuracy in lexicostatistic dating. *International journal of American linguistics*, 21(2):121–137.

Hasti Toossi, Guo Huai, Jinyu Liu, Eric Khiu, A. Seza Doğruöz, and En-Shiun Lee. 2024. A reproducibility study on quantifying language similarity: The impact of missing values in the URIEL knowledge base. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pages 233–241, Mexico City, Mexico. Association for Computational Linguistics.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021. Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.

Gertjan van Noord. 1997. Textcat.

Wichmann, Søren, Eric W. Holman, and Cecil H. Brown. 2022. The ASJP database (version 20).

Appendix

A Macro Language Conversion

Table 4 reports the list of manually added conversions of macro labels.

Macro	Language
est	ekk
zho	cmn
grn	gug
toki	tok
nep	npi
lav	lvs
ara	arb
ori	ory
msa	zlm
kom	kpv

Table 4: Conversion of macro labels to language labels.

B Implementation Details for Distance Measures

Below, we describe implementation details of version 0.1.1 of DistaLs.

B.1 Metadata

Wikipedia We collect the page counts from the webpage `https://en.wikipedia.org/wiki/List_of_Wikipedias`. Where possible, we use the `language` attribute from the column with sample text in each wiki's language. For the relatively few wikis where this information is not available, we derive the language from the wikicode. We manually verified that this fallback option results in the correct ISO639-3 codes as of September 12, 2025.

B.2 Typology

lang2vec We use the `syntax_average+phonology_average+inventory_average` category for original features, and the `syntax_knn+phonology_knn+inventory_knn` category for the KNN completed features.

We only compare lang2vec vectors for language pairs where at least 25% of the features have values for both languages. Users can change this threshold.

PHOIBLE Some languages have multiple phoneme inventories on PHOIBLE (coming from different sources and/or describing different dialects/sociolects).[5] If this applies to one or both languages, we calculate the distances between each inventory of the first language and each inventory of between each pair of the second language, and return the minimum distance. We do not take into account the allophone information.

Grambank We only compare Grambank feature vectors for language pairs where at least 25% of the features have values for both languages. Users can change this threshold.

Glottolog Where pseudo-families[6] are used in Glottolog for bookkeeping purposes, we remove the pseudo-families. For instance, the family tree path for German Sign Language is originally *German Sign Language – DSGic – L1 Sign Language – Sign Language*, but we remove the last two nodes.

B.3 Wordlist-based metrics

ASJP Some languages have multiple word form entries for one concept (because a wordlist contains multiple entries for a concept, and/or because ASJP contains multiple wordlists with the same language code). In such cases, we use the word form with the lowest Levenshtein distance to the other word form it is compared to.

We ignore vowel and consonant modifiers (cf. Brown et al., 2008) when calculating Levenshtein distances. In addition to normalizing the Levenshtein distance by the length of the longer word, we normalize it by the average distance between entries with *different* meanings in the two wordlists being compared. We only compare ASJP wordlists for language pairs where at least 25% of the concepts have word form entries for both languages. Users can change this threshold.

C Licenses

Licenses for each data source are listed in Table 5. DistaLs is released under the CC BY-SA 4.0, as it is required by some of the included data sources.

D P-values of Correlations Across Features

Figure 7 has the same shape as Figure 6 in the paper, but contains the p-values instead of the actual

[5] `https://phoible.org/faq#why-do-some-languages-have-multiple-entries-in-phoible`

[6] `https://glottolog.org/meta/glossary#sec-pseudofamilies`

Category	Feature	Source	License
Metadata	wiki_size	Wikipedia	CC BY-SA
	nlp_state	state and fate	—
	speakers	LinguaMeta	CC BY-SA 4.0
	scripts	LinguaMeta, GlotScript	MIT License
	AES	Glottolog	CC BY 4.0
	loc	Glottolog	CC BY 4.0
Typology	lang2vec	URIEL	CC BY-SA 4.0
	lang2vec_knn	URIEL	CC BY-SA 4.0
	PHOIBLE	PHOIBLE	GPL 3
	grambank_all	Grambank	CC BY 4.0
	grambank_.*	Grambank	CC BY 4.0
	glot_tree	Glottolog	CC BY 4.0
Wordlists	ASJP	ASJP	CC BY 4.0
	concepts	Conceptualizer	—
Text-driven	whitespace	LTI LangID	CC
	punctuation	LTI LangID	CC
	char_distr.	LTI LangID	CC
	textcat	LTI LangID	CC

Table 5: Licenses for all data sources included in DistaLs.

correlation. The values are generally very low, this is because of the large data size.

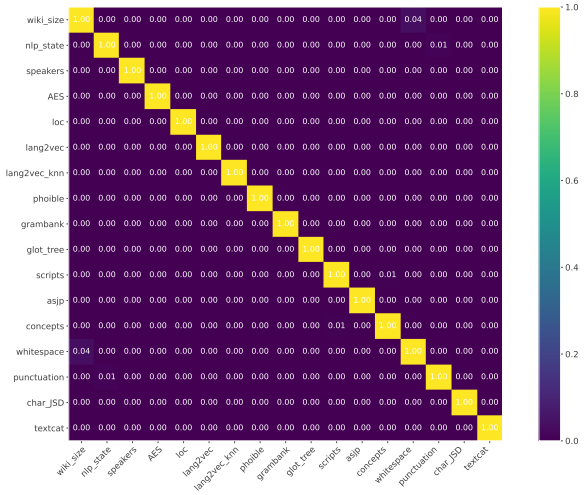


Figure 7: P-values of correlations across features.

MedTutor: A Retrieval-Augmented LLM System for Case-Based Medical Education

Dongsuk Jang[1,3] **Ziyao Shangguan**[1] **Kyle Tegtmeyer**[2]
Anurag Gupta[2] **Jan Czerminski**[2] **Sophie Chheang**[2] **Arman Cohan**[1]
[1]Department of Computer Science, Yale University
[2]Department of Radiology and Biomedical Imaging, Yale School of Medicine
[3]Interdisciplinary Program for Bioengineering, Seoul National University
{james.jang,ziyao.shangguan,arman.cohan}@yale.edu

Code Demo Video Dataset

Abstract

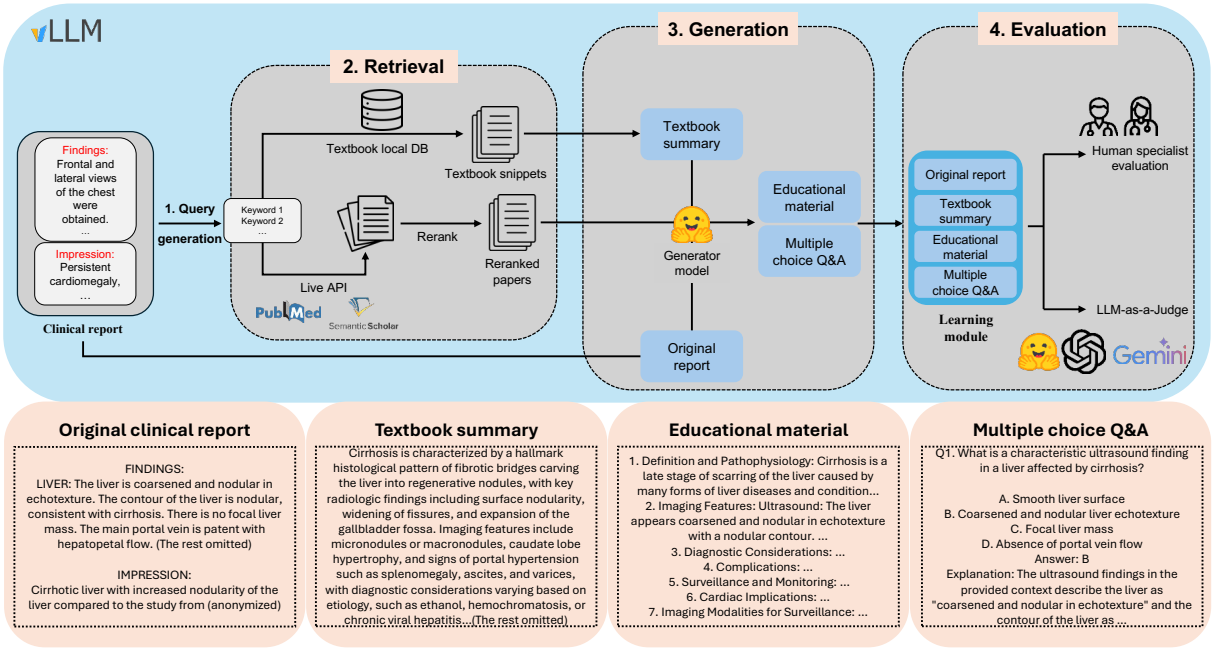
The learning process for medical residents presents significant challenges, demanding both the ability to interpret complex case reports and the rapid acquisition of accurate medical knowledge from reliable sources. Residents typically study case reports and engage in discussions with peers and mentors, but finding relevant educational materials and evidence to support their learning from these cases is often time-consuming and challenging. To address this, we introduce **MedTutor**, a novel system designed to augment resident training by automatically generating evidence-based educational content and multiple-choice questions from clinical case reports. MedTutor leverages a Retrieval-Augmented Generation (RAG) pipeline that takes clinical case reports as input and produces targeted educational materials. The system's architecture features a hybrid retrieval mechanism that synergistically queries a local knowledge base of medical textbooks and academic literature (using PubMed, Semantic Scholar APIs) for the latest related research, ensuring the generated content is both foundationally sound and current. The retrieved evidence is filtered and ordered using a state-of-the-art reranking model and then an LLM generates the final long-form output describing the main educational content regarding the case-report. We conduct a rigorous evaluation of the system. First, three radiologists assessed the quality of outputs, finding them to be of high clinical and educational value. Second, we perform a large-scale evaluation using an LLM-as-a Judge to understand if LLMs can be used to evaluate the output of the system. Our analysis using correlation between LLMs outputs and human expert judgments reveals a moderate alignment and highlights the continued necessity of expert oversight.

1 Introduction

The training of medical residents is an intensive learning process, built upon the foundation of studying and interpreting thousands of case reports. Residents routinely engage with clinical cases through discussions with peers and mentors, analyzing findings and differential diagnoses to deepen their understanding. However, while direct feedback from attending physicians is invaluable, the process of finding relevant educational material and supporting evidence for specific cases is often time-consuming and inconsistent (Rogers et al., 2019; Daniel et al., 2020). The sheer volume of medical literature and the challenge of identifying pertinent resources for each case can limit the depth of learning that residents achieve from their clinical experiences (Anderson and Anderson, 2019; Bednarczyk et al., 2014). There exists a significant opportunity to augment this traditional learning process with AI tools that can efficiently retrieve and synthesize educational content from clinical cases, drawing upon vast archives of medical knowledge. LLMs present a promising avenue for this augmentation, but their application in high-stakes medical domains is fraught with challenges, most notably the risk of factual inaccuracy (or hallucination) and the use of outdated knowledge (Abd-alrazaq et al., 2023; Li et al., 2023; Xie et al., 2023).

To overcome these challenges, we develop MedTutor, a system that grounds LLM generation in verifiable, contextually relevant medical knowledge to case reports through a RAG pipeline. Our primary goal is to provide medical residents with a reliable tool that transforms any given clinical report into a concise, and highly relevant educational module. We focus on radiology as the domain of study, although, our techniques are generalizable to other domains. The system begins by decomposing clinical reports into actionable diagnostic queries and keywords that can be effectively issued to a search index, enabling targeted retrieval of relevant educational material. It then initiates a hybrid retrieval process that simultaneously queries a curated database of medical textbooks, and per-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 319–353
November 4-9, 2025



Original clinical report

FINDINGS:
LIVER: The liver is coarsened and nodular in echotexture. The contour of the liver is nodular, consistent with cirrhosis. There is no focal liver mass. The main portal vein is patent with hepatopetal flow. (The rest omitted)

IMPRESSION:
Cirrhotic liver with increased nodularity of the liver compared to the study from (anonymized)

Textbook summary

Cirrhosis is characterized by a hallmark histological pattern of fibrotic bridges carving the liver into regenerative nodules, with key radiologic findings including surface nodularity, widening of fissures, and expansion of the gallbladder fossa. Imaging features include micronodules or macronodules, caudate lobe hypertrophy, and signs of portal hypertension such as splenomegaly, ascites, and varices, with diagnostic considerations varying based on etiology, such as ethanol, hemochromatosis, or chronic viral hepatitis...(The rest omitted)

Educational material

1. Definition and Pathophysiology: Cirrhosis is a late stage of scarring of the liver caused by many forms of liver diseases and condition...
2. Imaging Features: Ultrasound: The liver appears coarsened and nodular in echotexture with a nodular contour. ...
3. Diagnostic Considerations: ...
4. Complications: ...
5. Surveillance and Monitoring: ...
6. Cardiac Implications: ...
7. Imaging Modalities for Surveillance: ...

Multiple choice Q&A

Q1. What is a characteristic ultrasound finding in a liver affected by cirrhosis?

A. Smooth liver surface
B. Coarsened and nodular liver echotexture
C. Focal liver mass
D. Absence of portal vein flow
Answer: B
Explanation: The ultrasound findings in the provided context describe the liver as "coarsened and nodular in echotexture" and the contour of the liver as ...

Figure 1: The overall architecture of the MedTutor system.

forms live searches on academic search engines (i.e., PubMed and Semantic Scholar) for current published literature related to the case.

The retrieved evidence undergoes a multi-faceted processing step: academic articles and textbook snippets are reranked for relevance to the case, using a state-of-the-art reranking model, Qwen3-Reranker-8B (Zhang et al., 2025b). Finally, all processed evidence—the original report, keywords, top-ranked articles, and textbook summaries—is synthesized by a generator LLM into two distinct outputs: a comprehensive set of educational material and a set of multiple-choice questions (MCQs) designed to test understanding. The overview of the system is illustrated in Figure 1.

This work makes three primary contributions:

• We detail the design and implementation of the MedTutor system, a scalable and efficient architecture that leverages asynchronous I/O, parallel multi-GPU inference with vLLM (Kwon et al., 2023), and optimized batch processing to handle large workloads.

• We introduce a new, expert annotated benchmark dataset for evaluating the quality of AI-generated educational content. We run our pipeline with 6 LLMs(see Appendix A for details) across 2,000 clinical reports per each 5 major radiology datasets (*i.e.,* Yale Hospital Internal, MIMIC-CXR (Johnson et al., 2019, 2024), MIMIC-IV-note (Johnson et al., 2023), CheXpert Plus (Chambon et al., 2024), and ReXGradient-160K (Zhang et al., 2025a)).

• We collected comprehensive evaluations from three radiologists, alongside a LLM-as-a-Judge evaluation with four models for all experiments. This dataset, which we are planning to publicly release, will be a valuable resource for evaluating the quality and clinical utility of generative models in medicine.

Our analysis provides insights about the usefulness of our system to users and highlight the strengths and weaknesses of LLMs in evaluating educational content in our setting.

2 Related Work

Our work is situated at the intersection of RAG, the application of LLMs in medicine, and the critical need for trustworthy medical AI systems. We structure our review accordingly.

2.1 LLMs and RAG in the Medical Domain

The application of LLMs in medicine has shown immense promise. General-purpose foundation models have demonstrated impressive capabilities on standardized medical exams and complex diagnostic problems (Nori et al., 2023; Singhal et al., 2023). This has fueled a broader vision for generalist biomedical AI that can assist with a wide range of clinical tasks (Tu et al., 2023). However, the "black-box" nature of these models and their potential for factual errors or "hallucinations" remain

significant barriers to clinical adoption, necessitating robust evaluation frameworks (Huang et al., 2024; Li et al., 2023; Xie et al., 2023).

To mitigate these risks, RAG has emerged as a key paradigm for building dependable clinical tools (Lewis et al., 2021). By grounding LLM outputs in external, verifiable evidence from reliable medical literature, RAG provides a pathway to trustworthy AI. A recent perspective in *Nature Medicine* strongly advocates for RAG as a prerequisite for the responsible deployment of generative AI in healthcare (Yang et al., 2024). The field is now maturing to a point where standardized benchmarks for medical RAG are being established, allowing for more rigorous evaluation of these systems (Xiong et al., 2024a). Our work contributes to this growing body of literature by presenting a novel RAG-based system specifically designed for medical education, a domain where accuracy and reliability are paramount.

2.2 LLMs for Medical Education

LLMs show promise in generating medical exam content, though concerns about accuracy necessitate expert oversight (Zhu et al., 2024). Integrating RAG improves reliability by grounding output in external sources, with studies reporting notable gains in question-answering accuracy using medical textbooks (Chen et al., 2025; Wang et al., 2023). Benchmarks like MIRAGE further validate RAG's role in medical QA tasks (Xiong et al., 2024b).

For resident training, LLMs can assess skills and provide feedback, but expert review remains vital (Atsukawa et al., 2024). Systems enabling citation generation enhance factuality (Wang et al., 2025), while evaluation frameworks like LLM-as-a-Judge offer scalability despite only moderate alignment with human judgment (Zheng et al., 2025). New approaches continue to embed evidence-based medicine principles into RAG pipelines for clinically accurate educational content (Lu et al., 2025).

Our system, MedTutor, is distinct in its focus on transforming a single clinical report into a comprehensive educational module, featuring synthesized educational material and MCQs grounded in a hybrid retrieval from both medical textbooks and the latest academic literature. This approach is designed not to replace expert judgment but to augment it, fostering the self-directed learning skills that are crucial for lifelong professional development (Bravata et al., 2003; Williams and Ntiri, 2018).

3 MedTutor

MedTutor is a RAG system designed to support medical residents on case-based education. It involves a pipeline approach in retrieving highly relevant educational content from both textbooks and literature and produces a coherent educational material as well as multiple-choice questions related to a case. While MedTutor's design is general and can be applicable to many clinical practices, we focus our domain on radiology due to availability of public datasets and our access to domain experts.

3.1 The MedTutor Pipeline Stages

The input to MedTutor is a case report, which will be processed through a sequence of automated stages, each designed for parallel execution.

Case decomposition into search queries: The process begins with a source radiology report. Then we use an LLM (`Llama-3.3-70B-Instruct`) to process the radiology report and decompose it into multiple keyword based queries that will be used for retrieval. These queries are key diagnostic terms and findings. Prompts for case decomposition into search queries are shown in Appendix D.1.

Hybrid Evidence Retrieval: For each search query, the system performs a hybrid retrieval process in parallel described below: *(1) Local retrieval for textbook material:* Textbooks and notes are essential resources for medical education. In our MedTutor system, we first apply OCR to a radiology textbook (Dahnert and Ovid Technologies, 2017) using the `mistral-ocr-2503` model, then segment and index the material by page. We generate dense embeddings for these materials with the `Qwen3-Embedding-8B` model, which has demonstrated state-of-the-art performance in embedding and retrieval tasks on the MTEB benchmark (Muennighoff et al., 2022) among models of comparable size. These embeddings are stored in a precomputed vector database for subsequent queries. For local database search, we employ a bi-encoder architecture to generate dense vector representations for both the query and the pre-indexed textbook pages, subsequently identifying the most relevant page using cosine similarity. *(2) Retrieval using academic APIs:* Some case reports are more specialized or rare, requiring retrieving knowledge from latest academic literature. Therefore, we also employ retrieval from academic search engines. We use PubMed and Semantic Scholar APIs, two commonly used and freely available scholarly sys-

tems, to fetch the latest relevant research papers. To prevent rate-limiting, API calls are managed by an `asyncio.Semaphore`. If pre-fetched results for the queries are available, this step is skipped to improve efficiency.

Evidence Processing: The retrieved evidence is then processed through two concurrent tasks: *(1) Reranking:* As the search engine results using keyword queries can be noisy, we employ a reranking stage to prioritize the most relevant (top 2) documents to the case report. This is handled by a dedicated service running the `Qwen3-Reranker-8B` model, a strong reranker according to the MTEB benchmark. The reranker is given a contextualized query containing both the original report's text and the specific search keyword to improve relevance. *(2) Query-focused Summarization:* Concurrently, the content retrieved from the local textbook database is summarized with respect to the query by a generator LLM to distill key information related to the keywords into a concise way.

Generating Learning Modules: Finally, the original case report, the top retrieved content including the textbook snippets and abstracts of related papers, and the search keywords are passed to a generator LLM to generate a concise learning module. These learning modules contain comprehensive explanatory material contextualizing the case within broader medical knowledge, followed by multiple-choice questions designed to test understanding of key concepts. Prompts used for generating learning modules are in Appendix D.

Optimized Multi-Task Generation: The generation step is heavily optimized for efficiency. Instead of generating outputs sequentially, the system first constructs prompts for all cases received, and all sub-tasks.

Batch Construction: Two distinct batches of input prompts to LLMs are created: one for generating the final educational modules and another for generating multiple-choice questions. These input prompts are long-context (3530 tokens for MCQ, 3463 tokens for Educational module in average), containing the original report, the list of keywords, the abstracts of the top-ranked papers after reranking, and the generated textbook summaries. *Concurrent Batch Inference:* The two batches are sent concurrently to the generation service. The `generate_text_batch` method in our `VLLMHandler` passes the entire list of prompts to the vLLM engine in a single call. This fully leverages vLLM's continuous batching capability, allowing the GPU to process multiple requests simultaneously without padding, dramatically increasing throughput and reducing overall processing time. This architecture, particularly the use of batch generation with vLLM, allows MedTutor to process hundreds of complex reports far more efficiently than a naive, sequential approach, making it a practical tool for large-scale educational content creation.

Local Deployment: We deploy MedTutor completely locally using locally served open-source LLMs, without reliance on any cloud-based LLM APIs. This allows responsible and private handling of medical data.

3.2 System Design Details

The MedTutor pipeline is an asynchronous, multi-stage system designed for efficiency, scalability, and modularity. The architecture leverages parallel processing across multiple GPUs and optimized batching to handle large-scale report generation. The entire workflow is orchestrated by a central `asyncio` event loop, which communicates with dedicated `ModelWorker` processes via multiprocessing queues. A conceptual overview of the architecture is shown in Figure 1.

3.3 Architecture for Scalability

At the core of our system is a hybrid concurrency model designed to maximize throughput and resource utilization.

Asynchronous Orchestration: The main process runs on an `asyncio` event loop, managing I/O-bound tasks such as live API calls for literature retrieval and orchestrating the overall pipeline. This allows the system to handle thousands of concurrent operations efficiently without being blocked by network latency.

Parallel Multi-GPU Inference: To handle the computationally intensive model inference, we spawn separate `ModelWorker` processes for each required service (e.g., reranking, generation). Each worker is pinned to a specific GPU or set of GPUs as defined in the `configs.json` file. Within each worker, we use the vLLM engine, a state-of-the-art serving library that employs techniques like PagedAttention to achieve high-throughput, low-latency inference.

Inter-Process Communication: The main `asyncio` loop communicates with the `ModelWorker` processes using a robust queue-based system (`multiprocessing.Queue`). A

request, tagged with a unique ID, is placed on a request queue. The main loop then awaits an `asyncio.Future` associated with that ID. The worker process retrieves the request, performs the inference, and places the result on a response queue. A dedicated listener task in the main loop listens for responses and resolves the corresponding `Future`, seamlessly bridging the asynchronous and multi-process components.

4 System Evaluation

We conduct a multi-faceted evaluation to assess the quality of our MedTutor system. Given our focus on the radiology domain, the evaluation is done by three radiologists who scored the outputs on a 5-point Likert scale (1=Poor, 5=Excellent). Annotation guidelines and the annotation interface design are detailed in Appendices F and G. We evaluate both the intermediate "upstream" components of our pipeline and the final "downstream" generated content. Furthermore, we investigate the feasibility of using an LLM-as-a-Judge as a proxy for human evaluation of the AI generated educational content by analyzing its agreement with our human experts.

4.1 Upstream Component Quality

First, we evaluate the quality of the upstream components that feed into the final generator: search query extraction and retrieved paper relevance. This evaluation was conducted on a set of 50 clinical cases. As shown in Table 1, human experts found the search queries extracted by the system to be highly appropriate (Human Avg. score of 3.73). However, they were more critical of the relevance of the retrieved academic papers, giving an average score of 2.88. This suggests that while the system correctly identifies the main topics, the unfiltered, live-retrieved literature can contain articles that are not perfectly aligned with the specific clinical context of the report. In contrast, the LLM judges rated the paper relevance significantly higher (LLM Avg. 4.20), indicating a divergence in the assessment of contextual relevance between human experts and automated metrics.

4.2 Downstream Generation Quality

The primary evaluation focused on the quality of the final, user-facing outputs: textbook summaries, MCQs, and educational material. Three radiologists annotated the outputs from two generator models, Llama-3.3-70B-Instruct and MedGemma-27B. The detailed results are presented in Table 2.

Evaluator	Query Appropriateness	Paper Relevance
Human Evaluators	**3.73**	**2.88**
MedGemma-27B	3.73	4.34
GPT-4.1-mini	4.15	4.52
Gemini-2.5-Flash	4.27	4.58
Gemini-2.5-Pro	4.03	3.37
LLM Avg.	**4.05**	**4.20**

Table 1: Comparison of evaluator scores. The " Evaluators" row represents the combined results from two independent radiologists (n=50 each).

Both models produced high-quality outputs according to our expert evaluators. Llama 3.3-70B-Instruct achieved a respectable average human score of 3.44, demonstrating its capability in synthesizing complex medical information into educational content. MedGemma-27B, a model more specialized for the medical domain, performed slightly better, with an average human score of 3.65. The experts particularly noted the higher quality of the MCQs generated by MedGemma-27B (3.53) compared to those from Llama-3.3-70B-Instruct (3.11). This suggests that the domain-specific nature of MedGemma-27B provides a distinct advantage in generating educational content, such as plausible distractors for multiple-choice questions.

When comparing human evaluations to the LLM-as-a-Judge scores, we note an interesting trend. The LLM judges also preferred MedGemma-27B over Llama 3.3, aligning with the relative ranking of the human experts. However, the LLMs consistently assigned higher absolute scores than the human radiologists. This suggests that while LLM-as-a-Judge can be a valuable tool for scalable, relative comparisons between models, its scoring calibration differs from that of human experts, indicating a tendency for score inflation. These findings suggest a promising path toward semi-automated evaluation while reinforcing the role of human experts as the gold standard for assessing clinical utility. Full LLM-as-a-Judge results are in Tab A.

4.3 Inter-Annotator Agreement

To ensure the reliability of our human evaluations, we measured the inter-annotator agreement (IAA) between the two board-certified radiologists using Krippendorff's Alpha (Krippendorff, 2011).

Model	Evaluator	Textbook Summary	Educational Material	MCQ Quality	Overall Average
Llama 3.3-70B-Instruct	**Human Evaluators**	**3.43**	**3.78**	**3.11**	**3.44**
	MedGemma-27B	3.64	3.66	3.79	3.70
	GPT-4.1-mini	4.34	4.50	4.19	4.34
	Gemini-2.5-Flash	2.82	3.58	4.08	3.49
	Gemini-2.5-Pro	3.95	4.28	4.14	4.12
	LLM Avg.	**3.69**	**4.01**	**4.05**	**3.91**
MedGemma-27B	**Human Evaluators**	**3.58**	**3.84**	**3.53**	**3.65**
	MedGemma-27B	3.65	4.09	4.22	3.99
	GPT-4.1-mini	4.21	4.79	4.60	4.53
	Gemini-2.5-Flash	3.05	4.61	4.47	4.04
	Gemini-2.5-Pro	3.84	4.18	4.15	4.06
	LLM Avg.	**3.69**	**4.42**	**4.36**	**4.16**

Table 2: Main Generation Task Quality: Direct Comparison of Human Expert and LLM-as-a-Judge Evaluations. The "Human Evaluators" scores represent the combined results from three independent radiologists (n=50 each). All scores are on a 1-5 scale (5=best).

The alpha coefficient is calculated as: $\alpha = 1 - \frac{D_o}{D_e}$ Here, D_o is the observed disagreement, calculated as the average difference between the ratings from each human annotator, A_1 and A_2, across all M evaluated items. Specifically, if $r_{i,1}$ and $r_{i,2}$ are the ratings for item i from A_1 and A_2 respectively, then:

$$D_o = \frac{1}{M} \sum_{i=1}^{M} \delta^2(r_{i,1}, r_{i,2})$$

D_e represents the disagreement expected by chance, calculated based on the individual rating distributions of A_1 and A_2. For the difference function δ^2, we first recoded the 1-to-5 Likert scale ratings into a 3-point interval scale (1-2 $\rightarrow$ 1; 3 $\rightarrow$ 2; 4-5 $\rightarrow$ 3) and then applied a squared difference: $\delta^2(u, v) = (u - v)^2$.

The results, presented in Table 4, show a range of agreement levels. We observed good agreement for the *Textbook Summary* from MedGemma-27B ($\alpha = 0.661$) and fair agreement for *Paper Relevance* ($\alpha = 0.493$).

Overall, our annotators demonstrated moderate to good agreement across most tasks (with the exception of MCQ quality), which is in line with agreement levels reported in prior work on high-quality datasets (Liu et al., 2024; Bavaresco et al., 2025). The lower agreement for MCQ evaluation ($\alpha = 0.048$) suggests that the criteria for this specific task may require more detailed guidelines to improve consistency.

5 Conclusion

In this work, we introduce **MedTutor**, a novel, open-source system designed to augment clinical education by transforming clinical reports into structured, evidence-backed learning modules. Our system addresses the critical challenges of factual accuracy and knowledge freshness in medical AI by employing a sophisticated RAG pipeline. This pipeline features a hybrid retrieval mechanism that synthesizes knowledge from both foundational medical textbooks and real-time academic literature, ensuring the generated educational modules are both reliable and current.

Our rigorous evaluation, conducted by board-certified radiologists, confirmed that MedTutor can produce high-quality, clinically valuable educational content. Furthermore, our large-scale LLM-as-a-Judge analysis revealed a moderate but promising correlation with human expert judgments, suggesting a viable path toward scalable automated evaluation while underscoring the continued importance of expert oversight.

By publicly releasing the MedTutor system, its user interface, and the comprehensive evaluation dataset, we make two key contributions. First, we provide a practical tool that can be immediately adapted by other institutions to enhance their own training programs. Second, we offer a valuable benchmark and framework for future research into building trustworthy and effective generative AI systems for the high-stakes medical domain. We believe this work represents a significant step toward fostering more effective and efficient clinician-AI

collaboration in medical education.

6 Limitations

While MedTutor demonstrates a promising approach to augmenting medical education, we acknowledge several limitations that offer avenues for future work.

First, our evaluation is primarily focused on the domain of radiology. Although the system's architecture is designed to be generalizable, its effectiveness and the nuances of its application in other medical specialties with different reporting styles and knowledge structures, such as pathology or cardiology, have not yet been explored. Future studies should assess the adaptability and performance of MedTutor across a broader range of clinical domains.

Second, the human evaluation, while rigorous and conducted by domain experts, was performed on a dataset of 50 clinical cases. A larger-scale study involving a greater number of cases and a more diverse cohort of radiologists would be beneficial to further validate our findings and provide more robust statistical power to the conclusions drawn.

Finally, our analysis of inter-annotator agreement and the LLM-as-a-Judge evaluations highlights challenges in consistently generating high-quality subjective content. The lower agreement scores for MCQs, for instance, suggest that these outputs require further refinement. This indicates that more advanced prompting techniques, fine-tuning of the generator models, or more sophisticated evaluation guidelines may be necessary to improve the reliability and educational value of these more complex, creative tasks.

Acknowledgments

This research was supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number: HI19C1352).

References

Alaa A. Abd-alrazaq, Rawan AlSaad, Dari Alhuwail, Arfan Ahmed, P. Healy, Syed Latifi, S. Aziz, R. Damseh, Sadam Alabed Alrazak, and Javaid Sheikh. 2023. Large language models in medical education: Opportunities, challenges, and future directions. *JMIR Medical Education*, 9.

Michael Anderson and S. Anderson. 2019. How should ai be developed, validated, and implemented in patient care? *AMA journal of ethics*, 21 2:E125–130.

Natsuko Atsukawa, Hiroyuki Tatekawa, Tatsushi Oura, Shunichi Matsushita, Daisuke Horiuchi, H. Takita, Yasuhito Mitsuyama, Ayako Omori, T. Shimono, Yukio Miki, and D. Ueda. 2024. Evaluation of radiology residents' reporting skills using large language models: An observational study. *arXiv preprint arXiv:2404.56789*.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, André F. T. Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K Surikuchi, Ece Takmaz, and Alberto Testoni. 2025. Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks. *Preprint*, arXiv:2406.18403.

J. Bednarczyk, M. Pauls, Jason A. Fridfinnson, and E. Weldon. 2014. Characteristics of evidence-based medicine training in royal college of physicians and surgeons of canada emergency medicine residencies - a national survey of program directors. *BMC Medical Education*, 14:57 – 57.

Dawn MT Bravata, Stephen J Huot, Hadley S Abernathy, K. Skeff, and D. Bravata. 2003. The development and implementation of a curriculum to improve clinicians' self-directed learning skills: a pilot project. *BMC Medical Education*, 3:7 – 7.

Pierre Chambon, Jean-Benoit Delbrouck, Thomas Sounack, Shih-Cheng Huang, Zhihong Chen, Maya Varma, Steven QH Truong, Chu The Chuong, and Curtis P. Langlotz. 2024. Chexpert plus: Augmenting a large chest x-ray dataset with text radiology reports, patient demographics and additional image formats. *Preprint*, arXiv:2405.19538.

Rong Chen, Siyun Zhang, Yiyi Zheng, Qiuhua Yu, and Chu-huai Wang. 2025. Enhancing treatment decision-making for low back pain: A novel framework integrating large language models with retrieval-augmented generation technology. *arXiv preprint arXiv:2501.34567*.

Wolfgang. Dahnert and Inc. Ovid Technologies. 2017. *Radiology review manual*, 8th ed. edition. Wolters Kluwer, Philadelphia.

D. Daniel, Sue E. Poynter, C. Landrigan, C. Czeisler, J. Burns, and T. Wolbrink. 2020. Pediatric resident engagement with an online critical care curriculum during the intensive care rotation*. *Pediatric Critical Care Medicine*, 21:986 – 991.

Yining Huang, Keke Tang, and Meilian Chen. 2024. A comprehensive survey on evaluating large language model applications in the medical industry. *arXiv preprint arXiv:2401.12345*.

A. Johnson, T. Pollard, R. Mark, S. Berkowitz, and S. Horng. 2024. Mimic-cxr database (version 2.1.0).

Alistair Johnson, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. 2023. Mimic-iv-note: Deidentified free-text clinical notes (version 2.2).

Alistair E W Johnson, Tom J Pollard, Seth J Berkowitz, and 1 others. 2019. Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific data*, 6(1):317.

Klaus Krippendorff. 2011. Computing Krippendorff's Alpha-Reliability. Technical report, Annenberg School for Communication, University of Pennsylvania.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Dongfang Li, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Ziyang Chen, Baotian Hu, Aiguo Wu, and Min Zhang. 2023. A survey of large language models attribution. *ArXiv*, abs/2311.03731.

Yixin Liu, Alexander R. Fabbri, Jiawen Chen, Yilun Zhao, Simeng Han, Shafiq Joty, Pengfei Liu, Dragomir Radev, Chien-Sheng Wu, and Arman Cohan. 2024. Benchmarking generation and evaluation capabilities of large language models for instruction controllable summarization. *Preprint*, arXiv:2311.09184.

Keer Lu, Zheng Liang, Da Pan, Shusen Zhang, Xin Wu, Weipeng Chen, Zenan Zhou, Guosheng Dong, Bin Cui, and Wentao Zhang. 2025. Med-r^2: Crafting trustworthy llm physicians via retrieval and reasoning of evidence-based medicine. *arXiv preprint arXiv:2505.89012*.

Meta. 2025. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Accessed: 2025-05-20.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. 2023. Capabilities of gpt-4 on medical challenge problems. *Preprint*, arXiv:2303.13375.

M. Rogers, Michelle Zeidan, Zac Flinders, A. Presson, and R. Burks. 2019. Educational resource utilization by current orthopaedic surgical residents: A nationwide survey. *JAAOS Global Research & Reviews*, 3.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, and 12 others. 2023. Towards expert-level medical question answering with large language models. *Preprint*, arXiv:2305.09617.

Tao Tu, Shekoofeh Azizi, Danny Driess, Mike Schaekermann, Mohamed Amin, Pi-Chuan Chang, Andrew Carroll, Chuck Lau, Ryutaro Tanno, Ira Ktena, Basil Mustafa, Aakanksha Chowdhery, Yun Liu, Simon Kornblith, David Fleet, Philip Mansfield, Sushant Prakash, Renee Wong, Sunny Virmani, and 13 others. 2023. Towards generalist biomedical ai. *Preprint*, arXiv:2307.14334.

Xiao Wang, Mengjue Tan, Qiao Jin, Guangzhi Xiong, Yu Hu, Aidong Zhang, Zhiyong Lu, and Minjia Zhang. 2025. Medcite: Can language models generate verifiable text for medicine? *arXiv preprint arXiv:2503.67890*.

Yubo Wang, Xueguang Ma, and Wenhu Chen. 2023. Augmenting black-box llms with medical textbooks for biomedical question answering. *arXiv preprint arXiv:2308.12345*.

Adrienne A. Williams and Shana O. Ntiri. 2018. An online, self-directed curriculum of core research concepts and skills. *MedEdPORTAL : the Journal of Teaching and Learning Resources*, 14.

Qianqian Xie, E. Schenck, He S. Yang, Yong Chen, Yifan Peng, and Fei Wang. 2023. Faithful ai in medicine: A systematic review with large language models and beyond. *medRxiv*.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024a. Benchmarking retrieval-augmented generation for medicine. *Preprint*, arXiv:2402.13178.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024b. Benchmarking retrieval-augmented generation for medicine. *arXiv preprint arXiv:2403.45678*.

Rui Yang, Yilin Ning, Emilia Keppo, Mingxuan Liu, Chuan Hong, Danielle S Bitterman, Jasmine Chiat Ling Ong, Daniel Shu Wei Ting, and Nan Liu. 2024. Retrieval-augmented generation for generative artificial intelligence in medicine. *Preprint*, arXiv:2406.12449.

Xiaoman Zhang, Julián N. Acosta, Josh Miller, Ouwen Huang, and Pranav Rajpurkar. 2025a. Rexgradient-160k: A large-scale publicly available dataset of chest radiographs with free-text reports. *Preprint*, arXiv:2505.00228.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Weibing Zheng, Laurah Turner, Jess Kropczynski, Murat Ozer, Tri Nguyen, and S. Halse. 2025. Llm-as-a-fuzzy-judge: Fine-tuning large language models as a clinical evaluation judge with fuzzy logic. *arXiv preprint arXiv:2504.78901*.

Yunqi Zhu, Wen Tang, Ying Sun, and Xuebing Yang. 2024. The potential of llms in medical education: Generating questions and answers for qualification exams. *arXiv preprint arXiv:2402.23456*.

Appendix Contents

A LLM-as-a-Judge Evaluation

Model	Judge	Textbook Summary	Educational Material	MCQ	Average
Llama 3.1-8B-Instruct	MedGemma-27B	3.64 (±0.95)	3.49 (±1.10)	3.69 (±1.05)	3.61
	GPT-4.1-mini	4.06 (±0.85)	4.18 (±0.90)	3.86 (±0.92)	4.03
	Gemini-2.5-Pro	3.59 (±1.15)	3.55 (±1.20)	3.92 (±1.18)	3.69
	Gemini-2.5-Flash	3.64 (±1.30)	3.49 (±1.25)	3.88 (±1.28)	3.67
	Avg. (Judges)	**3.73**	**3.68**	**3.84**	**3.75**
Qwen3-8B	MedGemma-27B	3.39 (±1.01)	4.01 (±0.95)	3.78 (±0.88)	3.73
	GPT-4.1-mini	3.42 (±0.90)	4.49 (±0.75)	4.22 (±0.81)	4.04
	Gemini-2.5-Pro	3.45 (±1.10)	4.11 (±0.99)	3.81 (±0.95)	3.79
	Gemini-2.5-Flash	3.39 (±1.25)	4.01 (±1.15)	3.75 (±1.05)	3.72
	Avg. (Judges)	**3.41**	**4.16**	**3.89**	**3.82**
Llama-4-Scout-17B-16E-Instruct	MedGemma-27B	3.68 (±0.88)	4.08 (±0.85)	3.85 (±0.80)	3.87
	GPT-4.1-mini	4.30 (±0.70)	4.28 (±0.65)	4.18 (±0.72)	4.25
	Gemini-2.5-Pro	3.71 (±0.95)	4.15 (±0.90)	4.01 (±0.88)	3.96
	Gemini-2.5-Flash	3.68 (±1.10)	4.08 (±1.05)	3.95 (±1.00)	3.90
	Avg. (Judges)	**3.84**	**4.15**	**4.00**	**4.00**
Qwen3-32B	MedGemma-27B	3.55 (±0.75)	4.64 (±0.60)	3.99 (±0.65)	4.06
	GPT-4.1-mini	3.99 (±0.65)	4.19 (±0.50)	4.48 (±0.55)	4.22
	Gemini-2.5-Pro	3.61 (±0.80)	4.70 (±0.70)	4.25 (±0.75)	4.19
	Gemini-2.5-Flash	3.55 (±1.20)	4.64 (±1.10)	4.18 (±1.00)	4.12
	Avg. (Judges)	**3.68**	**4.54**	**4.23**	**4.15**
Llama-3.3-70B-Instruct	MedGemma-27B	3.64 (±0.68)	3.66 (±0.61)	3.79 (±0.55)	3.70
	GPT-4.1-mini	4.34 (±0.72)	4.50 (±0.55)	4.19 (±0.60)	4.34
	Gemini-2.5-Pro	3.95 (±1.23)	4.28 (±0.38)	4.14 (±0.55)	4.12
	Gemini-2.5-Flash	2.82 (±1.45)	3.58 (±1.44)	4.08 (±1.46)	3.49
	Avg. (Judges)	**3.69**	**4.01**	**4.05**	**3.91**
MedGemma-27B	MedGemma-27B	3.65 (±0.88)	4.09 (±0.61)	4.22 (±0.60)	3.99
	GPT-4.1-mini	4.21 (±0.81)	4.79 (±0.48)	4.60 (±0.51)	4.53
	Gemini-2.5-Pro	3.84 (±1.18)	4.18 (±0.45)	4.15 (±0.72)	4.06
	Gemini-2.5-Flash	3.05 (±1.58)	4.61 (±0.90)	4.47 (±1.18)	4.04
	Avg. (Judges)	**3.69**	**4.42**	**4.36**	**4.16**

Table 3: Aggregated LLM-as-a-Judge evaluation results across all datasets, comparing different judges. The **Avg. (Judges)** row indicates the mean of scores across the judges. All scores are on a 1-5 scale (5=best). Llama-4-Scout-17B-16E-Instruct(Meta, 2025) was inferenced in FP8.

B Detailed Inter-Annotator Agreement

Model Context	Evaluation Metric	Krippendorff's Alpha (α)	Pairwise Kappa (κ)	% Exact Agreement	% Within $\pm$1 Point	Correlation (r)
Upstream	Keyword Appropriateness	0.335	0.627	41%	80%	0.709
	Paper Relevance	0.474	0.675	59%	95%	0.778
Llama3-70B	Textbook Summary	0.347	0.555	48%	84%	0.587
	Educational Material	-0.228	0.382	50%	94%	0.325
	MCQ	-0.159	0.222	29%	81%	0.375
MedGemma-2B	Textbook Summary	0.627	0.812	66%	96%	0.721
	Educational Material	0.354	0.673	72%	100%	0.589
	MCQ	0.114	0.629	46%	90%	0.596

Table 4: Detailed Inter-Annotator Agreement (IAA) between three radiologists across different evaluation tasks. Krippendorff's Alpha (α) and Avg. Pairwise Kappa (κ) measure reliability, while agreement percentages and Pearson correlation (r) provide further insight into rater consistency.

Overall, the agreement scores suggest that MedGemma-27B's outputs were evaluated more consistently by the radiologists than those from Llama3.3-70B. As shown in Table 4, MedGemma-27B's Textbook Summary achieved the highest reliability, with a Krippendorff's Alpha of 0.627, approaching the threshold for acceptable agreement, and a substantial average pairwise Kappa of 0.812. The upstream task of Paper Relevance also demonstrated moderate to substantial agreement across most measures.

Conversely, the outputs from Llama3.3-70B, particularly for more subjective tasks like Educational Material and MCQ evaluation, yielded negative Alpha values, indicating systematic disagreement among the raters (Figure 3). The evaluation of MCQs proved challenging for both models, though agreement was notably higher for MedGemma-27B (Figure 4). These findings highlight that while structured summarization tasks can achieve high inter-rater reliability, evaluating more complex, subjective-generative tasks may require more detailed guidelines to ensure rater consistency.

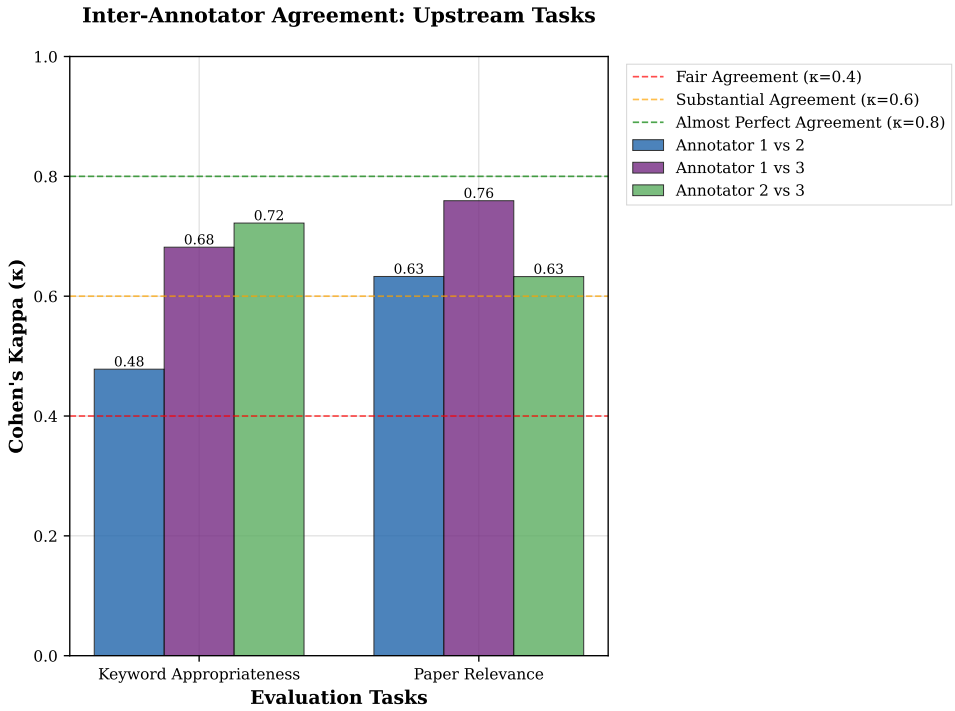


Figure 2: Pairwise Cohen's Kappa (κ) scores for Upstream Tasks. This figure shows the agreement between three pairs of annotators for keyword appropriateness and paper relevance.

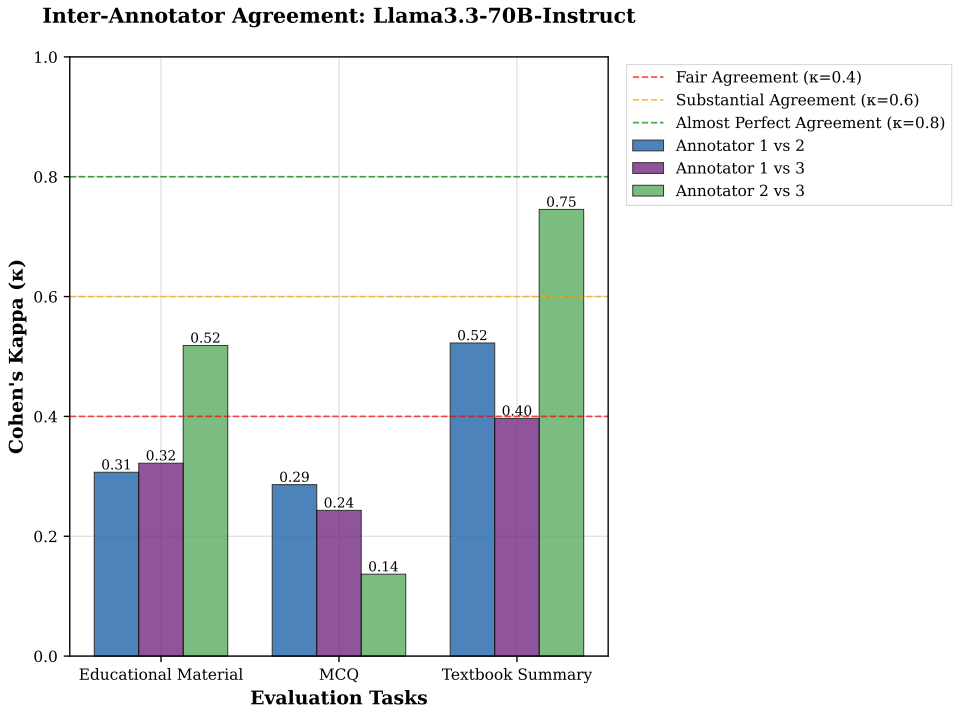


Figure 3: Pairwise Cohen's Kappa (κ) scores for Llama3.3-70B-Instruct Generated Content.

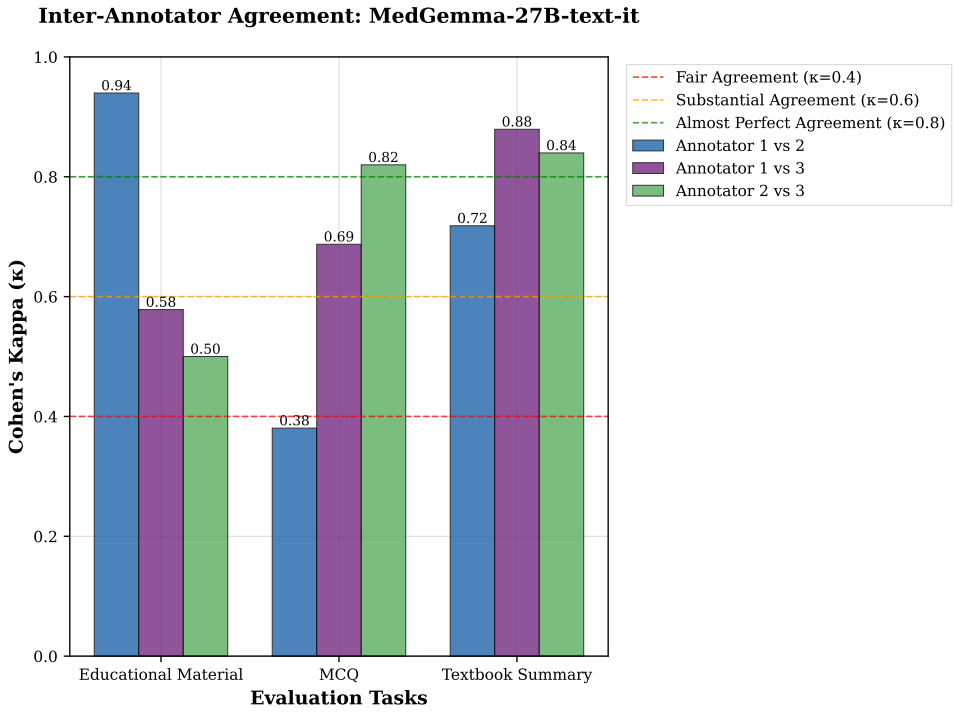


Figure 4: Pairwise Cohen's Kappa (κ) scores for MedGemma-27B-text-it Generated Content.

C MedTutor Dataset Samples and Public Dataset Information

This appendix provides the best inter-annotator agreement examples of the highest and lowest-scoring cases from the 50 cases evaluated by three expert radiologists. These cases were sampled (10 from each of the 5 datasets; Yale Internal, MIMIC-IV-note, MIMIC-CXR, CheXpert-Plus, ReXGradient-160K) and generated by two different models (`Llama-3.3-70B-Instruct` and `MedGemma-27B-text-it`). Each example includes the original clinical report and its corresponding generated report from MedTutor.

Also, we publicly release a large-scale dataset(total 144K) generated by our system. This includes reports from **CheXpert-Plus**, **MIMIC-IV-note**, and **MIMIC-CXR** (2,000 reports each), processed by six different generator models and 4 evaluator models. Due to licensing and de-identification challenges, the Yale-internal and ReXGradient datasets are not included in the public release.

C.1 Highest-scoring case generated by `Llama-3.3-70B-Instruct`.

C.2 Lowest-scoring case generated by `Llama-3.3-70B-Instruct`.

C.3 Highest-scoring case generated by `MedGemma-27B-text-it`.

C.4 Lowest-scoring case generated by `MedGemma-27B-text-it`.

C.1 Highest-Scoring Case with Llama-3.3-70B-Instruct

Case Information

Dataset: `MIMIC-IV-note`
Generator Model: `Llama-3.3-70B-Instruct`
Case ID: `19287224-RR-6`

Original Radiology Report

INDICATION: NO_PO contrast; History: () with abd pain NO_PO contrast// abd pain r/o appendicitiis
TECHNIQUE: Single phase split bolus contrast: MDCT axial images were acquired through the abdomen and pelvis following intravenous contrast administration with split bolus technique.
Oral contrast was administered.
Coronal and sagittal reformations were performed and reviewed on PACS.
DOSE: Acquisition sequence:
1) Stationary Acquisition 7.5 s, 0.5 cm; CTDIvol = 35.2 mGy (Body) DLP = 17.6 mGy-cm.
2) Spiral Acquisition 7.3 s, 55.8 cm; CTDIvol = 9.8 mGy (Body) DLP = 548.4 mGy-cm.
Total DLP (Body) = 566 mGy-cm.
COMPARISON: None.
FINDINGS:
LOWER CHEST: Visualized lung fields are within normal limits. There is no evidence of pleural or pericardial effusion.
ABDOMEN:
HEPATOBILIARY: The liver demonstrates homogenous attenuation throughout. There is no evidence of focal lesions. There is no evidence of intrahepatic or extrahepatic biliary dilatation. The gallbladder is within normal limits.
PANCREAS: The pancreas has normal attenuation throughout, without evidence of focal lesions or pancreatic ductal dilatation. There is no peripancreatic stranding.
SPLEEN: The spleen shows normal size and attenuation throughout, without evidence of focal lesions.
ADRENALS: The right and left adrenal glands are normal in size and shape.
URINARY: The kidneys are of normal and symmetric size with normal nephrogram. There is no evidence of focal renal lesions or hydronephrosis. There is no perinephric abnormality.
GASTROINTESTINAL: The stomach is unremarkable. Small bowel loops demonstrate normal caliber, wall thickness, and enhancement throughout. There are scattered colonic diverticula. The appendix is fluid-filled and dilated with wall hyperemia. A measures 13 mm with periappendiceal stranding and a small appendicular at the base. There is no periappendiceal abscess.
PELVIS: The urinary bladder and distal ureters are unremarkable. There is a small amount of free fluid in the pelvis.
REPRODUCTIVE ORGANS: The visualized reproductive organs are unremarkable.
LYMPH NODES: There is no retroperitoneal or mesenteric lymphadenopathy. There is no pelvic or inguinal lymphadenopathy.
VASCULAR: There is no abdominal aortic aneurysm. No significant atherosclerotic disease is noted.

BONES: There is no evidence of worrisome osseous lesions or acute fracture.
Moderate multilevel degenerative changes noted of the lumbar spine.
SOFT TISSUES: The abdominal and pelvic wall is within normal limits.
IMPRESSION:
1. Acute appendicitis.
2. Colonic diverticulosis without diverticulitis.
NOTIFICATION: Findings wet read to the electronic dashboard and discussed with ()

Generated Educational Materials

Keywords: Acute appendicitis, Colonic diverticulosis

Retrieved Paper 1: Retrospective case-control study to predict a potential underlying appendiceal tumor in an acute appendicitis context based on a CT-scoring system.
OBJECTIVES: To assess CT signs to discriminate an appendiceal tumor versus a non-tumoral appendix in an acute appendicitis context.
METHODS: A 10-year bicentric retrospective case-control study was performed in adults. Patients with a histopathological appendiceal tumor and appendicitis were paired for age and sex with patients with non-tumorous appendicitis (1/3 ratio, respectively). Two senior radiologists blindly analyzed numerous CT findings... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/33454458/ | Source: PubMed

Retrieved Paper 2: Dual energy CT in acute appendicitis: value of low mono-energy.
OBJECTIVES: To assess the potential role of low monoenergetic images in the evaluation of acute appendicitis.
METHODS: A retrospective study of 42 patients with pathology proven acute appendicitis underwent contrast-enhanced-CT conducted on a single-source-DECT before surgery. Attenuation, SNR, and CNR were calculated on both monoenergetic and conventional images and compared to 24 abdominal CT-scans with normal appendix. Representative... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/33992882/ | Source: PubMed

Retrieved Paper 3: Association between colorectal cancer and colonic diverticulosis: case-control study based on computed tomographic colonography.
BACKGROUND: The association between colorectal cancer (CRC) and diverticulosis is uncertain, and there is little data reported in Asia. This study examined the association of both diseases based on the computed tomographic colonography (CTC).
METHODS: This study was designed as a case-control study. Korean subjects undergoing CTC between April 2002 to April 2008 in Samsung Medical Center were eligible. Patients with histologically proven... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/21516446/ | Source: PubMed

Retrieved Paper 4: REVISED CONCEPTS ON DIVERTICULAR DISEASE OF THE COLON. II. SO-CALLED DIVERTICULITIS: DIVERTICULAR SIGMOIDITIS AND PERISIGMOIDITIS; DIVERTICULAR ABSCESS, FISTULA, AND FRANK PERITONITIS.
No Abstract
URL: https://pubmed.ncbi.nlm.nih.gov/14272097/ | Source: PubMed

Textbook Summary 1
Acute appendicitis is inflammation typically caused by luminal obstruction (fecolith, lymphoid hyperplasia) leading to ischemia. Radiologic findings on plain films are infrequent (<50%) but may include an appendicolith, cecal wall thickening, or signs of perforation like extraluminal gas or loss of fat planes. CT is highly accurate, showing a distended appendix (>7mm), wall thickening, periappendicular fat stranding, and potentially an appendicolith or complications like perforation. Nonvisualization of the appendix on CT in the setting of right lower quadrant pain makes appendicitis highly unlikely.

Textbook Summary 2
Colonic diverticulosis involves acquired herniations of the colonic mucosa and submucosa through the muscularis propria, typically appearing as thin-walled, air-filled outpouchings on imaging. Radiographically, these diverticula are often seen as rounded structures, sometimes with a bubbly appearance, and may show residual contrast from prior studies. Ultrasound can show thin-walled outpouchings with echogenic material and potential shadowing, while CT demonstrates rounded outpouchings containing air or contrast, often associated with circumferential colonic wall thickening due to muscular hypertrophy.

Assessment Question 1
According to the provided textbook summary, which CT finding is highly suggestive of acute appendicitis?

- A. Normal appendix diameter
- B. Appendix diameter > 7mm
- C. Absence of periappendicular fat stranding
- D. Nonvisualization of the appendix

Answer: B
Explanation: The textbook summary states that CT findings suggestive of acute appendicitis include a distended appendix (>7mm).

Assessment Question 2
Based on the provided abstracts, which CT finding is a positive predictive factor for an underlying appendiceal neoplasm in the context of acute appendicitis?

- A. Appendicolith
- B. Fat stranding
- C. Mural calcifications
- D. Appendix diameter < 15mm

Answer: C
Explanation: The abstract from the study on appendiceal tumors identifies mural calcifications as a positive predictive factor (OR = 47, p = 0.0001) for an underlying tumor in patients presenting with acute appendicitis.

Assessment Question 3
What is the characteristic radiographic appearance of colonic diverticula as described in the textbook summary?

- A. Thick-walled outpouchings filled with fluid
- B. Thin-walled outpouchings, often rounded, containing air or contrast
- C. Irregular thickening of the colonic wall
- D. Absence of air within the colon

Answer: B
Explanation: The textbook summary describes colonic diverticula as acquired herniations appearing as thin-walled, air-filled outpouchings on imaging, often rounded and potentially containing contrast.

Assessment Question 4
According to the provided abstract analyzing the association between colorectal cancer and diverticulosis, what was the main conclusion regarding this association?

- A. Colorectal cancer is strongly associated with diverticulosis.
- B. Diverticulosis is more common in patients with right-sided colon cancer.
- C. Colorectal cancer is not associated with diverticulosis.
- D. Computed tomographic colonography is not useful for assessing diverticulosis.

Answer: C
Explanation: The abstract from the case-control study using CT colonography concluded that colorectal cancer is not associated with diverticulosis, as the distribution of diverticulosis was similar between the cancer and control groups.

Final Educational Material
Radiology Feedback
 # Acute Appendicitis
• Imaging Findings: CT is the preferred modality. Key findings include an appendix diameter >7 mm, wall thickening, periappendiceal fat stranding, and potentially an appendicolith.
• Differential Diagnosis: Nonvisualization of the appendix on CT in the setting of right lower quadrant pain makes appendicitis highly unlikely.
• Advanced Techniques: Low monoenergetic CT images can enhance visualization of inflamed bowel wall and mucosal enhancement, potentially improving diagnostic accuracy.
• Clinical Correlation: While appendicitis is often caused by obstruction, CT can help identify potential underlying neoplasms, especially in atypical presentations. Features suggesting neoplasm include mural calcifications, an appendix mass, focal asymmetric wall abnormality, or diameter $\geq$ 15 mm.
 # Colonic Diverticulosis
• Imaging Findings: Diverticula appear as thin-walled, air-filled or contrast-filled outpouchings from the colon, typically rounded. CT demonstrates these outpouchings, often associated with circumferential colonic wall thickening due to muscular hypertrophy.
• Clinical Correlation: Diverticulosis itself is the presence of these outpouchings and is generally asymptomatic. It is not directly associated with an increased risk of colorectal cancer.

C.2 Lowest-Scoring Case with Llama-3.3-70B-Instruct

Case Information

Dataset: RexGradient-160K
Generator Model: Llama-3.3-70B-Instruct
Case ID: pGRDNRZB2HZ56F0ZL_aGRDNZSUP68GUCQN2_s1.2.826.0.1.3680043.8.498.16841148712021794680399240737673267273

Original Radiology Report

Indication: Fall yesterday. Fever this morning.
Findings: 4228 hours. The heart size and mediastinal contours are stable. There is stable mild subsegmental atelectasis at both lung bases. No confluent airspace opacity, pleural effusion or pneumothorax. Fracture of the proximal right humerus again noted.
Impression: Stable mild bibasilar atelectasis. No acute cardiopulmonary process.

Generated Educational Materials

Keywords: atelectasis, fracture

Retrieved Paper 1: Debunking a mythology: Atelectasis is not a cause of postoperative fever.
Most physicians appreciate that practicing medicine is a commitment to continuous learning. However, "learning" can be mistakenly understood as simply the acquisition of facts and new knowledge. But learning also necessitates the constant re-examination and challenging of one's existing body of knowledge, as misinformation persists when one's beliefs are not challenged or questioned in the light of new information. One example is the pervasive... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/39566396/ | Source: PubMed

Retrieved Paper 2: Use of artificial intelligence in triaging of chest radiographs to reduce radiologists' workload.
OBJECTIVES: To evaluate whether deep learning-based detection algorithms (DLD)-based triaging can reduce outpatient chest radiograph interpretation workload while maintaining noninferior sensitivity.
METHODS: This retrospective study included patients who underwent initial chest radiography at the outpatient clinic between June 1 and June 30, 2017. Readers interpreted radiographs with/without a commercially available DLD that detects nine... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/37615766/ | Source: PubMed

Retrieved Paper 3: Assessment of proximal tibial fractures with 3D FRACTURE (fast field echo resembling a CT using restricted echo-spacing) MRI-intra-individual comparison with CT.
OBJECTIVES: To evaluate the feasibility and diagnostic performance of a 3D FRACTURE (fast field echo resembling a CT using restricted echo-spacing) MRI sequence for the detection and classification of proximal tibial fractures compared with CT.
METHODS: We retrospectively included 126 patients (85 male; 39.6±14.5 years) from two centers following acute knee injury. Patients underwent knee MRI at 3T including FRACTURE-MRI. Additional CT was... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/40126605/ | Source: PubMed

Retrieved Paper 4: How I Do It: Evaluating Cardiac Implantable Devices and Noncardiac Mimics on Chest Radiographs.
Cardiac implantable electronic devices (CIEDs), including pacemakers and defibrillators, are increasingly used to manage various cardiac conditions. This article reviews the radiographic appearance, typical components, and placement of CIEDs, including newer technologies like leadless pacemakers and MRI-conditional devices. The article also highlights the imaging findings of common complications such as lead dislodgement, fracture, and... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/40358448/ | Source: PubMed

---

Textbook Summary 1
Atelectasis, or lung collapse, presents radiologically with increased lung density, vessel crowding, and potential fissure/mediastinal displacement. Obstructive atelectasis involves air resorption distal to a blockage (tumor, mucus plug, foreign body), while nonobstructive atelectasis retains some air. Other forms include passive (pleural effusion/pneumothorax), adhesive (surfactant deficiency), cicatrizing (fibrosis), and discoid/rounded atelectasis (often related to pleural inflammation or obstruction). Specific patterns, like the "Luftsichel" sign, can indicate left upper lobe collapse.

Textbook Summary 2
Enteropathy-associated T-cell lymphoma, a type of non-Hodgkin lymphoma, often presents with bowel wall thickening, ulceration, or strictures, particularly in the proximal small bowel. Radiologic findings may include circumferential wall thickening, mesenteric fat infiltration, and nonbulky lymphadenopathy, with a high frequency of FDG uptake on PET scans. Complications like bowel perforation are common, especially in Type II lymphoma, and differentiating it from large B-cell lymphoma or refractory celiac disease is crucial.

Assessment Question 1
According to the provided abstract debunking a mythology, what is the common misconception regarding atelectasis?

- A. Atelectasis is a common cause of postoperative consolidation.
- B. Atelectasis is a common cause of postoperative fever.
- C. Atelectasis is difficult to differentiate from pneumonia on imaging.
- D. Atelectasis is primarily caused by pleural effusion.

Answer: B
Explanation: The abstract explicitly states that the pervasive belief that postoperative atelectasis causes fever is a myth, despite ample evidence to the contrary.

Assessment Question 2
Which of the following is NOT a type of atelectasis mentioned in the textbook summary?

- A. Obstructive atelectasis
- B. Passive atelectasis
- C. Adhesive atelectasis
- D. Vascular atelectasis

Answer: D
Explanation: The textbook summary lists obstructive, nonobstructive, passive, adhesive, cicatrizing, and discoid/rounded atelectasis, but not vascular atelectasis.

Assessment Question 3
According to the abstract comparing FRACTURE-MRI and CT for proximal tibial fractures, what was the sensitivity of FRACTURE-MRI for detecting fractures?

- A. 78.2
- B. 85.7
- C. 91.5
- D. 97.1

Answer: C
Explanation: The abstract states that fractures were detected by FRACTURE-MRI with a sensitivity of 91.5

Assessment Question 4
The abstract on evaluating cardiac implantable devices mentions which potential complication related to device leads?

- A. Dislodgement
- B. Calcification
- C. Consolidation
- D. Cardiomegaly

Answer: A
Explanation: The abstract explicitly lists lead dislodgement, fracture, and perforation as common complications of cardiac implantable electronic devices.

Final Educational Material
Okay, here is the synthesized feedback focusing on atelectasis and fracture, based on the provided information:
 # Atelectasis
• Clinical Context: Atelectasis is commonly seen postoperatively and in the context of fever, but it is not a direct cause of fever itself. This is a persistent misconception.
• Imaging Appearance: Radiographically, atelectasis appears as increased lung density, often with crowding of pulmonary vessels. It can be subsegmental, as seen in this case.
• Differentiation: It's important to differentiate atelectasis from consolidation (infection), which typically presents with air bronchograms and may have a more lobar distribution.
• AI Application: AI-based triaging tools can help identify atelectasis on chest radiographs, potentially reducing radiologist workload while maintaining diagnostic sensitivity.
 # Fracture
• Imaging Modalities: While CT is the standard for fracture evaluation, specialized MRI sequences like FRACTURE-MRI are being investigated as alternatives, potentially reducing radiation exposure and costs, especially when MRI is already indicated.
• MRI Performance: Studies show FRACTURE-MRI has high sensitivity and specificity for detecting and classifying fractures, comparable to CT in certain locations like the proximal tibia.
• Complications: Imaging is crucial for identifying complications of implanted devices, such as lead fracture in pacemakers/defibrillators, which can be visualized on chest radiographs.

C.3 Highest-Scoring Case with MedGemma-27B-text-it

Case Information

Dataset: `CheXpert-Plus`
Generator Model: `MedGemma-27B-text-it`
Case ID: `23803`

Original Radiology Report

Findings: Two views of the chest demonstrate reticular opacities bilaterally, right greater than left, with a basilar predominance, likely secondary to underlying fibrotic lung disease. There is no evidence of pulmonary edema or pleural effusion. There is cardiomegaly. Left anterior chest wall cardiac pacer appears unchanged in position, with two ventricular leads.
Impression:
1. FINDINGS CONSISTENT WITH FIBROTIC LUNG DISEASE AS DESCRIBED ABOVE.
2. NO EVIDENCE OF PULMONARY EDEMA.
3. STABLE CARDIOMEGALY WITH STABLE ARRANGEMENT OF PACER LEADS.

Generated Educational Materials

Keywords: Fibrotic Lung Disease

Retrieved Paper 1: Collaborative radiologic and histopathologic assessment of fibrotic lung disease.
The idiopathic interstitial pneumonias (IIPs) are a seemingly disconnected collection of diseases usually associated with the presence of pulmonary fibrosis. Categorization of the IIPs continues to be problematic despite recent attempts to refine the diagnostic criteria and suggests that rather than separate diseases, these pneumonias represent a spectrum of injury and abnormal repair of the alveolar wall. Although the initiating injury or... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/20501710/ | Source: PubMed

Retrieved Paper 2: From the radiologic pathology archives: organization and fibrosis as a response to lung injury in diffuse alveolar damage, organizing pneumonia, and acute fibrinous and organizing pneumonia.
Organization, characterized by fibroblast proliferation, is a common and nearly universal response to lung injury whether it is focal or diffuse. Despite the vast range of injurious agents, the lung's response to injury is quite limited, with a similar pattern of reaction seen radiologically and histologically regardless of the underlying cause. Although there is a tendency to divide organization into distinct entities, the underlying injury to... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/24224590/ | Source: PubMed

Textbook Summary 1
In fibrotic lung disease, key radiologic findings include a reticulonodular pattern, ground-glass opacities, and honeycombing, with a basilar and subpleural distribution. Diagnostic considerations for fibrotic lung disease, such as usual interstitial pneumonia (UIP), involve identifying a combination of clinical and imaging features, including progressive dyspnea, clubbing, and bibasilar crackles, along with characteristic HRCT findings like traction bronchiectasis and macrocystic honeycombing. HRCT plays a crucial role in differentiating UIP from other idiopathic interstitial pneumonias and guiding biopsy sites for definitive diagnosis.

Assessment Question 1
What is a characteristic radiologic finding in fibrotic lung disease?

- A. Pulmonary edema
- B. Reticulonodular pattern
- C. Cardiomegaly
- D. Pleural effusion

Answer: B
Explanation: Fibrotic lung disease is characterized by specific radiologic findings, including a reticulonodular pattern, which is a key feature of this condition.

Assessment Question 2
What is the typical distribution of radiologic findings in fibrotic lung disease?

- A. Apical and central
- B. Basilar and subpleural
- C. Upper lobe and peripheral
- D. Diffuse and homogeneous

Answer: B
Explanation: Fibrotic lung disease typically shows a basilar and subpleural distribution of radiologic findings, which is an important characteristic for diagnosis.

Final Educational Material
Fibrotic Lung Disease
Fibrotic lung disease is characterized by key radiologic findings including a reticulonodular pattern, ground-glass opacities, and honeycombing, typically with a basilar and subpleural distribution. The diagnosis often involves identifying a combination of clinical features such as progressive dyspnea, clubbing, and bibasilar crackles, along with characteristic HRCT findings like traction bronchiectasis and macrocystic honeycombing. High-resolution computed tomography (HRCT) plays a crucial role in differentiating usual interstitial pneumonia (UIP) from other idiopathic interstitial pneumonias and in guiding biopsy sites for a definitive diagnosis. A collaborative diagnostic approach combining radiologic and histopathologic assessments can lead to a more reliable identification of the predominant pathways leading to pulmonary fibrosis, which has implications for therapy and future research directions.

C.4 Lowest-Scoring Case with MedGemma-27B-text-it

Case Information

Dataset: RexGradient-160K
Generator Model: MedGemma-27B-text-it
Case ID: pGRDNRZB2HZ56F0ZL_aGRDNZSUP68GUCQN2_s1.2.826.0.1.3680043.8.498.16841148712021794680399240737673267273

Original Radiology Report

Indication: Fall yesterday. Fever this morning.
Findings: 4228 hours. The heart size and mediastinal contours are stable. There is stable mild subsegmental atelectasis at both lung bases. No confluent airspace opacity, pleural effusion or pneumothorax. Fracture of the proximal right humerus again noted.
Impression: Stable mild bibasilar atelectasis. No acute cardiopulmonary process.

Generated Educational Materials

Keywords: atelectasis, fracture

Retrieved Paper 1: Debunking a mythology: Atelectasis is not a cause of postoperative fever.
Most physicians appreciate that practicing medicine is a commitment to continuous learning. However, "learning" can be mistakenly understood as simply the acquisition of facts and new knowledge. But learning also necessitates the constant re-examination and challenging of one's existing body of knowledge, as misinformation persists when one's beliefs are not challenged or questioned in the light of new information. One example is the pervasive... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/39566396/ | Source: PubMed

Retrieved Paper 2: Use of artificial intelligence in triaging of chest radiographs to reduce radiologists' workload.
OBJECTIVES: To evaluate whether deep learning-based detection algorithms (DLD)-based triaging can reduce outpatient chest radiograph interpretation workload while maintaining noninferior sensitivity.
METHODS: This retrospective study included patients who underwent initial chest radiography at the outpatient clinic between June 1 and June 30, 2017. Readers interpreted radiographs with/without a commercially available DLD that detects nine... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/37615766/ | Source: PubMed

Retrieved Paper 3: Assessment of proximal tibial fractures with 3D FRACTURE (fast field echo resembling a CT using restricted echo-spacing) MRI-intra-individual comparison with CT.
OBJECTIVES: To evaluate the feasibility and diagnostic performance of a 3D FRACTURE (fast field echo resembling a CT using restricted echo-spacing) MRI sequence for the detection and classification of proximal tibial fractures compared with CT.
METHODS: We retrospectively included 126 patients (85 male; 39.6±14.5 years) from two centers following acute knee injury. Patients underwent knee MRI at 3T including FRACTURE-MRI. Additional CT was... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/40126605/ | Source: PubMed

Retrieved Paper 4: How I Do It: Evaluating Cardiac Implantable Devices and Noncardiac Mimics on Chest Radiographs.
Cardiac implantable electronic devices (CIEDs), including pacemakers and defibrillators, are increasingly used to manage various cardiac conditions. This article reviews the radiographic appearance, typical components, and placement of CIEDs, including newer technologies like leadless pacemakers and MRI-conditional devices. The article also highlights the imaging findings of common complications such as lead dislodgement, fracture, and... (Omitted)
URL: https://pubmed.ncbi.nlm.nih.gov/40358448/ | Source: PubMed

---

Textbook Summary 1
Atelectasis is characterized by radiologic findings of increased lung density, crowding of pulmonary vessels, and displacement of fissures, hilum, and mediastinum, often with elevation of the hemidiaphragm. On MRI, obstructive atelectasis shows high signal intensity on T2WI, while nonobstructive atelectasis shows low signal intensity. The diagnosis of atelectasis considers various causes, including obstructive, nonobstructive, passive, adhesive, and cicatrizing types, each with distinct pathophysiology and imaging features.

Textbook Summary 2
Fracture is not a primary focus in the provided textbook pages, which mainly discuss lymphoma and malignant melanoma, but radiologic findings related to fractures are not mentioned. However, in the context of lymphoma, particularly enteropathy-associated T-cell lymphoma, complications such as bowel perforation can occur, which may be associated with a fracture of the bowel wall. Diagnostic considerations for lymphoma include imaging findings like circumferential wall thickening on CT and FDG uptake on PET.

Assessment Question 1
What is a characteristic radiologic finding of atelectasis?

- A. Decreased lung density
- B. Increased lung density
- C. Dilatation of pulmonary vessels
- D. Elevation of the diaphragm without crowding of pulmonary vessels

Answer: B
Explanation: Atelectasis is characterized by radiologic findings of increased lung density, crowding of pulmonary vessels, and displacement of fissures, hilum, and mediastinum.

Assessment Question 2
Which of the following statements about postoperative atelectasis is supported by the provided context?

- A. Atelectasis is a common cause of postoperative fever.
- B. Atelectasis is rarely seen in postoperative patients.
- C. The belief that atelectasis causes postoperative fever is a pervasive myth.
- D. Atelectasis always leads to significant respiratory complications.

Answer: C
Explanation: The context mentions that the belief that postoperative atelectasis causes fever is a myth, despite ample evidence to the contrary, as discussed in the abstract titled "Debunking a mythology: Atelectasis is not a cause of postoperative fever."

Assessment Question 3
What is the sensitivity of FRACTURE-MRI in detecting fractures, as mentioned in the provided context?

- A. 70
- B. 80
- C. 91.5
- D. 95

Answer: C
Explanation: The context states that fractures were detected by FRACTURE-MRI with a sensitivity of 91.5

Assessment Question 4
What is a potential advantage of using FRACTURE-MRI for diagnosing proximal tibial fractures, as mentioned in the context?

- A. Higher radiation exposure
- B. Reduced accuracy in fracture classification
- C. Reduced costs and radiation exposure
- D. Increased need for CT scans

Answer: C
Explanation: The context suggests that FRACTURE-MRI might be a helpful alternative to computed tomography in an acute trauma setting by reducing costs and radiation exposure in patients requiring a preoperative MRI anyway.

Final Educational Material
Atelectasis
Atelectasis is characterized by increased lung density, crowding of pulmonary vessels, and displacement of fissures, hilum, and mediastinum, often with elevation of the hemidiaphragm. It's essential to differentiate atelectasis from consolidation, as their imaging characteristics and clinical implications differ. Atelectasis can be caused by various factors, including obstructive, nonobstructive, passive, adhesive, and cicatrizing types. Notably, atelectasis is not a cause of postoperative fever, contrary to a common myth. The use of artificial intelligence in triaging chest radiographs can help reduce the workload of radiologists while maintaining sensitivity for detecting atelectasis and other radiologic findings.
Fracture
Fracture diagnosis and classification can be accurately performed using 3D FRACTURE (fast field echo resembling a CT using restricted echo-spacing) MRI sequences, which offer a feasible alternative to CT scans in acute trauma settings. This can help reduce radiation exposure and costs, especially in patients requiring preoperative MRI. When evaluating fractures, particularly in the context of cardiac implantable devices, it's crucial to identify potential complications such as lead dislodgement, fracture, and perforation. Accurate identification of these devices and their positioning on chest radiographs is vital for early detection of complications and directing appropriate patient care.

D Default System Prompts for MedTutor

D.1 Keyword Generation Prompt

Keyword Generation Prompt

System Prompt:

```
You are an expert medical language model. Given the full radiology report
and the extracted Impression section, extract all specific disease names,
diagnostic labels, and named pathological entities mentioned or implied in
either section. Focus only on established or suspected diagnoses, such as
named conditions.

Only include diagnoses that are positively identified or suspected in the
report. Do not include any conditions that are explicitly ruled out,
negated, or stated as absent.

Do not include general phrases, symptoms, or clinical findings that are not
formal diagnoses.

Output your answer as a valid JSON object with the following format:

{ "keywords": ["diagnosis 1", "diagnosis 2", "diagnosis 3"] }

If no diagnoses are present, return:

{
"keywords": []
}
```

User Instruction Template:

```
Final_report: {full_report_text}
Impression: {impression_text}
```

D.2 Textbook Summary Prompt

Textbook Summary Prompt

System Prompt:

```
You are a concise and accurate radiology assistant, skilled in summarizing
medical texts.
```

User Instruction Template:

```
Please summarize the following textbook pages focusing on the keyword '{keyword}'.
The summary should highlight key radiologic findings and diagnostic
considerations. Be concise, using 2-3 sentences and your own words.
Output only the summary text itself, with no additional conversational
text or headers.

Textbook Pages Content:
{pages_block_text}
```

D.3 MCQ Generation Prompt

Multiple Choice Q&A Generation Prompt

System Prompt:

```
You are a specialized AI assistant for creating multiple-choice questions (MCQs)
for radiology education. You must focus *exclusively* on the provided
**Primary Diagnostic Keywords**.
```

User Instruction Template:

```
### Primary Diagnostic Keywords to Focus On:
- {keywords_list_str}

### Full Context (for reference)
{mcq_input_context}

### Your Task
Based *only* on the provided context, generate 2 multiple-choice questions
**for each Primary Diagnostic Keyword listed above**. Do not generate questions
for any other terms or topics mentioned in the context. Each question must
test understanding of the information related to the primary keywords.

Follow this format exactly:

### Multiple Choice Questions

#### {{Diagnosis Keyword 1}}

Q1. {{Question stem}}
A. {{Option A}}
B. {{Option B}}
C. {{Option C}}
D. {{Option D}}
Answer: {{Correct Option Letter}}
Explanation: {{Brief explanation based on the provided context.}}
```

D.4 Educational Material Generation Prompt

Educational Material Prompt

System Prompt:

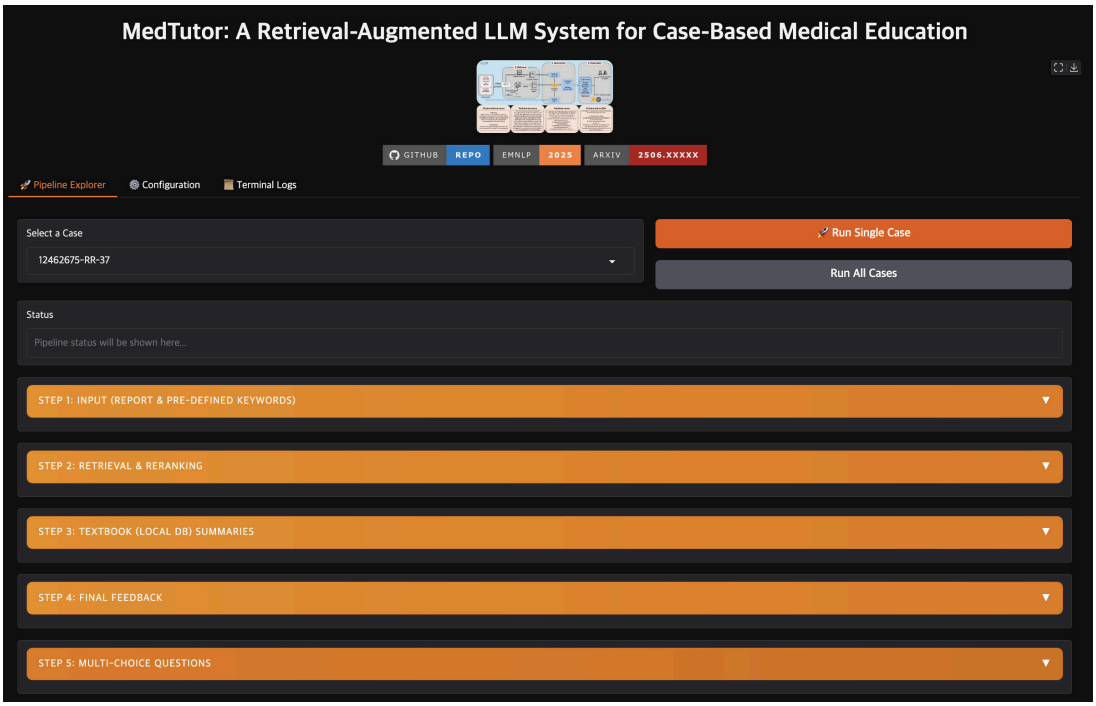
```
You are an expert radiology AI assistant. Your task is to synthesize the
provided information into concise, educational feedback focused *only* on the
primary diagnostic keywords provided. Do not explain or elaborate on other
terms from the original report unless they are directly relevant to the
primary keywords.
```

User Instruction Template:

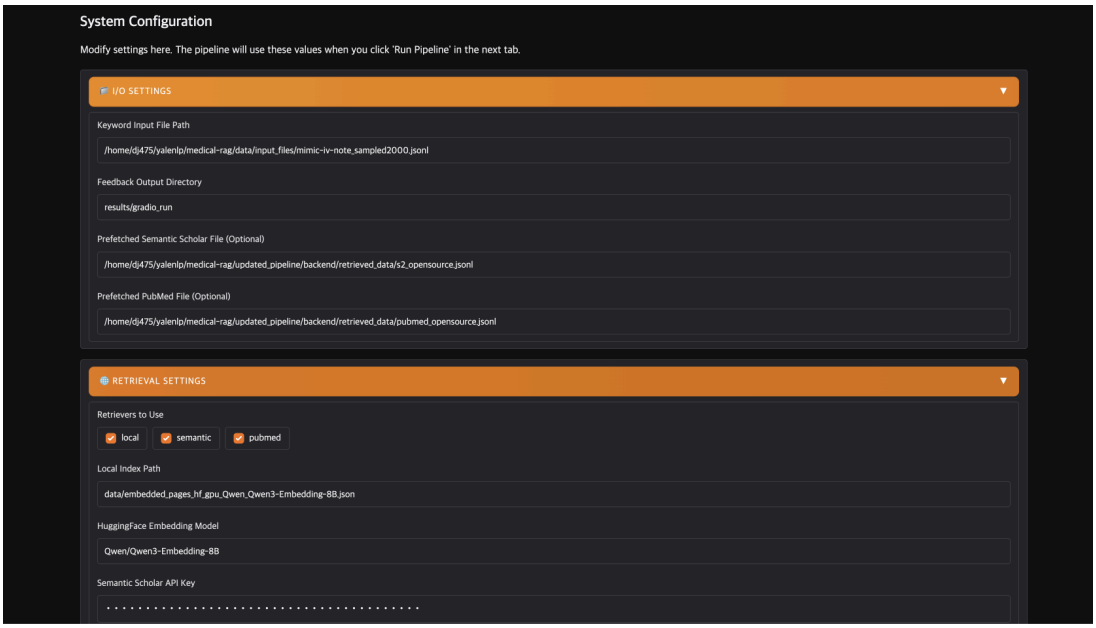
```
### Primary Diagnostic Keywords
- {keywords_list_str}

### Original Reviewer Report (for context only)
```

```
{original_reviewer_report}

### Supporting Educational Material
{user_block_for_final_stages}

### Your Task
Based on all the information above, provide a concise, synthesized feedback.
Structure your response with a section for each **Primary Diagnostic Keyword**.
Focus only on clinical teaching points and imaging pearls related to these
primary keywords.
```

E MedTutor System UI

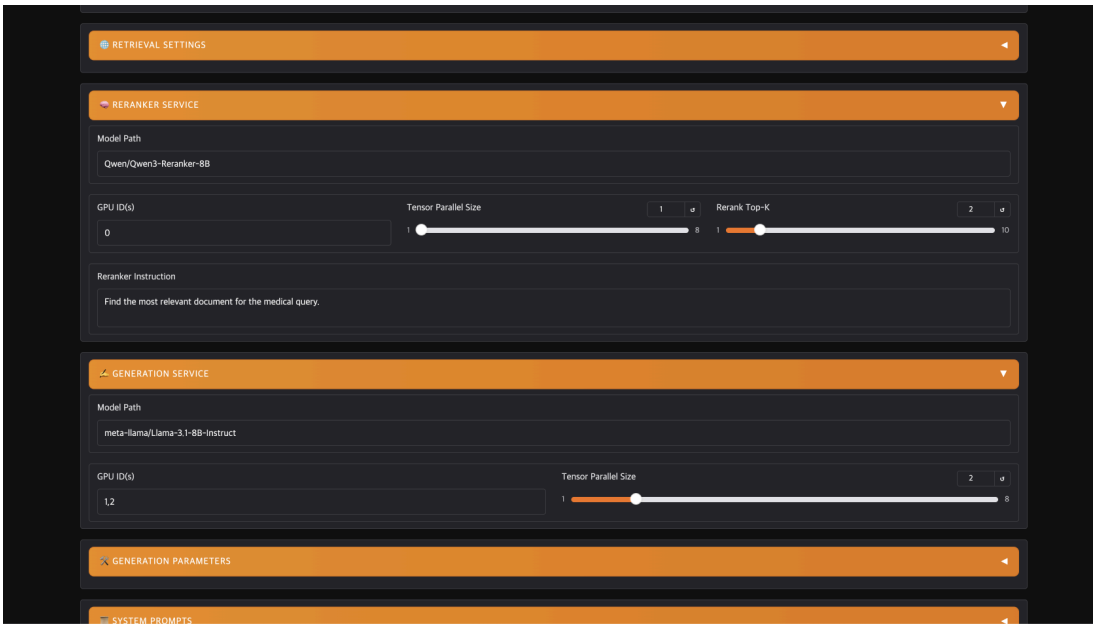


(a) Main user interface of MedTutor.

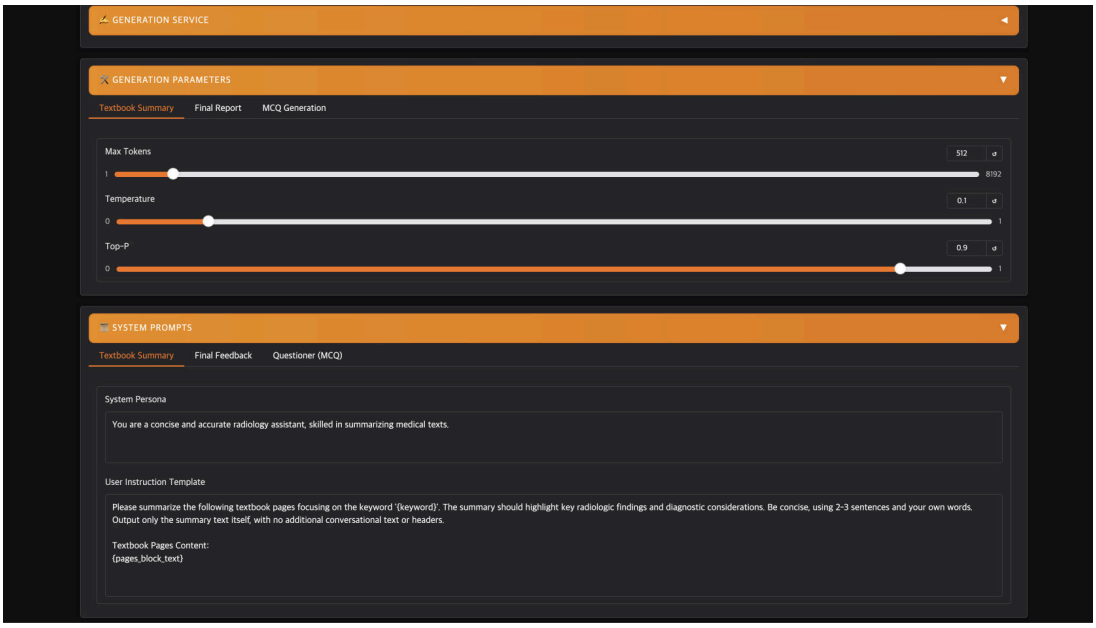


(b) Configuration settings for model selection and system prompts.

Figure 5: The MedTutor UI (Part 1 of 2): Main dashboard and initial configuration settings.



(a) Further configuration for data sources and retrieval.



(b) Finalizing configuration and execution options.

Figure 6: The MedTutor UI (Part 2 of 2): Additional configuration panels for data processing and task execution.

F Human Annotation Guideline

I. Evaluation of Information Quality per Keyword

For each diagnostic keyword identified from the original report, we evaluate the following components:

F.1 Retrieved & Reranked Academic Papers

- **Relevance to Keyword & Original Report:** How directly related is each paper or retrieved snippet to the given keyword and the context of the original radiology report?

F.2 Generated Textbook Summary

- **Accuracy & Factuality:** Is the summary an accurate and factual representation of information related to the keyword (compared to general radiology knowledge or, if available, the source textbook)?
- **Helpfulness & Relevance:** Is the summary helpful and related to the case report provided as input?
- **Coverage of Key Information:** Does the summary include the most critical information (e.g., key imaging findings, diagnostic criteria) related to the keyword?

F.3 Example Multiple Choice Questions

- **Relevance & Correctness:** Are the questions relevant to the keyword? Is the answer provided and rationale correct? Are the answer choices relevant?

II. Evaluation of the "Educational material" Paragraph (per Keyword)

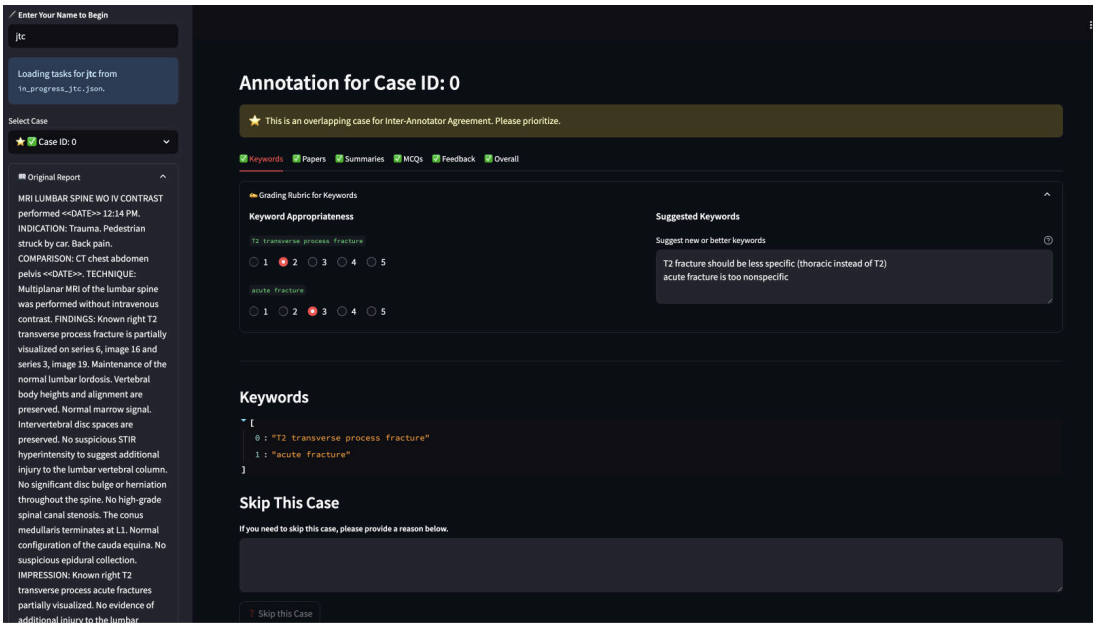
- **Clinical & Educational Utility:** How clinically relevant, accurate, and educationally valuable is this educational material paragraph for a radiology trainee in understanding the keyword within the context of the original report? (This encompasses quality, clinical insight, contextual appropriateness, and trustworthiness.)

III. Evaluation of Overall Educational Material Structure & Quality

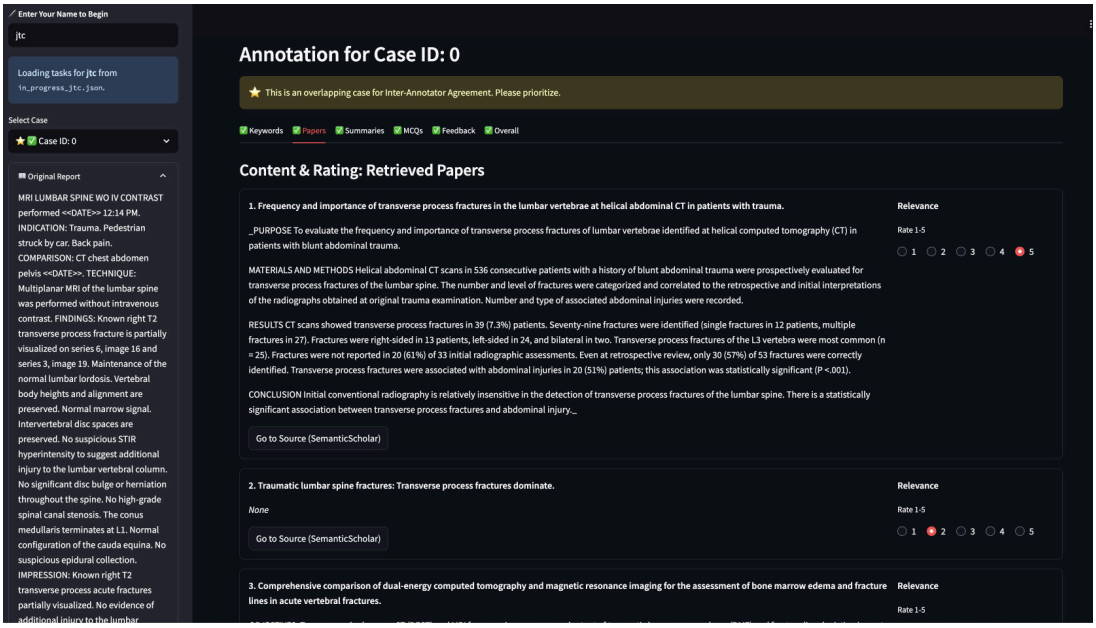
For the entire generated report:

- **Appropriateness of Keywords:** Are the keywords (used to structure the feedback) appropriate and comprehensive for the given original radiology report? Specifically, is the keyword general enough that it can be searched in a textbook or Radiopaedia (e.g., “rib fracture”, not “anterior 4th rib fracture”), and related to a pathology worth learning more about (e.g., “cholangiocarcinoma”, not “mass”)?

G Human Annotator System UI

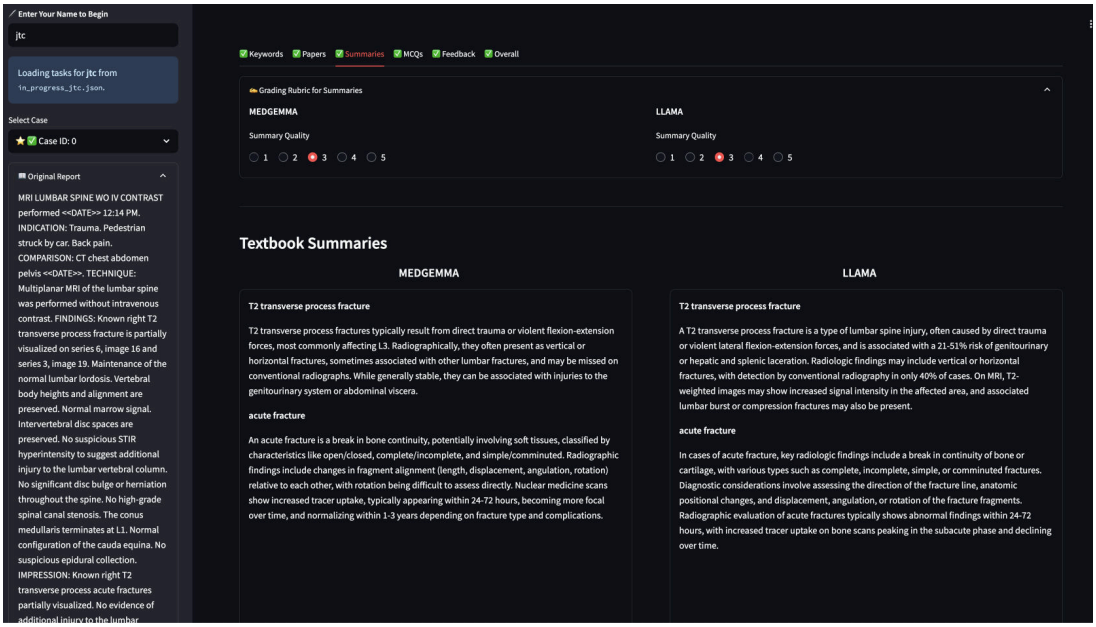


(a) Keyword Evaluation Page

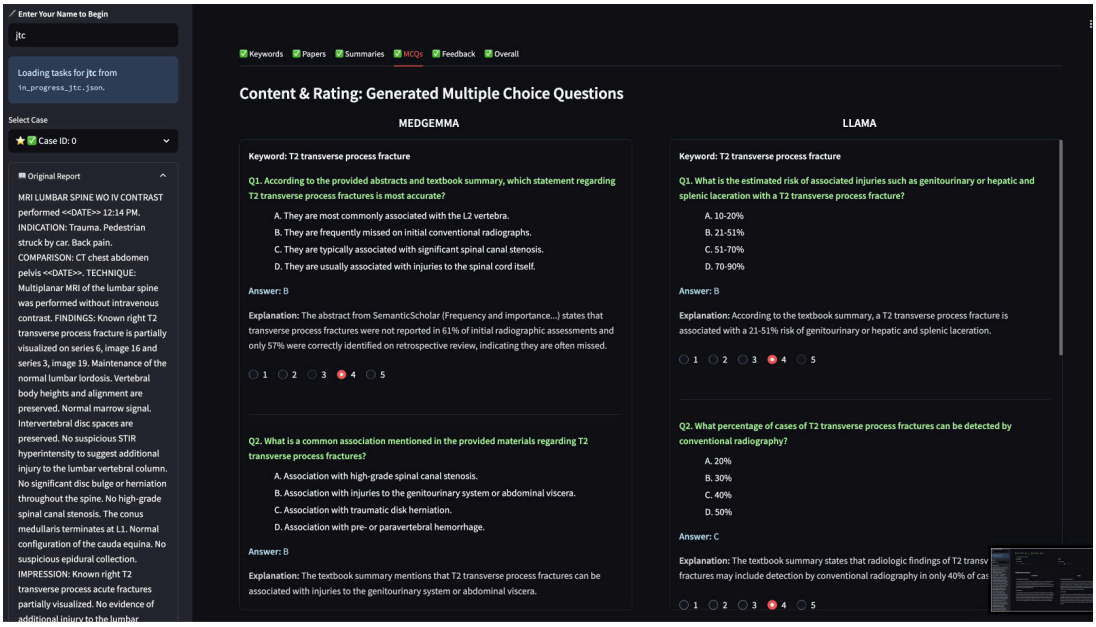


(b) Paper Evaluation Page

Figure 7: Annotation system UI (Part 1 of 3): Interfaces for evaluating keywords and retrieved papers.

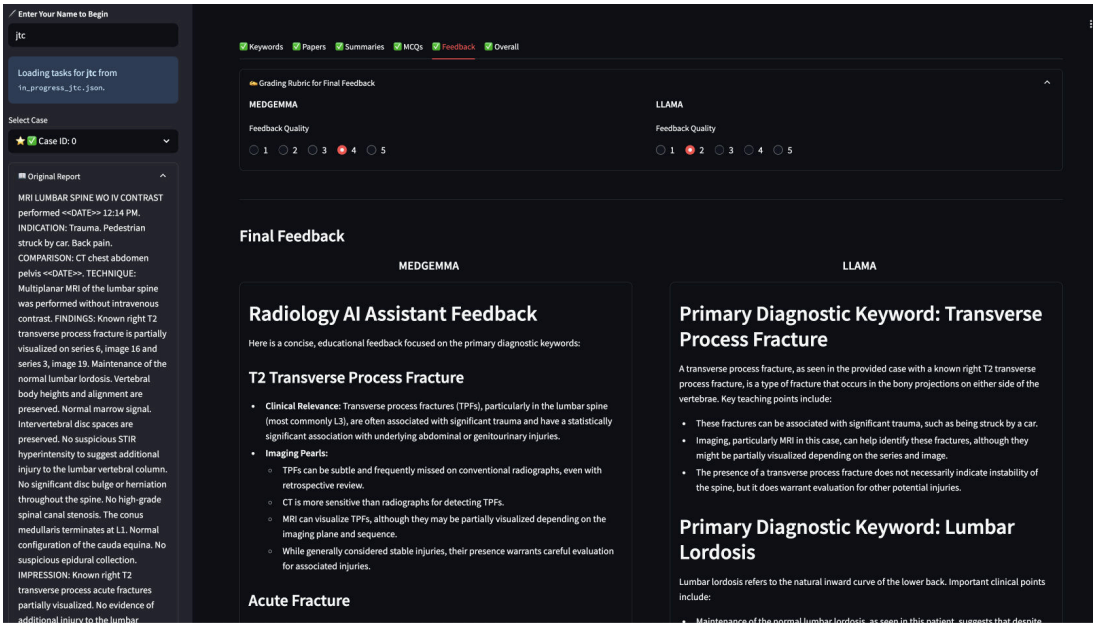


(a) Textbook Summary Evaluation Page

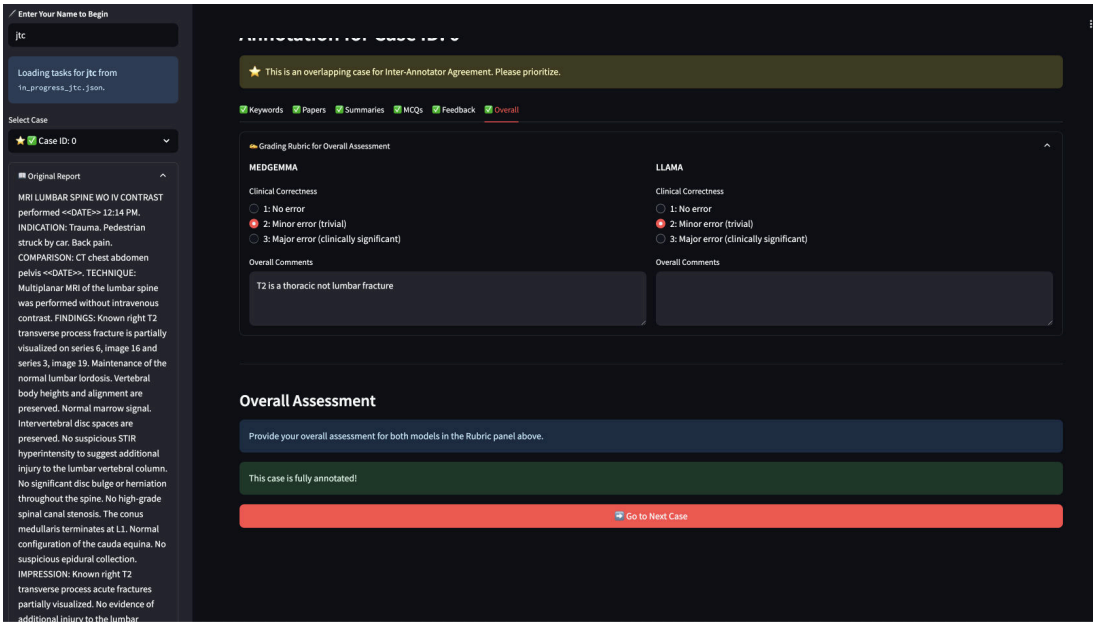


(b) MCQ Evaluation Page

Figure 8: Annotation system UI (Part 2 of 3): Interfaces for evaluating textbook summaries and multiple-choice questions.



(a) Educational Material Evaluation Page



(b) Overall Quality Evaluation Page

Figure 9: Annotation system UI (Part 3 of 3): Interfaces for evaluating the final synthesized educational material and overall quality.

Co-DETECT: Collaborative Discovery of Edge Cases in Text Classification

Chenfei Xiong[Z*] **Jingwei Ni**[E Z*] **Yu Fan**[E*] **Vilém Zouhar**[E]
Donya Rooein[B E] **Lorena Calvo-Bartolomé**[C] **Alexander Hoyle**[E]
Zhijing Jin[T] **Mrinmaya Sachan**[E] **Markus Leippold**[Z]
Dirk Hovy[B] **Mennatallah El-Assady**[E] **Elliott Ash**[E]
[E]ETH Zürich [Z]University of Zürich
[B]Bocconi University [T]University of Toronto [C]Universidad Carlos III
{jingni, yufan, ashe}@ethz.ch

Abstract

We introduce Co-DETECT (Collaborative Discovery of Edge cases in TExt ClassificaTion), a novel mixed-initiative annotation framework that integrates human expertise with automatic annotation guided by large language models (LLMs). Co-DETECT starts with an initial, sketch-level codebook and dataset provided by a domain expert, then leverages the LLM to annotate the data and identify edge cases that are not well described by the initial codebook. Specifically, Co-DETECT flags challenging examples, induces high-level, generalizable descriptions of edge cases, and assists user in incorporating edge case handling rules to improve the codebook. This iterative process enables more effective handling of nuanced phenomena through compact, generalizable annotation rules. Extensive user study, qualitative, and quantitative analyses prove the effectiveness of Co-DETECT.[1]

1 Introduction

Social scientists often find themselves in situations requiring data annotation based on human judgment and specific expertise (Wilkerson and Casas, 2017; Kennedy et al., 2018; Demszky et al., 2020; Drápal et al., 2023). For example, a political scientist studying hate speech on social media may need to develop a codebook that clearly defines what constitutes hate speech in the social media scenario with illustrative positive and negative examples. After developing such a codebook, the researcher may need to recruit annotators possessing sufficient domain expertise to appropriately apply the established guidelines to actual social media content.

However, both tasks—codebook development and data annotation—involve significant human effort.

Firstly, it is challenging even for domain experts to develop reliable codebook (Halterman and Keith, 2025). To start with, ambiguity and subjectivity are inherent obstacles, as interpreting complex human behaviors and communications often yields multiple valid perspectives, leading to edge cases that need specific rules to handle (Fornaciari et al., 2021; Fuchs et al., 2021; Fleisig et al., 2023; Fan et al., 2025b). It is usually infeasible for an expert to manually examine the entire target corpus in order to identify the many edge cases that arise. In addition, codebook developers may introduce biases or subjective interpretations shaped by their domain knowledge and socio-demographic background, which can limit the effectiveness of the codebook in handling difficult or ambiguous cases. Even when an edge case is identified, the expert may struggle to explicitly articulate the subtleties and intuitions that guide their judgments, a phenomenon commonly known as *Polanyi's paradox*[2] (Autor, 2014; Fügener et al., 2022). As a result, although domain experts may possess rich implicit understandings of certain social phenomena, they face significant challenges in capturing and codifying this tacit knowledge within verbal, structured annotation frameworks.

Secondly, large-scale human annotation is often infeasible in many use cases (e.g., Xie and Zhang, 2024). Employing qualified annotators is costly, especially when the task requires domain-specific expertise. Furthermore, domain-specific data frequently involve nuanced and complex contexts (e.g., Ziems et al., 2024; Fan et al., 2025a; Zhao et al., 2025), which demand greater cognitive effort and longer annotation times to ensure high quality. As a result, large-scale human annotation is often prohibitively expensive in terms of both cost and time. To address this challenge,

[*]Equal contribution.

[1]Frontend: codetect.vercel.app; code and demonstration video: github.com/EdisonNi-hku/Co-DETECT

[2]In everyday language, this phenomenon is often summarized by the phrase: "We can know more than we can tell."

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 354–364
November 4-9, 2025

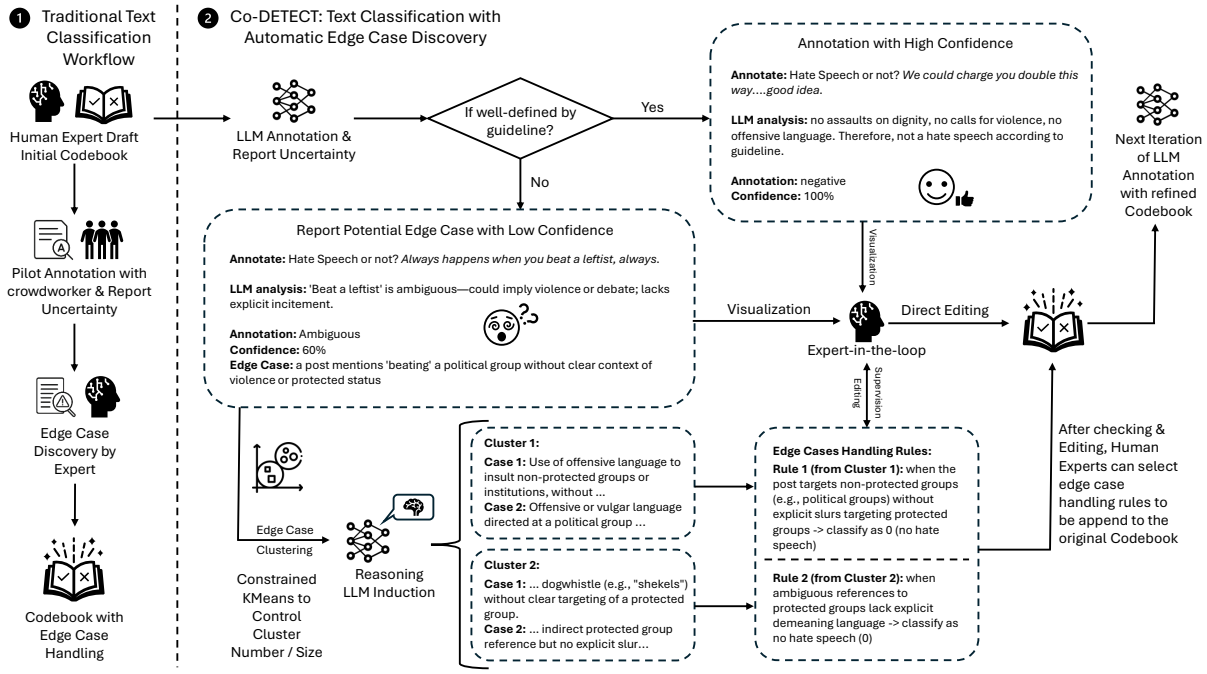


Figure 1: ❶ traditional workflow of text annotation, where experts rely on their own or crowdworkers to identify edge cases and update codebook based on the discovered prevalent edge cases. ❷ Co-DETECT mixed-initiative workflow of edge case discovery, where LLMs propose prevalent and representative edge cases and the visual interface assists human expert to verify the proposed edge cases.

recent research explores leveraging LLMs for automatic annotation (Gilardi et al., 2023; Pangakis et al., 2023; Ding et al., 2023; He et al., 2024b,a; Dunivin, 2024; Törnberg, 2024). These approaches typically assume the availability of well-developed codebooks for LLM prompting (Halterman and Keith, 2025; Xiao et al., 2023). However, how codebook development and the annotation process interact—and, crucially, how expert knowledge shapes this interaction—remains underexplored. In practice, domain experts often iterate between the codebook and annotation results, updating the codebook based on insights gathered during annotation (e.g., Kirsten et al., 2025). This iterative process is essential for uncovering edge cases that a previous codebook may have overlooked, enabling experts to revise the codebook accordingly.

To address these gaps, we introduce Co-DETECT, a mixed-initiative text analysis tool designed to support domain experts in discovering and managing edge cases (Figure 1). Co-DETECT takes annotation guidelines (codebooks) and a target annotation corpus as input. It identifies edge cases—data points poorly defined by the provided codebook, clusters them strategically, and proposes aggregated edge categories with representative examples. The user then evaluates these suggestions, determines their validity, and updates the codebook accordingly with clear handling rules. Finally, annotation can proceed in a new iteration using the revised codebook.

This expert-in-the-loop approach enables humans and AI to complement one another by leveraging their respective strengths in collaboration. Prior work shows that humans often struggle to identify edge cases due to limited *metaknowledge*—the ability to assess the scope and boundaries of their own knowledge (Fügener et al., 2022; Evans and Foster, 2011). By contrast, AI systems are better at uncovering edge cases, though their ability to address them remains limited (Ni et al., 2025a). Accordingly, incorporating AI into an expert-in-the-loop framework is advisable: AI can surface edge cases and enrich the codebook, while human oversight ensures appropriate interpretation and handling. In summary, our contributions are:

1. We develop Co-DETECT for domain experts that iteratively updates the codebook under human supervision. Consists of an LLM-based induction algorithm suggesting representative edge cases and a user-friendly interface enabling domain experts to more effectively handle edge cases.

2. We conduct user studies with domain experts from diverse backgrounds, shedding light on the effectiveness of Co-DETECT and directions for

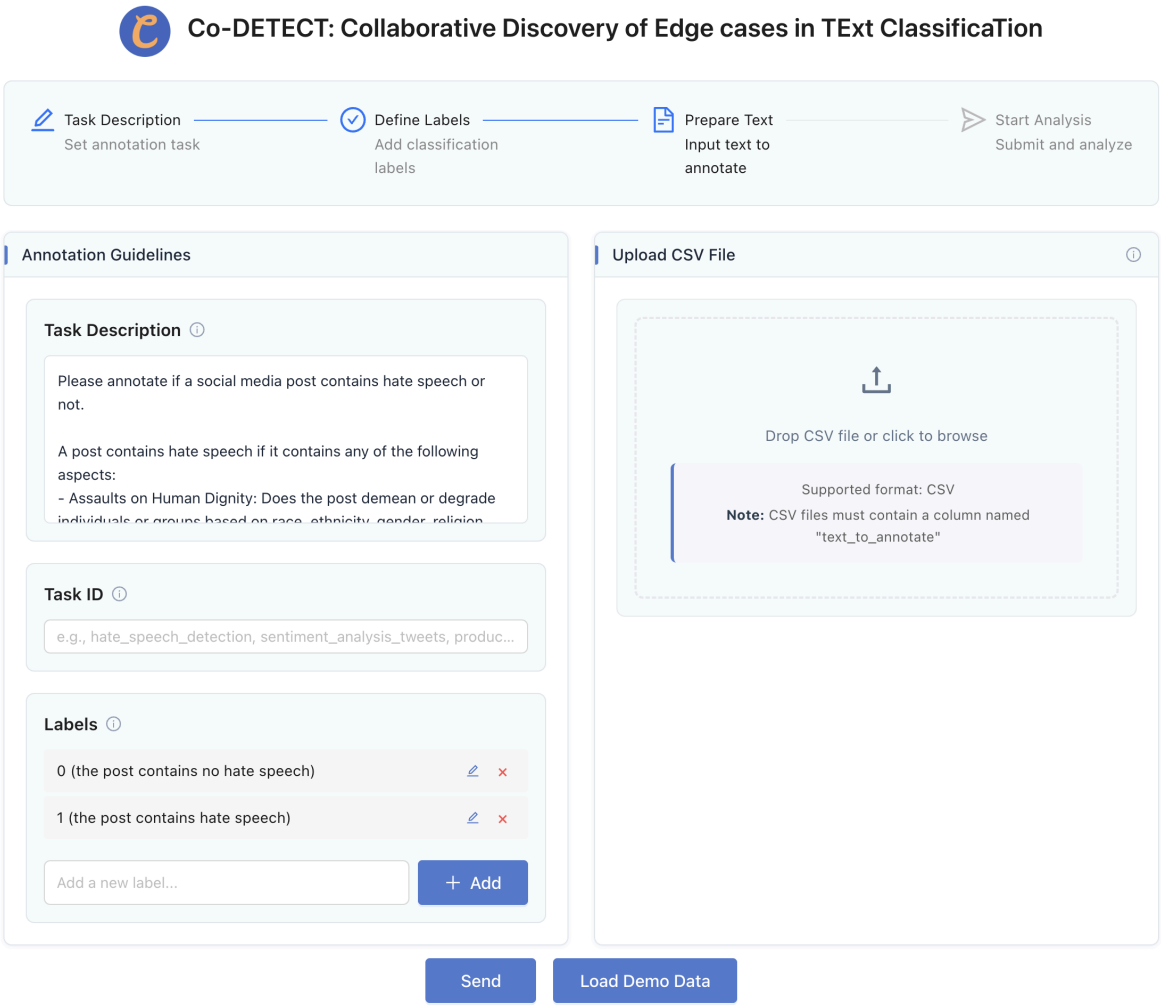


Figure 2: User Interface – Input Page

future work.

2 Frontend and User Work Flow

In this section, we provide a detailed introduction to the user workflow (illustrated in fig. 1), including a preparation stage section 2.1 and a dashboard analysis stage section 2.2.

2.1 Preparation Stage

Onboarding. To ensure a smooth onboarding experience, the first launch of Co-DETECT triggers an intro.js tour that guides users through the input and dashboard pages. Furthermore, users can also click "Load Demo Data" to explore a sample usecase in annotating hate speech from social media.

Input Page. After familiarized with Co-DETECT, the user can start from the input page (Figure 2), where they need to provide a initial draft of the codebook, including (1) a task definition (e.g., a post contains hate speech if it contains assaults on human dignity, calls for violence, or vulgarity.); (2) classification labels (e.g., 1 for hate speech and 0 for no hate speech); and (3) a task ID which is useful for saving all annotation outcomes, edge cases, and codebooks to the backend. Besides the codebook, the user also need to provide a csv file containing 500 to 1000 target texts to be annotated. We suggest this number of texts to ensure the representativeness of edge cases with reasonable budget. With the input prepared, the user can click "send" to pass the inputs to the LLM analysts.

2.2 Dashboard Analysis Stage

Current Guidelines. At the upper left-hand side of the dashboard, the user-provided codebook is displayed, allowing users to optimize their annotation task descriptions and manage labels.

Exploring Annotation Results. At the upper middle of the dashboard, we provide a scatter plot

Figure 3: User Interface – Analysis Dashboard

showing all annotated text samples. Each point represents an example, clustered according to embedding similarity. Different colors indicate different annotation labels, and point size denotes annotation uncertainty—how likely the samples belong certain edge cases. Clicking on points in the scatter plot, the annotation details of the corresponding items will pop out on the upper right "All Examples" list, including LLM analysis, annotation confidence, and edge case suggestions.

Analyzing Edge Cases. Either clicking large points (uncertain annotations) in the upper scatter plot or "Edge" items in the upper right list will connect the user to the lower middle scatter plot and lower right list. The lower middle plot presents the clusters of potential edge cases that may require attention. These samples are automatically identified by the system as challenging or requiring more precise annotation guidelines. The panel named "Suggested Edge Cases" on the lower right outlines high-level descriptions of each edge case cluster, and examples in each cluster.

Edge Case Handling and Iterative Optimization. Once users find any cluster in "Suggested Edge Cases" reasonable, they can add the corresponding edge case handling rule to the lower left panel "Edge Case Handling". For example, clusters A and C are added in Figure 3. Users can also edit the edge case handling rules freely. Once they are satisfied with the added rules, they can click "Iterate" on the top left to re-annotate the corpus with the codebook augmented with "Edge Case Handling". Codebook of previous iterations will be saved in the top left panel—"Previous Guidelines".

3 Backend Algorithm for Edge Case Discovery

Problem Formulation. As illustrated in fig. 1, either traditional text annotation or Co-DETECT requires a target corpus and task definition (i.e., initial codebook) as inputs. At this stage, the user may lack insights about the corpus, including limited edge case handlings in the codebook. Co-DETECT aims at discovering edge cases that are ambiguously defined by the codebook. The edge cases proposed by Co-DETECT should be:

- **Descriptive**: capture the core features of exact edge case samples and the reason why they are ambiguous.
- **High-Level**: while being descriptive, the edge case descriptions should not over specifically describe certain samples. Only then can they be added to the codebook and generalized to unseen data points.

To fulfill these desiderata, the edge case discovery algorithm of Co-DETECT details as follows:

Step 1: Item-Level Edge Cases. We start from using a non-reasoning LLM[3] (e.g., in our case GPT-4.1) to quickly annotate all data points. We prompt

[3]We use reasoning LLMs referring to LLMs with test-time long CoT reasoning (e.g., DeepSeek-R1 (DeepSeek-AI, 2025) and OpenAI O3 (OpenAI, 2025b)); and non-reasoning LLMs referring to those answer immediately (e.g., GPT-4.1 (OpenAI, 2025a)).

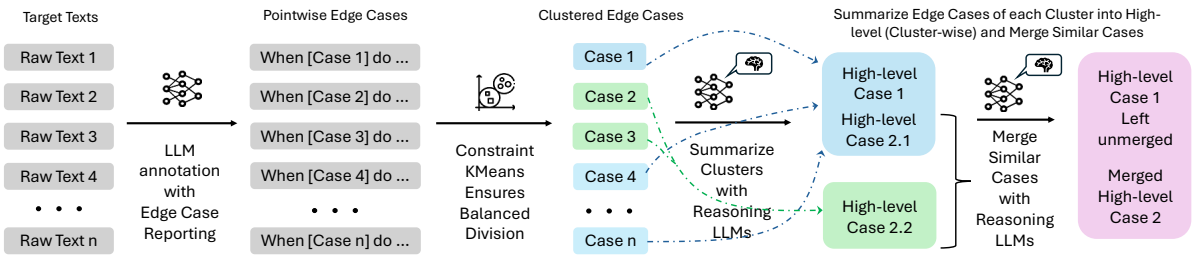


Figure 4: Co-DETECT's backend algorithm for automatically discover representative edge cases. Firstly, an LLM annotator report pointwise edge cases. Secondly, a reasoning LLM aggregates item-wise edge cases into more representative high-level edge cases, with the help of clustering algorithms.

the LLM to (1) annotate, (2) provide a confidence score reflecting the annotation correctness (following Tian et al. (2023b)), and (3) explain why the case is an edge case if the confidence is low. The explanations are in a form of edge case handling rules like "when `[Case Description]`, do `[Action]`". `[Case Description]` describes why the sample is ambiguously defined, so it might be too specific and low-level. `[Action]` is an LLM suggested handling for the edge case.

Step 2: Cluster-Level Edge Cases. To avoid over specific `[Case Description]` that fails to generalize to other samples, we need to aggregate item-level edge cases with similar ambiguity and describe them in a higher level. This is a challenging task requiring (1) covering all item-level edge cases; and (2) strategically finding logical similarities between reasons for ambiguity. Therefore, we employ a SOTA reasoning LLM—DeepSeek-R1 (DeepSeek-AI, 2025) to cluster `[Case Description]` and generate high-level edge cases and handling rules. Specifically, we extract the item-level `[Case Description]`, embed them with semantic embedding models[4], and cluster them with constrained KMeans (Levy-Kramer, 2018). Each cluster of `[Case Description]` and corresponding `[Action]` are fed to DeepSeek-R1 to generate Cluster-wise Edge Cases. Constrained KMeans ensures that all clusters have 10 to 20 samples, so that the input (i.e., each cluster) to DeepSeek-R1 will not be too large or small, as we empirically find that large clusters (>20) may increase the reasoning burden and lead to hallucination, while small clusters (<5) may generate over-specific edge cases. Cluster-level edge cases are also companied with `[Action]` to handling them.

Step 3: Merge Cluster-Level Edge Cases. Since each cluster may have overlapped edge cases, we finally call DeepSeek-R1 again to merge cluster-level edge cases and their handling rules. This also ensures that similar edge cases are not handled with different rules.

4 User Study

To evaluate Co-DETECT's effectiveness and collect feedback for further improvement, we conduct a systematic user study with domain experts. Prior to the study, participants were asked to prepare a text annotation task and an accompanying corpus from their own research domains (i.e., areas where they possess domain expertise). At the beginning of the user study, participants first complete a pre-interaction survey gathering their background information. Then, they interact with the system for approximately 45 minutes, including the LLM response time. Finally, they complete a post-interaction survey, collecting comprehensive user feedback. We recruit 10 users in total. 5 of them are not involved in the design of Co-DETECT. The remaining five are co-authors of the paper but were not familiarized with the workflow before the user study.

4.1 Pre-Interaction Survey Takeaways

We summarize the key findings from the pre-interaction survey below. For the full survey form, please refer to Appendix A.

Diverse Experience and Background of the Participants. Our participants have a broad academic background in social science, computational linguistics, and interdisciplinary training. They also exhibit diverse experience in both social science qualitative coding and LLM-assisted annotation, from no experience to expert level.

Heavy Reliance on Manual Effort for Edge Case Discovery. Concerning common workflows for identifying edge cases, 80% of participants manually review subsets of data to detect potential edge

[4]In our project, we use OpenAI text-embedding-3-large (OpenAI, 2024) for convenience.

cases. Some also report employing other human (e.g., crowdworkers) or AI annotators to do pilot annotation and flag potential edge cases.

Moderate Prior Knowledge of Edge Cases. 70% of participants report knowing certain edge cases in their intended datasets. Therefore, it would be valuable to check if Co-DETECT can mine already-known edge cases or discover new edge cases.

4.2 Post-Interaction Survey Takeaways

Below, we highlight the main insights from the post-interaction survey, which center on four key aspects of user experience with Co-DETECT: ease of use, interpretability of visualizations, validity of edge cases, and overall satisfaction and feedback. For the full survey form, please refer to Appendix B.

The Majority Finds Co-DETECT Workflow Easy to Follow. The survey results reveal generally positive feedback on interface ease of use and task clarity, with most participants (80%) finding navigation intuitive and interaction straightforward. Some requested additional visualizations (e.g., density distribution of the confidence scores) or export features for enhanced usability.

Co-DETECT Can Identify Relevant Edge Cases. 60% of participants approved Co-DETECT's ability to clearly identify relevant edge cases. For example, one participant reported that the edge case handling rules suggested are "clear, realistic, and concise", and directly inform their acceptance or rejection decisions, pointing out ways to improve the precision and coverage of these suggestions. 90% of participants report that Co-DETECT may help discover new edge cases beyond their prior knowledge of the dataset.

Useful Iterative Workflow and Overall Satisfaction. 80% of participants find the iterative feature of Co-DETECT useful for refining their annotation guidelines. All participants were satisfied with the Co-DETECT system's support in generating annotation guidelines and identifying new edge cases.

Constructive Critiques. Besides the generally positive feedback on user experience of Co-DETECT, the post-interaction survey also gives us valuable critiques, highlighting areas for future improvement of Co-DETECT. 40% of participants express concern that Co-DETECT may overlook potential edge cases although the identified edge cases seem reasonable. For instance, one participant also indicated that there was room to enhance the coherence and descriptive clarity of the edge cases.

Dataset	1st Iter.	2nd Iter.
GabHateCorpus	0.2144	**0.2523**
GoEmotions-Positive	0.0300	**0.3297**
GoEmotions-Negative	0.2823	**0.3046**

Table 1: Classification F1 Scores using the original codebook (from the 1st iteration) and the improved codebook after one Co-DETECT iteration (from the 2nd iteration).

4.3 Quantitative Human Evaluation on Edge Case Validity

We further conducted a quantitative human evaluation with three participants[5]. Each participant was asked to randomly select 1 to 2 samples from each edge case cluster and manually assess how many were accurately captured by the Co-DETECT-suggested edge case descriptions. Among 41 randomly selected samples, 33 (**80.5%**) were reported as well-described by the suggested edge case descriptions. The edge case descriptions are also found to be sufficiently high-level to cover more than one samples. We further find that it often takes less than 5 seconds for an expert to identify if a sample is covered by an edge case description or not, indicating that Co-DETECT may not impose a heavy cognitive load on users when supervising suggested edge case clusters.

5 Can Improved Codebook Benefit Automatic Annotation?

The main goal of Co-DETECT is to help experts improve codebooks, but does a better codebook actually enhance automatic annotation? To investigate this, we provide GPT-4.1 with codebooks before and after Co-DETECT enhancement and compare its classification F1, varying only the codebook. We strategically pick a hate speech detection dataset—GabHateCorpus (Kennedy et al., 2021) and an emotion classification—GoEmotions (Demszky et al., 2020; Positive / Negative Emotion Detection) for this evaluation, because these tasks are highly subjective (Davani et al., 2022; Ni et al., 2025a) and thus challenging for codebook drafting. They are therefore challenging for advanced LLMs like GPT-4.1 that are smart enough to understand

[5]Due to the original user study is already time-intensive, participation in the quantitative evaluation was optional.

the nuanced perturbations within the codebook. It is also challenging for human experts to manually improve codebook as it is hard for individuals to capture various subjectivity.

The results are exhibited in table 1, where we observe an increase in F1 scores across different datasets. Notably, Co-DETECT only augments codebook by appending edge case handling rules. The initial codebook for GoEmotion-Positive has very low F1 score due to an extremely low recall—the model rarely predicts positive emotions that are not explicitly stated by the raw codebook. Thereby, we showcase that Co-DETECT can improve classification outcomes with improved codebook, even for subjective tasks that are both challenging for LLMs and individual experts.

6 Related Work

Annotation with LLM Assistance. Both NLP (Kim et al., 2024; Ni et al., 2024, 2025b) and HCI (He et al., 2024b; Törnberg, 2023) community have widely explored human-AI collaborations for text classification. Tian et al. (2023a) find that the verbalized confidence of LLMs indicates classification quality, and annotations where LLM reports high annotation confidence may outperform human annotator (Ni et al., 2024, 2025b; Törnberg, 2023). We follow this stream of work to calibrate the quality of LLM annotation using verbalized confidence scores. In Human-AI interaction, Wang et al. (2024) and Kim et al. (2024) develop mixed-initiative tools to enhance automatic annotation with minimal human supervision. However, these methods focus on annotation accuracy and assume a predefined codebook. In contrast, our work targets efficiency in codebook development and edge-case discovery, critical steps especially in the initial stages of text classification (Törnberg, 2024).

Goal-Driven Clustering in NLP. One critical step of our edge case discovery algorithm is to cluster low-level specific edge cases into high-level representative edge cases. Such goal-driven clustering (Wang et al., 2023) is essentially relevant to many NLP sub-fields, such as topic modeling (Pham et al., 2024), inductive reasoning (Lam et al., 2024), corpus comparison (Zhong et al., 2023), information retrieval (Ni et al., 2025b) etc. In such tasks, LLM plays an important role in understanding users' goal and steering / interpreting the clustering accordingly (Zhang et al., 2023; Viswanathan et al., 2024; Movva et al., 2025). Our work contributes to adapting goal-driven clustering to edge case discovery, leveraging analytical skills of reasoning models (DeepSeek-AI, 2025).

7 Conclusion

We developed Co-DETECT to systematically identifies descriptive and generalizable edge cases and collaboratively improve codebook with human expert. To achieve this, Co-DETECT induces representative edge cases leveraging multi-step clustering and reasoning LLMs. Then the user can supervise the quality of suggested edge cases and decide whether to include them into the codebook or not. Comprehensive user study, and other qualitative and quanititative evaluations prove the effectiveness of Co-DETECT.

8 Limitations and Future Work

While our user study and both qualitative and quantitative analyses demonstrate the effectiveness of Co-DETECT, it also has limitations that we plan to address in future work.

The primary limitation lies in the overreliance on LLM-reported confidence levels, which may introduce significant biases. For example, models may depend on superficial features or spurious correlations learned during pretraining, resulting in unfaithfully high confidence scores and thus the neglect of important edge cases. Moreover, human annotators may rely too heavily on the model's suggestions, potentially overlooking relevant edge cases or alternative interpretations.

To address this, we plan to incorporate sparse autoencoders in future work to provide interpretable features for edge case detection. This will allow users to assess whether a detection is driven by spurious features or by genuine factors that warrant further refinement and specification in the codebook.

Ethics Statement

This research involved voluntary participation in user studies, during which participants provided professional background information and evaluated interface functionality for annotation and edge-case identification tasks. Participants were clearly informed about the study objectives, tasks, and their right to withdraw at any time. Collected data were securely stored, anonymized, and analyzed collectively to ensure confidentiality and privacy. The study posed minimal risks to participants, aligned

with standard professional activities, and adhered closely to ethical guidelines for human-centered research.

Broader Impact Statement

Our system pipeline emphasizes an interactive and iterative approach designed to enhance annotation accuracy and generalization through the systematic management of challenging edge cases. This approach is based on the assumption that improved confidence metrics in a model correlate with enhanced annotation performance. Nevertheless, analogous to the *Clever Hans* phenomenon—where an intelligent system identifies unintended cues instead of genuinely learning underlying knowledge—it is crucial to critically assess the robustness of this pipeline against potential biases and unintended shortcuts that may result from repeated feedback loops and rule-induction processes.

One potential concern involves deriving edge-case rules primarily from model confidence metrics and automatically suggested edge-case instances. If the model's selection and clustering of these edge cases rely predominantly upon internally generated confidence measures, there is a risk that inductively derived rules may reinforce model-specific biases rather than reflect genuinely generalizable conceptual regularities. For example, the model might inadvertently identify clusters based on spurious correlations between input texts and target labels instead of addressing genuine annotation challenges.

Therefore, such risks necessitate increased caution. Systems optimized exclusively within internal frameworks may achieve superficially impressive performance improvements without truly acquiring underlying domain expertise. Consequently, we emphasize the critical importance of human domain expertise. Users should ensure that only rules verified by domain experts are included when updating the codebook.

Acknowledgment

We would like to express our gratitude to *Cong Jiang, Felix Riechmann, Yang Tian, Tingyu Yu,* and *Darya Zare* for their generous support throughout this study.

References

David Autor. 2014. Polanyi's paradox and the shape of employment growth. Technical report, National Bureau of Economic Research.

Aida Mostafazadeh Davani, Mark Díaz, and Vinodkumar Prabhakaran. 2022. Dealing with disagreements: Looking beyond the majority vote in subjective annotations. *Transactions of the Association for Computational Linguistics*, 10:92–110.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4040–4054, Online. Association for Computational Linguistics.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Boyang Li, Shafiq Joty, and Lidong Bing. 2023. Is GPT-3 a good data annotator? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11173–11195, Toronto, Canada. Association for Computational Linguistics.

Jakub Drápal, Hannes Westermann, Jaromir Savelka, et al. 2023. Using large language models to support thematic analysis in empirical legal studies. In *JURIX*, pages 197–206.

Zackary Okun Dunivin. 2024. Scalable qualitative coding with llms: Chain-of-thought reasoning matches human performance in some hermeneutic tasks. *Preprint*, arXiv:2401.15170.

James A Evans and Jacob G Foster. 2011. Metaknowledge. *Science*, 331(6018):721–725.

Yu Fan, Jingwei Ni, Jakob Merane, Etienne Salimbeni, Yang Tian, Yoan Hermstrüwer, Yinya Huang, Mubashara Akhtar, Florian Geering, Oliver Dreyer, Daniel Brunner, Markus Leippold, Mrinmaya Sachan, Alexander Stremitzer, Christoph Engel, Elliott Ash, and Joel Niklaus. 2025a. Lexam: Benchmarking legal reasoning on 340 law exams. *Preprint*, arXiv:2505.12864.

Yu Fan, Yang Tian, Shauli Ravfogel, Mrinmaya Sachan, Elliott Ash, and Alexander Hoyle. 2025b. The medium is not the message: Deconfounding text embeddings via linear concept erasure. *Preprint*, arXiv:2507.01234.

Eve Fleisig, Rediet Abebe, and Dan Klein. 2023. When the majority is wrong: Modeling annotator disagreement for subjective tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6715–6726, Singapore. Association for Computational Linguistics.

Tommaso Fornaciari, Alexandra Uma, Silviu Paun, Barbara Plank, Dirk Hovy, and Massimo Poesio. 2021. Beyond black & white: Leveraging annotator disagreement via soft-label multi-task learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 2591–2597, Online. Association for Computational Linguistics.

Lukas M Fuchs, Yu Fan, and Christian von Scheve. 2021. Value differences between refugees and german citizens: insights from a representative survey. *International Migration*, 59(5):59–81.

Andreas Fügener, Jörn Grahl, Alok Gupta, and Wolfgang Ketter. 2022. Cognitive challenges in human–artificial intelligence collaboration: Investigating the path toward productive delegation. *Information Systems Research*, 33(2):678–696.

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):e2305016120.

Andrew Halterman and Katherine A. Keith. 2025. Codebook llms: Evaluating llms as measurement tools for political science concepts. *Preprint*, arXiv:2407.10747.

Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2024a. AnnoLLM: Making large language models to be better crowdsourced annotators. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 165–190, Mexico City, Mexico. Association for Computational Linguistics.

Zeyu He, Chieh-Yang Huang, Chien-Kuang Cornelia Ding, Shaurya Rohatgi, and Ting-Hao Kenneth Huang. 2024b. If in a crowdsourced data annotation pipeline, a gpt-4. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA. Association for Computing Machinery.

Brendan Kennedy, Mohammad Atari, Aida Mostafazadeh Davani, Leigh Yeh, Ali Omrani, Yehsong Kim, Kris Coombs, Shreya Havaldar, Gwenyth Portillo-Wightman, Elaine Gonzalez, et al. 2018. The gab hate corpus: A collection of 27k posts annotated for hate speech. *PsyArXiv. July*, 18.

Brendan Kennedy, Mohammad Atari, Aida Mostafazadeh Davani, Yehsong Kim, Kris Coombs, Gwenyth Portillo-Wightman, Shreya Havaldar, Elaine Gonzalez, Joseph Hoover, Aida Azatian, Gabriel Cardenas, Alyzeh Hussain, Austin Lara, Adam Omary, Christina Park, Xin Wang, Clarisa Wijaya, Yong Zhang, Beth Meyerowitz, and Morteza Dehghani. 2021. The Gab Hate Corpus: a collection of 27k posts annotated for hate speech. OSF Preprint. 27,665 posts from Gab annotated by multiple annotators; CC-BY 4.0.

Hannah Kim, Kushan Mitra, Rafael Li Chen, Sajjadur Rahman, and Dan Zhang. 2024. MEGAnno+: A human-LLM collaborative annotation system. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 168–176, St. Julians, Malta. Association for Computational Linguistics.

Elisabeth Kirsten, Annalina Buckmann, Leona Lassak, Nele Borgert, Abraham Mhaidli, and Steffen Becker. 2025. From assistance to autonomy – a researcher study on the potential of ai support for qualitative data analysis. *Preprint*, arXiv:2501.19275.

Michelle S. Lam, Janice Teoh, James A. Landay, Jeffrey Heer, and Michael S. Bernstein. 2024. Concept induction: Analyzing unstructured text with high-level concepts using lloom. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–28. ACM.

Josh Levy-Kramer. 2018. k-means-constrained.

Rajiv Movva, Kenny Peng, Nikhil Garg, Jon Kleinberg, and Emma Pierson. 2025. Sparse autoencoders for hypothesis generation. *Preprint*, arXiv:2502.04382.

Jingwei Ni, Yu Fan, Vilém Zouhar, Donya Rooein, Alexander Hoyle, Mrinmaya Sachan, Markus Leippold, Dirk Hovy, and Elliott Ash. 2025a. Can large language models capture human annotator disagreements? *Preprint*, arXiv:2506.19467.

Jingwei Ni, Tobias Schimanski, Meihong Lin, Mrinmaya Sachan, Elliott Ash, and Markus Leippold. 2025b. DIRAS: Efficient LLM annotation of document relevance for retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5238–5258, Albuquerque, New Mexico. Association for Computational Linguistics.

Jingwei Ni, Minjing Shi, Dominik Stammbach, Mrinmaya Sachan, Elliott Ash, and Markus Leippold. 2024. AFaCTA: Assisting the annotation of factual claim detection with reliable LLM annotators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1890–1912, Bangkok, Thailand. Association for Computational Linguistics.

OpenAI. 2024. New embedding models and api updates: text-embedding-3-large. `https://openai.com/index/new-embedding-models-and-api-updates/`. Introduced text-embedding-3-large (3072-dim), offers strongest performance on MIRACL (54.9

OpenAI. 2025a. Introducing gpt-4.1, gpt-4.1 mini & nano. https://openai.com/index/gpt-4-1/. Released April 14, 2025; includes full, mini, and nano variants (1M token context window; optimized for coding and instruction-following).

OpenAI. 2025b. Introducing openai o3 and o4-mini. O3 is a reasoning-focused generative model with advanced capabilities in coding, math, and visual perception; system card provides detailed benchmarks and safety evaluations.

Nicholas Pangakis, Samuel Wolken, and Neil Fasching. 2023. Automated annotation with generative ai requires validation. *ArXiv*, abs/2306.00176.

Chau Minh Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyyer. 2024. TopicGPT: A prompt-based topic modeling framework. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2956–2984, Mexico City, Mexico. Association for Computational Linguistics.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023a. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442, Singapore. Association for Computational Linguistics.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. 2023b. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *Preprint*, arXiv:2305.14975.

Petter Törnberg. 2024. Best practices for text annotation with large language models. *ArXiv*, abs/2402.05129.

Petter Törnberg. 2023. Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning. *Preprint*, arXiv:2304.06588.

Vijay Viswanathan, Kiril Gashteovski, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. 2024. Large language models enable few-shot clustering. *Transactions of the Association for Computational Linguistics*, 12:321–333.

Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024. Human-llm collaborative annotation through effective verification of llm labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA. Association for Computing Machinery.

Zihan Wang, Jingbo Shang, and Ruiqi Zhong. 2023. Goal-driven explainable clustering via language descriptions. *Preprint*, arXiv:2305.13749.

John Wilkerson and Andreu Casas. 2017. Large-scale computerized text analysis in political science: Opportunities and challenges. *Annual Review of Political Science*, 20(1):529–544.

Ziang Xiao, Xingdi Yuan, Q Vera Liao, Rania Abdelghani, and Pierre-Yves Oudeyer. 2023. Supporting qualitative analysis with large language models: Combining codebook with gpt-3 for deductive coding. In *Companion proceedings of the 28th international conference on intelligent user interfaces*, pages 75–78.

Tian Xie and Xueru Zhang. 2024. Automating data annotation under strategic human agents: Risks and potential solutions. *Preprint*, arXiv:2405.08027.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023. ClusterLLM: Large language models as a guide for text clustering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13903–13920, Singapore. Association for Computational Linguistics.

Chengshuai Zhao, Zhen Tan, Chau-Wai Wong, Xinyan Zhao, Tianlong Chen, and Huan Liu. 2025. SCALE: Towards collaborative content analysis in social science with large language model agents and human intervention. *Preprint*, arXiv:2502.10937.

Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. 2023. Goal driven discovery of distributional differences via language descriptions. In *Advances in Neural Information Processing Systems*, volume 36, pages 40204–40237. Curran Associates, Inc.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. Can large language models transform computational social science? *Computational Linguistics*, 50(1):237–291.

A Pre-Interaction Survey Questions

In the pre-interaction survey, we included following questions:

(1) General User Background

- Please briefly describe your professional/academic background and current research areas.
- What is your previous experience with data annotation and qualitative coding? *(None, Beginner, Intermediate, Advanced, Expert)*
- Have you used annotation support or guideline-generation tools previously? *(Yes, No)*

(2) Expectation

- What is your normal workflow of identifying edge cases in text annotation?
- In the dataset that you plan to analyze with AutoDETECT, did you already know any edge cases? *(Yes, No)*

B Post-Interaction Survey Questions

In the post-interaction survey, we included following questions:

(1) Task Completeness, Clarity, and Ease of Use

- Were you clearly able to understand the steps for configuring an annotation task using the interface? *(Five-level Likert item, strongly agree to strongly disagree)*
- Did you encounter any difficulty navigating through different interface components (homepage to analysis dashboard)? *(Yes, No)*
- If yes, please briefly explain your difficulty.

(2) Annotation Results Visualization and Interpretation

- Were you able to clearly interpret and interact with the annotation results displayed in the scatter plots ("All Examples" and "Suggested Edge Cases")? *(Five-level Likert item, strongly agree to strongly disagree)*
- Did interactive features (e.g., clicking points to highlight examples across lists and plots) support your understanding of annotation results effectively? *(Five-level Likert item, supports fully to doesn't support at all)*
- Would you prefer alternative ways of visualizing or interacting with annotation results visually? *(Yes, No)*
- If yes, please describe briefly.

(3) Edge Case Identification and Handling

- Was the component provided by the system clearly identifying relevant and helpful edge cases (cases that require additional annotation guidance) in your corpus? *(Five-level Likert item, strongly agree to strongly disagree)*
- Do the edge cases make sense? *(Five-level Likert item, makes total sense to makes absolutely no sense)*
- Are the proposed rules easy to follow? *(Five-level Likert item, very easy to very difficult)*
- Please describe briefly your reasoning for accepting or rejecting suggested edge-case rules. What information or criteria were the most important for your decisions?

(4) Iterative Optimization Support

- Did you find the iterative approach ("Iterate" button functionality) helpful for progressively refining your annotation guidelines and labels? *(Five-level Likert item, very helpful to not helpful at all)*
- How many iterations (approximately) did you perform? Did subsequent iterations help significantly in clarifying your annotation guidelines? Please briefly explain.

(5) General User Experience and Satisfaction

- Did Co-DETECT help you to find some new edge cases that you didn't notice before? *(Yes, No, Maybe)*
- How satisfied are you overall with the functionality that this tool offers you in developing codebooks and annotation guidelines? *(Five-level Likert item, very satisfied to not satisfied at all)*
- Do you still have concern that e.g. there are missing edge cases not identified by the system? *(Yes, No, Maybe)*
- What features, if any, do you think are missing or need improvement in this tool?

(6) Open-ended Feedback and Improvement Suggestions

- What did you like the most about the user interface and its functionality?
- What improvements or additions would you propose to enhance the usability or functionality of the current interface?
- (Optional) Any additional comments or suggestions about the tool or your experience using it?

DVAGEN: Dynamic Vocabulary Augmented Generation

Wei Du[1]*, Nuowei Liu[1]*, Jie Wang[1]*, Jiahao Kuang[1]*,
Tao Ji[2], Xiaoling Wang[1], Yuanbin Wu[1]†
[1]School of Computer Science and Technology, East China Normal University
[2]School of Computer Science, Fudan University
{weidu.cs, nwliu, jiewang.cs, jhkuang}@stu.ecnu.edu.cn
taoji@fudan.edu.cn, {xlwang, ybwu}@cs.ecnu.edu.cn

Abstract

Language models trained with a fixed vocabulary struggle to generalize to novel or out-of-vocabulary words, limiting their flexibility in handling diverse token combinations. Existing dynamic vocabulary approaches attempt to address this limitation but face challenges such as fragmented codebases, lack of support for modern LLMs, and limited inference scalability. To overcome these issues, we introduce DVAGEN, a fully open-source, unified framework designed for training, evaluation, and visualization of dynamic vocabulary-augmented language models. Our framework modularizes the pipeline for ease of customization, integrates seamlessly with open-source LLMs, and is the first to provide both CLI and WebUI tools for real-time result inspection. We validate the effectiveness of dynamic vocabulary methods on modern LLMs and demonstrate support for batch inference, significantly improving inference throughput.[1]

1 Introduction

Most language models (Grattafiori et al., 2024; Yang et al., 2025) are typically trained on a fixed vocabulary, which constrains their ability to generalize beyond the training corpus. This limitation becomes apparent when encountering novel (out-of-vocabulary) words or when attempting to efficiently generate phrases composed of arbitrary token combinations.

Instead of relying solely on a pre-trained tokenizer with a static vocabulary, recent studies have adopted dynamic vocabulary approaches to augment the generation process, thereby enhancing modeling capabilities for natural languages (Lan et al., 2023; Cao et al., 2024; Liu et al., 2024) and even protein languages (Liu et al., 2025).

However, these prior works rely on diverse codebases, some of which are not yet fully open sourced, and lack a unified framework for training, inference, and visualization of results augmented by dynamic vocabulary. Furthermore, they primarily focus on GPT-2 based language models (Radford et al., 2019; Ferruz et al., 2022), without validation on Large Language Models (LLMs). Additionally, the absence of support for batch inference further increases inference latency, despite the methods demonstrating promising inference speed when processing a single input at a time.

To address these issues, we present DVAGEN and our key contributions are highlighted below.

A fully open-source, unified framework for training, evaluation, and visualization. We propose a unified framework that decouples individual modules, allowing users to customize each component. The framework supports the full training and inference pipeline (`train`, `chat`, and `eval`) for developing dynamic vocabulary-augmented applications. Furthermore, we are the first to offer both command-line tools and a WebUI interface for visualizing generation results.

Integration of existing open-source LLMs and their validation. Existing open-source LLMs can be easily integrated into DVAGEN in a plug-and-play manner. We further evaluate the language modeling performance of LLMs using our framework and validate the effectiveness of employing dynamic vocabulary methods on LLMs.

Supports batch inference to enhance inference throughput. Dynamic vocabulary methods demonstrate promising capabilities in reducing token usage during decoding. Owing to their sequence compression abilities, these methods exhibit low inference latency. Building on this foundation, DVAGEN supports batch inference, further maximizing inference speed.

* Equal Contribution
† Corresponding Author
[1]Codes are publicly available at https://github.com/AntNLP/DVAGen.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 365–372

2 Background

Research on dynamic vocabulary generation addresses key limitations of standard language models that rely on **static tokenization**. Traditional approaches like Byte-Pair Encoding (BPE) and WordPiece employ fixed vocabularies (Radford et al., 2019; Devlin et al., 2019), which struggle to incorporate domain-specific phrases or multi-token expressions during generation. Efforts to enhance static vocabularies, such as Multi-Word Tokenization (MWT) (Gee et al., 2023), expand vocabularies with frequent n-grams but remain inflexible post-training.

Retrieval-augmented methods have emerged as alternatives. Document-level approaches like RETRO (Borgeaud et al., 2022) integrate retrieved passages via cross-attention but incur high latency. The Copy-is-All-You-Need (CoG) framework (Lan et al., 2023) pioneered phrase-level retrieval through a two-stage pipeline: first retrieving relevant documents, then extracting candidate phrases. While CoG supports variable-length generation, it processes phrases as token sequences during input, creating inconsistency between input and output representations. CoG-2 (Cao et al., 2024) advanced this paradigm by enabling direct phrase retrieval without document pre-filtering. Using syntactic heuristics and self-reinforced training, CoG-2 improved knowledge-intensive task performance but maintains the output-only phrase handling limitation of its predecessor.

The **dynamic vocabulary** approach (Liu et al., 2024) constitutes a fundamental architectural shift. It introduces a causal Transformer-based phrase encoder that maps arbitrary text spans to atomic units, unifying their treatment in both input and output layers. This eliminates the token-phrase inconsistency seen in CoG variants. The method operates plug-and-play, requiring no model retraining when incorporating new phrases. The dynamic vocabulary's end-to-end atomic processing provides a more unified solution for fluency, efficiency, and verifiability in text generation.

3 DVAGen

In this section, we introduce the overall architecture of DVAGen, as illustrated in Figure 1.

3.1 DVAModel

The DVA framework integrates a dynamic phrase encoder with a standard language model (LM) through a projection layer, enabling on-the-fly vocabulary expansion. Formally, given a static vocabulary V and a phrase candidate set $P = \{p_1, \ldots, p_m\}$ from PhraseSampler, the model operates as follows:

Phrase Encoding For each phrase $p_i \in P$, tokenize it into subwords $[w_1, \ldots, w_s]$ using the Phrase Encoder's static tokenizer (e.g., GPT-2's BPE), then compute its embedding via the Phrase Encoder:

$$\mathbf{e}_{p_i} = \text{Projector}\left(\text{PhraseEncoder}(\mathbf{w}_{1:s})_s\right)$$

where $\mathbf{w}_{1:s}$ denotes subword embeddings of p_i, and the last token's hidden state is taken as the vector representation of p_i. A Projector then maps this to the LM's embedding space. Typically, the Phrase Encoder is a causal Transformer model, and the Projector is an MLP.

Vocabulary Expansion Concatenate phrase embeddings with LM's original input/output embedding matrices:

$$\mathbf{E}'_{\text{in}} = [\mathbf{E}_{\text{in}}, \mathbf{E}_P], \quad \mathbf{E}'_{\text{out}} = [\mathbf{E}_{\text{out}}, \mathbf{E}_P]$$

where $\mathbf{E}_P = [\mathbf{e}_{p_1}, \ldots, \mathbf{e}_{p_m}] \in \mathbb{R}^{d \times m}$. The expansion is dynamic, that is for each sentence, we can attach different $\mathbf{E}_P$ for generation.

Generation The DVAModel then generate on the expanded dynamic vocabulary. At step t, we compute logits on both token vocabulary V and phrase candidates P, and choose the output $y_t \in V \cup P$ accordingly:

$$y_t \sim \text{Softmax}(h_t \mathbf{E}'_{\text{out}}) \in \mathbb{R}^{|V|+|P|}$$

where h_t is the last hidden state of the LM. We note that DVAGen allows each sentence in a batch to have a different set of phrase candidates, in which case a phrase mask is incorporated in DVA Logits Processor to filter out a specific set in $\mathbf{E}_P$ for that sentence.

3.2 PhraseSampler

The PhraseSampler $\mathcal{S}$ extracts candidate phrases P from retrieved documents D using configurable strategies. DVAGen provides a set of pre-defined strategies:

- *NToken*: Samples n-gram token sequences from tokenized text.

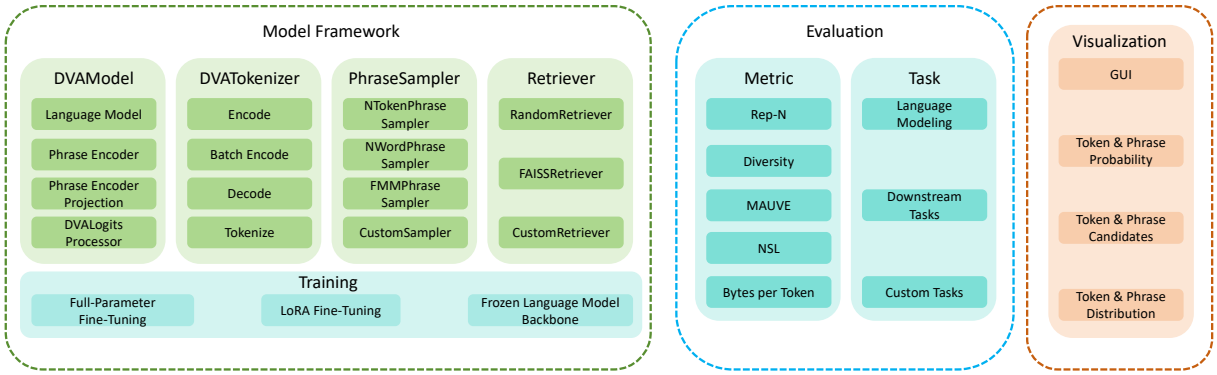


Figure 1: The overall architecture of DVAGEN. The light color boxes represent modules, while the dashed boxes indicate collections of modules with relevant functions.

- *NWord*: Extracts word-level n-grams via word tokenization (e.g., using NLTK).
- *FMM* (Forward Maximum Matching): Suppose the current position in the text is i from a corpus $\mathcal{C}$, then a segment $\mathcal{C}[i : i + m]$ is identified such that it appears in at least one sentence of the corpus $\mathcal{C}$, and m is the maximum value satisfying this condition. This segment will be extracted as a phrase, and the process continues from position $i + m$.

Custom samplers can be integrated by implementing the interface $P \leftarrow \mathcal{S}(D, \text{config})$.

3.3 DVATokenizer

Denote input text as $X = x_1, x_2, ..., x_L$ where L is the length of input text. There are three main function in DVATokenizer.

Tokenize This function can split text to phrases and tokens according to P:

$$\text{Tokenize}(X, P) = \{x'_1, x'_2, ..., x'_{L'}\}, L' \leq L$$

where $x'_i \in P \cup X$ is original token or candidate phrase.

Encode It can encode input text to input ids.

$$\text{Encode}(X, P) = \{\text{id}_1, \text{id}_2, ..., \text{id}_{L'}\}, L' \leq L$$

$$\text{type}(\text{id}_i) = \begin{cases} \text{token}, & \text{if id}_i < |V| \\ \text{phrase}, & \text{otherwise} \end{cases}$$

If the id_i's value exceeds the vocabulary size $|V|$, it is a token ID; otherwise, it is a phrase ID.

Decode This function is the inverse function of encode which can get the original text from the mixed ids.

$$\text{Decode}(\text{Encode}(\text{X}, \text{P})) = X$$

3.4 Training

DVAGEN supports full-parameter fine-tuning, LoRA fine-tuning (Hu et al., 2022), and training with a frozen language model backbone. Different training strategies share a similar processing paradigm, as illustrated in Figure 2. Given a training dataset D, we first sample a batch containing m training samples. A PhraseSampler is then applied to sample phrases from the batch. With an arbitrary combination of tokens and phrases within each training sample, we employ a DVATokenizer to encode them into discrete token and phrase ids for subsequent training.

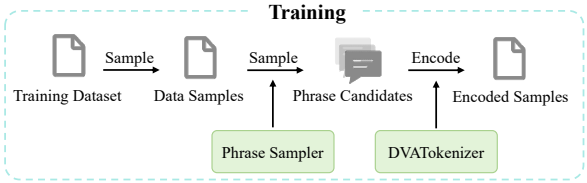


Figure 2: The overall training pipline of DVAGEN.

3.5 Inference and Evaluation

Inference The inference paradigm differs from the training process by incorporating the retrieval of the top k most relevant supporting documents. The overall inference pipeline is illustrated in Figure 3. Given an input prefix, we employ a Retriever implemented using an embedding model with a FAISS backend (Douze et al., 2024) to enable efficient similarity search on dense vectors. Building on top of LangChain[2], we utilize the GPU version of FAISS (Johnson et al., 2019) to accelerate the retrieval process. The retrieved supporting documents are then used to sample phrases using

[2] https://github.com/langchain-ai/langchain

a PhraseSampler, and the sampled phrases serve as candidates for constructing the dynamic vocabulary. It is worth noting that for different input prefixes, the phrase candidates vary according to the retrieved supporting documents. Unlike previous studies (Lan et al., 2023; Liu et al., 2024) that do not support batch inference, we implement the DVALogitsProcessor to mask irrelevant logits (i.e., those corresponding to phrase ids associated with phrases unrelated to the current input). The positions to be masked are determined based on phrase candidates from other inputs within the same batch.

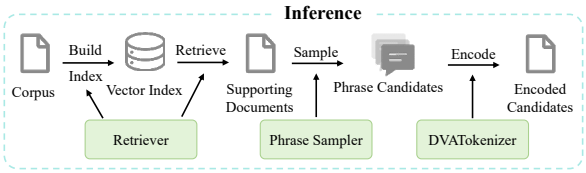


Figure 3: The overall inference pipline of DVAGEN.

Evaluation Evaluation across various tasks are supported in DVAGEN, including basic language modeling and a range of downstream tasks. To ensure ease of use and fair comparisons, our framework includes implementations of a wide array of evaluation metrics. These metrics are categorized into three types: (1) generation quality (e.g., MAUVE (Pillutla et al., 2021), Rep-N and Diversity (Welleck et al., 2019), and Perplexity), (2) sequence compression (e.g., Normalized Sequence Length and Bytes per Token (Dagan et al., 2024)), and (3) downstream task performance (e.g., Recall, Precision, F1, and ROUGE-L (Lin, 2004)).

3.6 Visualization

Our web application is developed using the Django[3], featuring a front-end interface built with Bulma[4] to provide a clean and responsive user experience, and DVAGEN handles all text generation tasks in backend.

Figure 4 presents the input area of the front-end interface. The brown input box allows users to enter an initial query or prefix text, while the yellow phrase input area beneath it is specifically designed for defining dynamic vocabulary—users can specify phrases that the model should prioritize during generation. After completing the inputs, users can click the "Generate" button to initiate the generation process.

Figure 5 displays the output area of the interface after generation is complete. The generated text appears in the grey output box. By clicking on any token or phrase in the box, users can view a scrollable list of candidate alternatives for that position, along with their associated probabilities. Users may then select a preferred alternative to replace the original token or phrase, thereby constructing a new prefix for controlled generation. To support more targeted interaction, the interface also offers filter options such as "Phrases", "Tokens & Phrases" and "Tokens", enabling users to refine the candidate list by category and focus on the specific categoriy of alternatives.

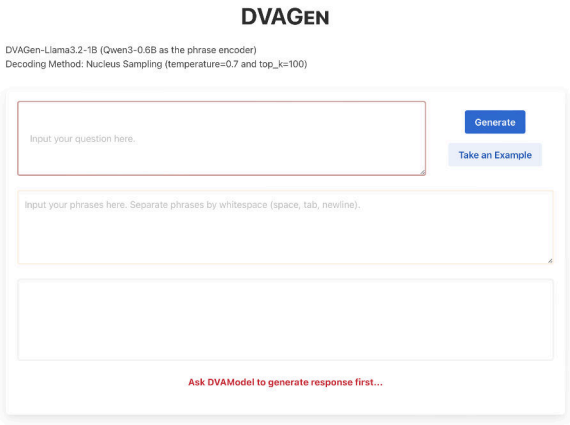


Figure 4: The screenshot of the DVAGEN WebUI interface.

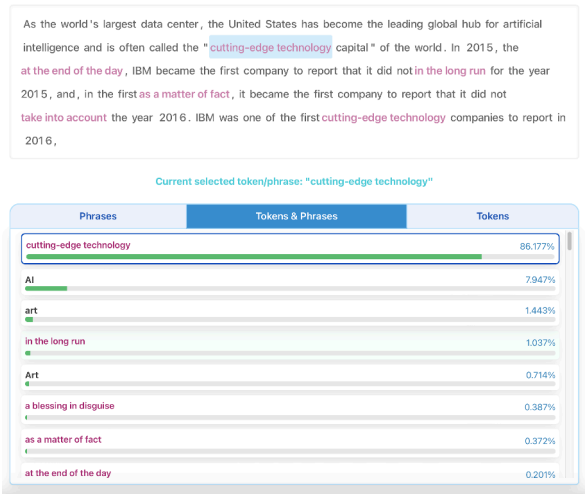


Figure 5: The screenshot displays the generation outputs. Phrases are highlighted in red, and the logit probabilities for each token and phrase can be viewed by clicking on the respective token or phrase.

[3] https://www.djangoproject.com/
[4] https://bulma.io/

To intuitively present the complex outputs generated by our model, we have developed a powerful visualization application.

Figure 6 illustrates the visualization generated by the application, which accurately captures the model's interpretation of each token and phrase within the generated text. Tokens and phrases are distinguished using distinct color schemes—for instance, blue represents tokens, while purple-red denotes phrases. The shade of each color reflects the corresponding generation probability, with deeper shades indicating higher probabilities. A heatmap positioned at the top of the visualization clearly maps probability values to color intensities. Additionally, the application offers extensive customization options, enabling users to adjust token and phrase colors, font sizes, and image resolution to suit their specific analytical or presentation requirements.

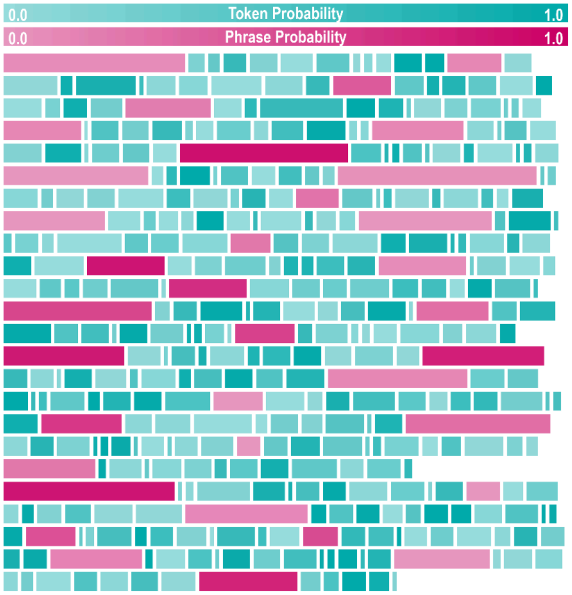


Figure 6: Visualization of generation outputs is represented in blue for token distribution and red for phrase distribution. The brightness of the colors reflects the logit probabilities, with darker colors indicating higher confidence.

4 Evaluation

4.1 Language Modeling

Setup We first show that DVAGEN can enhance the basic language modeling capabilities of LLMs. To evaluate this, we select the pre-trained base models Qwen3-0.6B[5], Qwen3-1.7B[6], Llama3.2-1B[7], and Llama3.2-3B[8]. These models are fine-tuned on the WikiText-103 training set to serve as baselines. To ensure a fair comparison across different model families, a GPT-2 (Radford et al., 2019) tokenizer is employed to extract the first 32 tokens from each test sample as input. During inference with DVAGEN, we utilize the NTokenPhraseSampler to sample phrases from the given input prefixes. Qwen3-Embedding-0.6B[9] is used as the embedding model for Retriever, and 32 documents are retrieved for each sample.

Results As shown in Table 1, our key findings can be summarized as follows:

(1) DVAGEN improves generation quality while requiring fewer tokens to generate the same content. DVAGEN helps maintain a lower Rep-N score and higher diversity, particularly for Llama models. The improvement in the MAUVE score indicates the generation of more natural and coherent language that closely matches human-written text. More importantly, training with DVAGEN significantly enhances sequence compression capability, demonstrating improved efficiency in reducing both tokens and decoding steps.

(2) Freezing the language model during training is memory-efficient and achieves performance on par with that of a trainable backbone. The results highlight that training with a frozen language model backbone yields performance on par with a trainable backbone while significantly reducing memory requirements. The evaluations offer insights for future research aimed at developing memory-efficient methods and reducing model size to enable more efficient training and inference.

4.2 Inference Performance

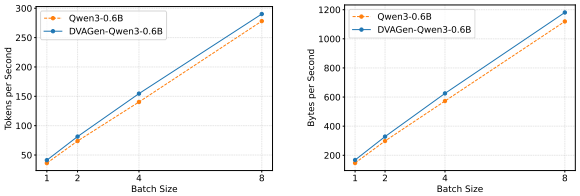
Setup To evaluate the generation performance of DVAGEN, we compare the tokens and UTF-8 bytes generated per second by Qwen3-0.6B and DVAGEN-Qwen3-0.6B. For a fair comparison, both models are evaluated on the WikiText-103 test set under identical settings. Each sample in a batch consists of 32 tokens as input prefix, and

[5] https://huggingface.co/Qwen/Qwen3-0.6B-Base
[6] https://huggingface.co/Qwen/Qwen3-1.7B-Base
[7] https://huggingface.co/meta-llama/Llama-3.2-1B
[8] https://huggingface.co/meta-llama/Llama-3.2-3B
[9] https://huggingface.co/Qwen/Qwen3-Embedding-0.6B

Models	MAUVE ↑	Rep-2 ↓	Rep-3 ↓	Rep-4 ↓	Diversity ↑	PPL ↓	NSL ↓	Bytes per Token ↑
Qwen3-0.6B	21.70	11.46	5.07	3.05	81.49	51.09	1.00	3.17
DVAGEN-Qwen3-0.6B ❄️	24.31	19.68	10.92	7.59	66.12	22.29	0.87	4.84
DVAGEN-Qwen3-0.6B 🔥	24.41	14.81	6.46	3.61	76.82	24.19	0.85	4.89
Qwen3-1.7B	22.74	10.21	4.14	2.19	84.18	57.55	1.00	3.27
DVAGEN-Qwen3-1.7B ❄️	24.92	18.64	10.26	7.21	67.76	21.73	0.86	4.88
DVAGEN-Qwen3-1.7B 🔥	23.57	16.05	7.88	4.99	73.47	19.93	0.99	4.63
Llama3.2-1B	24.74	17.28	8.73	5.29	71.50	28.65	1.00	4.00
DVAGEN-Llama3.2-1B 🔥	24.64	15.70	7.41	4.46	74.57	81.66	0.88	4.90
Llama3.2-3B	25.61	15.62	7.64	4.58	74.35	63.04	1.00	4.01
DVAGEN-Llama3.2-3B ❄️	25.26	16.85	8.50	5.41	71.96	43.98	0.87	4.97
DVAGEN-Llama3.2-3B 🔥	26.40	14.30	6.49	3.79	77.09	37.19	0.93	4.90

Table 1: Evaluation results on the WikiText-103 test set. Qwen3-0.6B is used as the phrase encoder and remains trainable across all settings. The 🔥 and ❄️ markers indicate whether the language model backbone is trainable or frozen, respectively.

the models are required to generate exactly 128 new tokens (or phrases when using DVAGEN) by setting both the minimum and maximum number of generated tokens to the same value. Note that, in alignment with previous work (Lan et al., 2023; Liu et al., 2024), the retrieval cost is not included, as the construction of phrase candidates can be conducted offline. The evaluations are conducted using a single NVIDIA RTX 4090 GPU.



(a) (b)

Figure 7: Evaluation of performance in generating tokens and UTF-8 encoded bytes per second.

Results The evaluation results are presented in Figure 7, revealing two main findings:

(1) Compared to the base model, DVAGEN achieves higher inference speed despite having more parameters. DVAGEN-Qwen3-0.6B employs two Qwen3-0.6B models, serving as the language model and phrase encoder, respectively, resulting in a model twice the size of the base model. However, due to phrases typically consisting of multiple tokens and containing more UTF-8 bytes, DVAGEN achieves faster inference than the base model under the same settings.

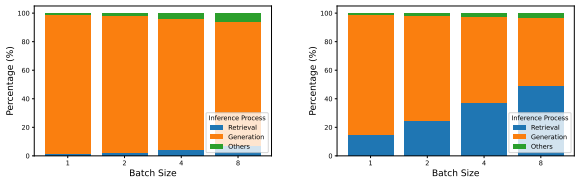
(2) DVAGEN supports batch inference and consistently maintains high inference speed as the batch size increases. The implementation of batch inference improves generation efficiency by approximately 7× compared to processing one input at a time, as in the original implementations (Lan et al., 2023; Liu et al., 2024). Furthermore, the consistent inference performance with larger batch sizes demonstrates that DVAGEN effectively scales with workload, increases throughput, and optimizes GPU memory utilization without incurring additional latency.

4.3 Retrieval Performance

Setup In addition to the inference performance (excluding the retrieval process) discussed in Section 4.2, we also conduct experiments to compare retrieval latency during inference. The experimental setup is similar to the previous one, except that a single NVIDIA RTX 3090 GPU is used. Since many real-time applications require longer contexts and the Retriever requires substantial GPU memory during inference, we also evaluate its retrieval performance on CPU devices. This experiment is conducted on a server equipped with an Intel(R) Xeon(R) Silver 4214R CPU @ 2.40 GHz and 512 GB of RAM.

Results The comparisons of the proportions of different stages (primarily retrieval and generation) during inference are shown in Figure 8. The key observations are as follows:

(1) When a GPU is utilized for inference, the generation stage consistently dominates the overall inference time, substantially exceeding the



(a) (b)

Figure 8: Evaluation of retrieval performance: (a) retrieval performed on a GPU device, (b) retrieval performed on a CPU device.

time spent on retrieval. As discussed in Section 3.5, DVAGEN retrieves relevant supporting documents on the GPU by default, with the retrieval process accounting for approximately 7% of the inference time when the batch size is set to 8. In addition, since retrieval within a batch is performed sequentially due to the independence of samples, retrieval time increases with larger batch sizes.

(2) Compared with GPU-based retrieval, CPU-based retrieval requires substantially more time. When the batch size is set to 8, CPU-based retrieval accounts for approximately half of the total inference time, slightly exceeding the duration of the generation stage (performed on a GPU device). This observation suggests that when using CPU-based retrieval to save GPU memory, the trade-off between batch size and inference latency should be carefully considered.

5 Conclusion

In this work, we present a fully open-source, modular framework that unifies training, evaluation, and visualization for dynamic vocabulary-augmented applications. The framework enables seamless customization of individual components and is the first to provide both CLI and WebUI tools for generation analysis. By supporting plug-and-play integration of existing open-source LLMs, we demonstrate the effectiveness of dynamic vocabulary methods in enhancing language modeling performance. Furthermore, with support for batch inference, our framework significantly improves inference throughput, offering a practical and scalable solution for deploying efficient LLM-based systems.

Acknowledgments

The authors wish to thank all reviewers for their helpful comments and suggestions. The corresponding author is Yuanbin Wu. This research was (partially) supported by National Key R&D Program of China (2024YFC3308103).

References

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. *Preprint*, arXiv:2112.04426.

Bowen Cao, Deng Cai, Leyang Cui, Xuxin Cheng, Wei Bi, Yuexian Zou, and Shuming Shi. 2024. Retrieval is accurate generation. *arXiv preprint arXiv:2402.17532*.

Gautier Dagan, Gabriel Synnaeve, and Baptiste Roziere. 2024. Getting the most out of your tokenizer for pre-training and domain adaptation. *arXiv preprint arXiv:2402.01035*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Noelia Ferruz, Steffen Schmidt, and Birte Höcker. 2022. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348.

Leonidas Gee, Leonardo Rigutini, Marco Ernandes, and Andrea Zugarini. 2023. Multi-word tokenization for sequence compression. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 612–621, Singapore. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Tian Lan, Deng Cai, Yan Wang, Heyan Huang, and Xian-Ling Mao. 2023. Copy is all you need. In *The Eleventh International Conference on Learning Representations*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Nuowei Liu, Jiahao Kuang, Yanting Liu, Changzhi Sun, Tao Ji, Yuanbin Wu, and Man Lan. 2025. Protein design with dynamic protein vocabulary. *arXiv preprint arXiv:2505.18966*.

Yanting Liu, Tao Ji, Changzhi Sun, Yuanbin Wu, and Xiaoling Wang. 2024. Generation with dynamic vocabulary. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18931–18948, Miami, Florida, USA. Association for Computational Linguistics.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

MCPEval: Automatic MCP-based Deep Evaluation for AI Agent Models

Zhiwei Liu, Jielin Qiu, Shiyu Wang, Jianguo Zhang, Zuxin Liu, Roshan Ram, Haolin Chen, Weiran Yao, Shelby Heinecke, Silvio Savarese, Huan Wang* and **Caiming Xiong***

Salesforce AI Research
*Corresponding Authors: {huan.wang, cxiong}@salesforce.com

Abstract

The rapid rise of Large Language Models (LLMs)-based intelligent agents underscores the need for robust, scalable evaluation frameworks. Existing methods rely on static benchmarks and labor-intensive data collection, limiting practical assessment. We introduce *MCPEval*, an open-source Model Context Protocol (MCP)-based framework that automates end-to-end task generation and deep evaluation of LLM agents across diverse domains. *MCPEval* standardizes metrics, seamlessly integrates with native agent tools, and eliminates manual effort in building evaluation pipelines. Empirical results across five real-world domains show its effectiveness in revealing nuanced, domain-specific performance. We publicly release *MCPEval*[1] to promote reproducible and standardized LLM agent evaluation.

Keywords: Agent System, Model Context Protocol, Automated Testing

1 Introduction

The rapid proliferation of Large Language Models (LLMs) has catalyzed a transformative shift in artificial intelligence, giving rise to autonomous agents capable of sophisticated reasoning, planning, and executing complex tasks (OpenAI, 2023; Liu et al., 2024b,a; Zhang et al., 2024b; Ma et al., 2024b). These systems, defined by their ability to autonomously conceive and adapt plans while interacting with dynamic environments through external tools, represent a new paradigm in AI (Ouyang et al., 2022; Zhang et al., 2023a). Consequently, there is an urgent and growing focus on developing comprehensive and systematic evaluation frameworks to standardize the assessment (Zhu et al., 2024; Zheng et al., 2024; Ji et al., 2024), thereby enabling the quick deployment and adaptation of these powerful systems.

Current evaluation methods face significant limitations. Early benchmarks, while foundational, often relied on static, predefined tasks, thus failing to capture the interactive real-world agentic workflows (Liu et al., 2023a,b; Fan et al., 2024). While the field has progressed toward dynamic and interactive benchmarks (Qin et al., 2023; Zheng et al., 2023), a persistent challenge remains: the lack of deep integration with practical tools and the absence of standardized protocols for agent-environment communication, which impedes reproducibility and robust comparison (Qin et al., 2023; Deng et al., 2023; Huang et al., 2024).

Addressing these critical gaps, we introduce *MCPEval*, an automatic deep evaluation system for AI agents built upon the Model Context Protocol (MCP)[2]. The recent emergence of MCP as a standard for governing interactions between LLMs and external systems offers a foundational layer for building scalable and interactive agents (Anthropic, 2024; Allganize, 2024; Lumer et al., 2025). This standardization opens the quick development of new evaluation frameworks like MCP-Radar (Gao et al., 2025) and MCPWorld (Yan et al., 2025).

Nevertheless, these approaches face significant scalability limitations, either requiring manual task creation and relies on labor intensive human evaluations for complex scenarios, or evaluated on static benchmark completion without comprehensive analysis. *MCPEval* advances beyond these systems by introducing a fully automated evaluation process. *MCPEval* not only overcomes the manual bottlenecks and scalability issues in collecting tasks, but also provides a deep analysis workflow to evaluate models. As a result, our system can rapidly evaluate model's interactive capabilities towards new MCP tools and servers at scale, providing developers with actionable feedback to optimize their implementations. The user interface

[1] https://github.com/SalesforceAIResearch/MCPEval

[2] https://modelcontextprotocol.io/introduction

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 373–402
November 4-9, 2025

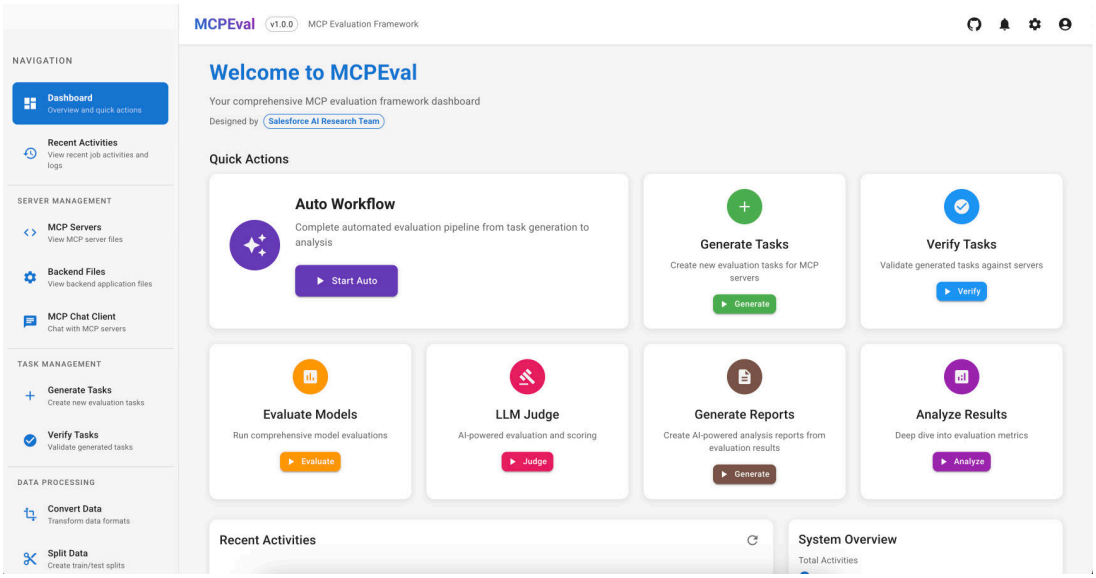


Figure 1: User interface of the MCPEval framework. The dashboard provides streamlined access to core functionalities such as automatic task generation, verification, model evaluation, and result analysis. It integrates real-time activity tracking and system overviews to ensure transparency and ease of use.

design of *MCPEval* is illustrated in Figure 1.

MCPEval goes beyond traditional success/failure metrics by systematically collecting detailed task trajectories and protocol interaction data, creating unprecedented visibility into agent behavior and generating valuable datasets for iterative improvement. Additionally, because both task creation and verification are fully automated, the resulting high-quality trajectories can be immediately leveraged for rapid fine-tuning and continual improvement of agent models. The comprehensive evaluation reports generated by *MCPEval* also provide actionable insights towards the correctness of agent-platform communication at a granular level.

Through extensive experiments involving state-of-the-art LLMs and five real-world benchmarks, *MCPEval* reveals scenarios where smaller, tool-enhanced models perform comparably to larger, more resource-intensive models. These findings highlight significant opportunities for cost-effective deployments without compromising performance. *MCPEval* is made available as an open-source platform to support reproducibility, foster robust evaluation practices, and accelerate practical advancements in the broader LLM research community.

2 Related Work

This section reviews the shift in LLM evaluation from static benchmarks to dynamic, agent-based methods, highlighting frameworks like the Model Context Protocol (MCP) and the role of synthetic data in creating new test scenarios. More comprehensive related work discussion can be found in Appendix D.

The Evolution of LLM & Agent Evaluation Frameworks LLM evaluation has evolved from static benchmarks like HELM (Liang et al., 2022), BIG-bench (Srivastava et al., 2022), and MMLU (Hendrycks et al., 2020) to more interactive methods. To address data contamination, newer benchmarks employ strategies like temporal cutoffs, e.g., LiveBench (White et al., 2024)) and LLM-based generation, e.g. MMLU-Pro (Zhang et al., 2024a). Recognizing the limitations of single-turn formats, conversational and agentic benchmarks emerged, such as MT-Bench (Zheng et al., 2023), AgentBoard (Ma et al., 2024a), and AgentBench (Liu et al., 2023a), which focus on multi-turn agent performance. However, many still lack deep tool integration, prompting the development of domain-specific, task-oriented evaluations, such as WebArena (Xie et al., 2024) and others (Zhou et al., 2023; Liu et al., 2023b; Kokane et al., 2024; Koh et al., 2024; Jimenez et al., 2023; Geng and Chang, 2025; Zhang et al., 2025).

Evaluating Agents with Deep System Integration As agents increasingly operate in realistic digital environments, evaluation has shifted toward measuring system-level interaction. Platforms like OSWorld (Xie et al., 2024) assess GUI-based task execution, while frameworks like

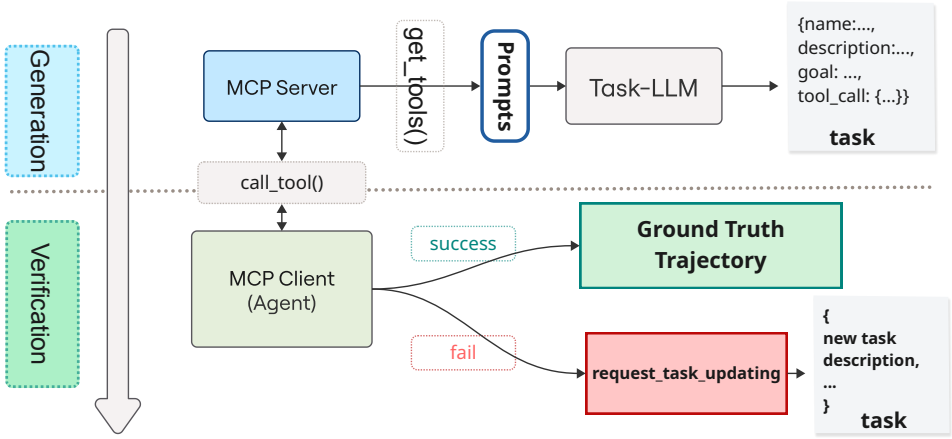


Figure 2: Two-step MCP-based task generation workflow, including initial generation phase and verification phase.

LangChain (Chase et al., 2022), AutoGen (Wu et al., 2023), and CrewAI (Roveda et al., 2023) highlight the lack of robust evaluation tools. The MCP (Lumer et al., 2025; Allganize, 2024) has emerged as a key standard for LLM-tool communication, enabling scalable, protocol-based evaluation. Frameworks like MCP-Radar (Gao et al., 2025) and MCPWorld (Yan et al., 2025) assess tool-use effectiveness and task completion. Our work builds on this by offering a finer-grained analysis of agent-protocol interaction fidelity.

Synthetic Data Generation Recent work leverages LLMs to generate evaluation data, evolving from simple instruction-response pairs (Zhang et al., 2023b; Wang et al., 2023; Xu et al., 2023; Tan et al., 2025) to rich and interactive scenarios (Ma et al., 2024b; Tang et al., 2024; Liu et al., 2024c). Execution feedback enables closed-loop systems that automatically verify task correctness, as seen in AgentEval (Arabzadeh et al., 2024) and LAM-Simulator (Hoang et al., 2025). Our approach also uses synthetic data to evaluate agent interactions under MCP, generating tool-use tasks.

3 *MCPEval* Framework

MCPEval adopts a evaluation workflow designed with task generation, task verification and model evaluation, facilitating efficient and extensible evaluation of LLM agents. The task generation begins at the calling MCP server with tool call method to collect the specifications of tools into prompts. We demonstrate this process in Figure 2.

Upon receiving those context, a Task-LLM generates detailed task instructions, ensuring the appropriate information for tool calls are included in the task instruction. However, we observe that these initial generated tasks may not include all necessary information to fill the tool parameters. Therefore, a task verification is necessary to generate high-quality task and collect the ground truth trajectory, which is illustrated in Figure 2.

During verification stage, we employ a frontier agent as MCP client and interact the MCP server. This frontier agent executes the generated tasks with actual tool calls. Successful execution trajectory leads directly to verified tasks and corresponding ground truth. In cases of task execution failure, the agent initiates a task updating request, prompting the generation of refined task descriptions. This iterative verification and refinement process ensures high-quality tasks suitable for comprehensive evaluation.

For model evaluation in Figure 3, *MCPEval* systematically assesses LLM agent models by placing the model-under-test as the MCP client and requiring it to complete the set of verified tasks. The framework then analyzes collected trajectories using two complementary perspectives: 1) Tool Call Matching, which rigorously compares the model's tool usage against ground truth reference trajectories, and 2) LLM Judging, which covers dimensions such as planning, execution flow, context awareness and etc.

Combining these analytic results, *MCPEval* automatically generates comprehensive reports detailing each agent model's strengths, weaknesses, and performance across multiple domains. This fully automated workflow eliminates manual data collection and evaluation bottlenecks, enabling large-scale, reproducible agent assessment.

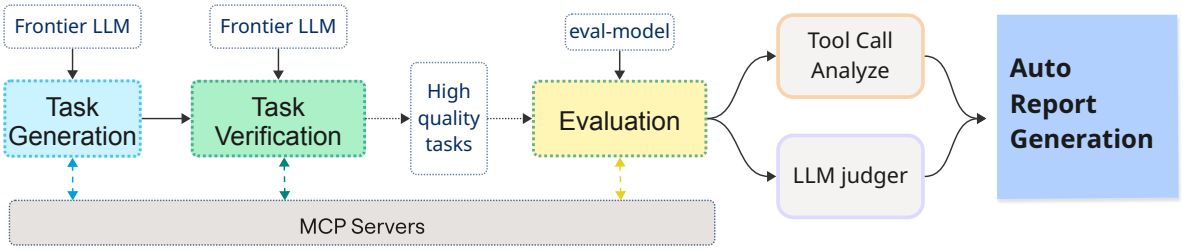


Figure 3: *MCPEval* evaluation workflow shows MCP client/server interaction, tool call correctness checking, LLM judger assessment, automated report generation.

4 Experiment

Model Selection Our evaluation includes ten models spanning different architectures and capabilities, including seven `OpenAI Models`: GPT-4o, GPT-4o-mini, GPT-4.1-mini, GPT-4.1-nano, O3, O3-mini, O4-mini, and three `Open-Source Models`: Mistral-Small-24B, Qwen3-32B, Qwen3-30B-A3B. Model versions are listed in Table 4.

Domain Coverage We evaluate performance across five application domains: (1) `Healthcare`: Medical terminology lookup, drug information, clinical trials, health topics, PubMed search; (2) `Airbnb`: Property search, listing details, booking information (3) `Sports`: Team statistics, player information, game schedules, league standings; (4) `National Parks`: Park information, visitor services, trail details, facility booking; (5) `YFinance`: Stock prices, financial data, market analysis, portfolio management.

Evaluation Criteria Detailed explanation of those evaluation criteria are in Appendix B. Our multi-level evaluation approach combines both tool call analyses and llm judger analyses over 676 tasks for all ten models: (1) `Tool Call Analysis`: direct scoring of tool call matching, including parameter matching score, tool name matching score and tool order matching score. A overall matching score is calculated as a weighted combination of all. We mainly report the overall scores in this paper. We calculate both the strict and flexible matching scores. Strict matching is to ensure generated tool calls trajectory is exact the same as ground truth. Flexible matching tolerant similar parameters and tool call orders is not important. (2) `LLM Judger Analysis`: Systematic evaluation of 50 model-domain combinations using frontier LLM as an expert judge. The judge rubrics are based on both trajectory-wise and final response completion-wise.

5 Results and Discussions

In this section, we introduce part of the experimental results and analysis from *MCPEval*. Appendix A includes more results and discussions.

5.1 Cross-Domain Model Comparison

We includes cross-domain model comparison results as in Table 1 and Table 2, which respectively from tool-call criteria and LLM judge criteria.

Regarding the tool-call performance, Table 1 shows that GPT-4 variants, especially `gpt4o`, consistently lead across domains. Open models display mixed performance. Notably, smaller models like `o4-mini` generalize well, outperforming larger open models in certain tasks. These findings highlight the value of *MCPEval* in revealing nuanced, domain-specific strengths and weaknesses that static benchmarks may overlook.

Beyond binary success rates, Table 2 uses LLM-Judge to assess interaction quality through trajectory (reasoning) and completion (task alignment) scores. GPT-4 models again lead, with `o3` excelling in Completion and `gpt4o-mini` and `gpt4.1-mini` showing strong Trajectory performance. Open models perform inconsistently, where some succeed in domains like Healthcare but falter in reasoning stability. Notably, `o4-mini` often recovers from weak reasoning to deliver strong final outputs. These results show how *MCPEval* paired with LLM-based evaluation reveals deeper insights into agent behavior across tasks.

5.2 Fine-grained Criteria Comparison

Besides the cross-domain comparison, we also demonstrate the performance comparison of different models w.r.t. part of those fine-grained criteria aspects, including *Name Match*, *Parameter Match*, *Order Match*, *Planning Execution*, *Requirements Coverage*, and *Completeness* on both tool call metrics and LLM Judger Metrics.

Table 1: Tool-call evaluation results: Strict and Flex tool matching scores across all MCP domains.

Model	Finance		Airbnb		Healthcare		Sports		National Parks		Average	
	Strict	Flex	Strict	Flex	Strict	Flex	Strict	Flex	Strict	Flex	Strict	Flex
mistral-small-24b	42.0%	43.0%	65.3%	72.3%	82.0%	87.0%	64.0%	66.0%	52.0%	62.0%	61.1%	66.1%
qwen3-30b-a3b	38.0%	38.0%	63.0%	69.0%	80.0%	85.0%	72.0%	74.0%	50.0%	57.0%	60.6%	64.6%
qwen3-32b	51.0%	52.0%	63.0%	70.0%	82.0%	87.0%	70.0%	72.0%	54.0%	63.0%	64.0%	68.8%
gpt4.1-mini	92.0%	93.0%	75.0%	**82.0%**	**89.0%**	**93.0%**	82.0%	84.0%	59.0%	66.0%	79.4%	83.6%
gpt4.1-nano	86.0%	88.0%	64.4%	68.7%	70.0%	75.0%	73.3%	75.0%	50.0%	59.0%	68.7%	73.1%
gpt4o-mini	92.0%	**94.0%**	71.5%	77.0%	86.7%	90.6%	77.0%	79.0%	58.0%	68.0%	77.0%	81.7%
gpt4o	**93.0%**	93.5%	**77.0%**	81.0%	87.0%	92.0%	**80.0%**	**82.0%**	**64.0%**	**73.0%**	**80.2%**	**84.3%**
o3-mini	83.0%	84.0%	57.5%	66.1%	28.0%	30.0%	61.0%	62.0%	51.0%	59.0%	56.1%	60.2%
o4-mini	79.0%	80.0%	61.5%	71.7%	67.0%	72.0%	73.0%	75.0%	50.0%	58.0%	66.1%	71.3%
o3	86.0%	87.0%	66.0%	79.0%	62.0%	69.0%	68.0%	70.0%	44.0%	54.0%	65.2%	71.8%

Table 2: LLM-Judge evaluation results: Trajectory (Traj) and Completion (Comp) across all MCP domains.

Model	Finance		Airbnb		Healthcare		Sports		National Parks		Average	
	Traj	Comp	Traj	Comp	Traj	Comp	Traj	Comp	Traj	Comp	Traj	Comp
mistral-small-24b	47.9%	37.3%	88.0%	85.5%	90.9%	83.5%	66.0%	70.3%	86.0%	55.5%	75.8%	66.4%
qwen3-30b-a3b	52.6%	33.9%	81.0%	71.2%	92.3%	83.7%	70.9%	58.7%	79.2%	59.9%	75.2%	61.5%
qwen3-32b	61.8%	45.1%	86.5%	83.8%	90.8%	84.3%	72.6%	61.0%	79.6%	62.8%	78.3%	67.4%
gpt4.1-mini	97.4%	92.2%	90.3%	79.8%	92.1%	78.1%	79.0%	67.9%	77.1%	74.3%	87.2%	78.5%
gpt4.1-nano	**97.7%**	86.3%	85.6%	68.7%	90.9%	80.6%	75.2%	54.7%	61.9%	52.7%	82.3%	68.6%
gpt4o-mini	96.7%	92.4%	90.2%	82.4%	**92.6%**	78.2%	80.1%	68.7%	83.0%	77.0%	88.5%	79.7%
gpt4o	97.0%	92.3%	91.1%	75.3%	92.4%	79.9%	80.8%	68.7%	**90.2%**	87.1%	**90.3%**	80.7%
o3-mini	63.6%	60.6%	91.4%	92.4%	77.9%	54.1%	86.8%	80.0%	90.1%	**90.9%**	82.0%	75.6%
o4-mini	37.2%	36.8%	91.6%	93.6%	85.8%	92.3%	**88.4%**	88.2%	81.3%	86.6%	76.9%	79.5%
o3	95.9%	**97.1%**	**92.3%**	**97.4%**	83.6%	**94.8%**	85.5%	**94.5%**	80.7%	90.8%	87.6%	**94.9%**

Looking at the evaluation results from our *MCPEval* framework, several key patterns emerge that must be interpreted within the context of the evaluation methodology. Since the tool call ground truth is generated from gpt-4.1 and other models are evaluated against this reference, the tool call metrics (Name Match, Param Match, Order Match) in Table 3 reflect alignment with gpt-4.1's specific approach rather than absolute tool calling quality. Based on the observation of gpt4.1-mini achieving the highest scores in parameter matching (0.878) and order matching (0.887), we hypothesis that gpt-4.1-mini is a distilled model from gpt-4.1.

Conversely, the o3 model's lower performance in tool call precision metrics (Name Match: 0.588, Param Match: 0.803) may not indicate inferior tool calling capabilities, but rather a fundamentally different approach to API interaction that diverges from gpt-4.1's style. The LLM judger metrics provide a more objective assessment of actual task performance, revealing o3's exceptional capabilities in planning (0.903), requirement coverage (0.969), and completeness (0.955). This suggests that while o3 may employ different tool calling patterns compared to gpt-4.1, it demonstrates superior high-level reasoning and task comprehension. The balanced performance of gpt4o and gpt4o-mini across both metric categories indicates these models successfully combine effective planning with tool calling approaches that align well with the gpt-4.1 reference standard. For open-source models like mistral-small-24b and the qwen3 variants, the lower scores

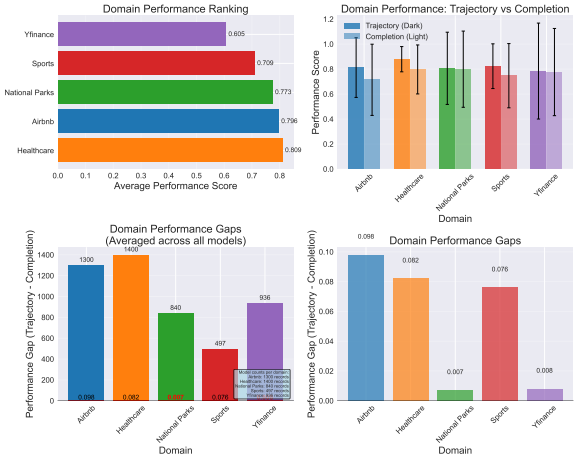


Figure 4: Domain performance analysis: (a) Domain ranking by LLM judger, (b) Trajectory vs completion comparison, (c) Task distribution, (d) Performance gaps by domain.

across all metrics reflect both genuine performance gaps and potential stylistic differences in tool usage patterns compared to the proprietary reference model, highlighting the challenge of fair evaluation when using model-specific ground truth standards.

5.3 Domain Performance Analysis

Figure 4 presents comprehensive domain performance analysis across multiple dimensions, revealing significant variations in task complexity and tool call quality across different domains. Figure 4 reveals a domain performance hierarchy led by Healthcare, followed by Airbnb, Sports, Finance, and National Parks, driven by API structure and task complexity. Healthcare excels due to standard-

Table 3: LLM Judger and Tool Usage Evaluation Metrics (mean ± std).

	Tool Call Metrics			LLM Judger Metrics			
Model	**Name Match**	**Param Match**	**Order Match**	**Planning**	**Execution**	**Req. Coverage**	**Completeness**
mistral-small-24b	0.548 ± 0.185	0.636 ± 0.165	0.645 ± 0.226	0.764 ± 0.176	0.723 ± 0.171	0.569 ± 0.148	0.509 ± 0.135
qwen3-30b-a3b	0.564 ± 0.162	0.731 ± 0.200	0.650 ± 0.213	0.790 ± 0.133	0.726 ± 0.147	0.649 ± 0.188	0.577 ± 0.181
qwen3-32b	0.592 ± 0.150	0.782 ± 0.117	0.693 ± 0.162	0.819 ± 0.104	0.757 ± 0.116	0.710 ± 0.176	0.638 ± 0.176
gpt4.1-mini	0.768 ± 0.163	**0.878 ± 0.087**	**0.887 ± 0.074**	0.875 ± 0.083	0.849 ± 0.097	0.808 ± 0.100	0.742 ± 0.105
gpt4.1-nano	0.630 ± 0.106	0.825 ± 0.130	0.738 ± 0.109	0.837 ± 0.131	0.801 ± 0.145	0.713 ± 0.168	0.633 ± 0.161
gpt4o-mini	0.754 ± 0.157	0.862 ± 0.074	0.849 ± 0.100	0.895 ± 0.058	0.867 ± 0.074	0.832 ± 0.084	0.770 ± 0.095
gpt4o	**0.793 ± 0.106**	0.876 ± 0.086	0.873 ± 0.081	0.902 ± 0.053	**0.884 ± 0.064**	0.838 ± 0.101	0.767 ± 0.119
o3-mini	0.511 ± 0.196	0.688 ± 0.216	0.605 ± 0.191	0.856 ± 0.121	0.824 ± 0.111	0.795 ± 0.170	0.737 ± 0.177
o4-mini	0.571 ± 0.113	0.818 ± 0.137	0.781 ± 0.082	0.783 ± 0.234	0.747 ± 0.216	0.819 ± 0.247	0.790 ± 0.236
o3	0.588 ± 0.186	0.803 ± 0.116	0.803 ± 0.134	**0.903 ± 0.042**	0.858 ± 0.067	**0.969 ± 0.029**	**0.955 ± 0.029**

ized data and strong alignment between API outputs and user needs, while National Parks struggles with complex geographical data integration. Execution vs. completion patterns vary, where Finance stands out for strong completions despite middling trajectories, and Airbnb shows a large gap, highlighting difficulty in generating comprehensive results. Task distribution is highest in Healthcare and Airbnb, ensuring robust analysis. Overall, domain-specific gaps reflect differences in API design quality and task demands.

5.4 Performance Gap Analysis

Figure 5 presents comprehensive analysis of performance gaps across models and domains, revealing fundamental patterns in LLM tool-use capabilities and architectural characteristics. Figure 5 reveals a consistent trend: models generally perform better at trajectory execution than completion, highlighting a maturity gap in synthesis capabilities. Most models show positive execution-completion gaps, with open-source models displaying larger gaps than OpenAI models. Notably, O3 exhibits a rare negative gap, excelling in completion quality. Domain-level analysis shows similar trends, with Finance having the smallest gap and Airbnb the largest, reflecting differences in API structure and task complexity. A weak negative correlation between performance and gap size suggests that stronger models tend to be more balanced, though architecture-specific factors also play a role.

5.5 Model Performance Hierarchy

Figure 6 illustrates the model performance 2-d comparison on both tool call and LLM judger. Regarding tool call criteria, we use name matching and parameter matching scores, while LLM judger criteria averages the scores w.r.t. trajectory and completion. It highlights a clear performance hierarchy

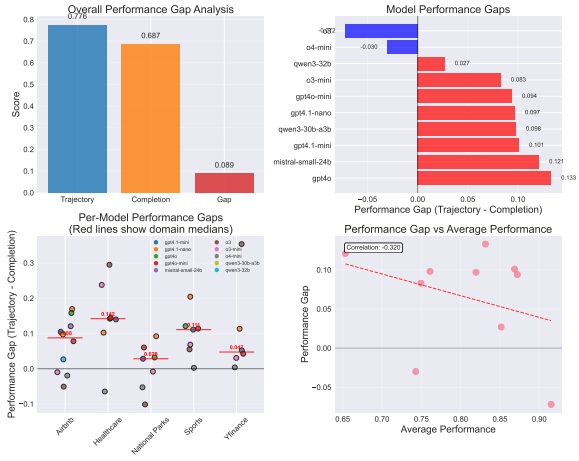


Figure 5: Performance gap analysis: (a) Overall gap distribution, (b) Model-wise gaps, (c) Domain-wise gaps, (d) Gap-performance correlation.

among models, with OpenAI's O3, GPT-4o-mini, and GPT-4.1-mini consistently leading across metrics. OpenAI models outperform open-source ones in overall quality with lower variance, indicating more stable tool-use capabilities. Additionally, the higher parameter match scores than the name match score indicate that models understand the context but may select different tool compared with ground truth from GPT-4.1. A trade-off between trajectory execution and completion quality is evident, with most models excelling in execution but not completion, except O3, which uniquely excels in completion. The top models also show specialization, underscoring that strength in one area does not imply dominance in another, reflecting varied optimization strategies.

5.6 Performance Correlation Analysis

Figure 7 shows the universal execution-completion gap across all LLM models and domains. Finance shows the most pronounced gap, with models achieving 95%+ trajectory scores but dropping

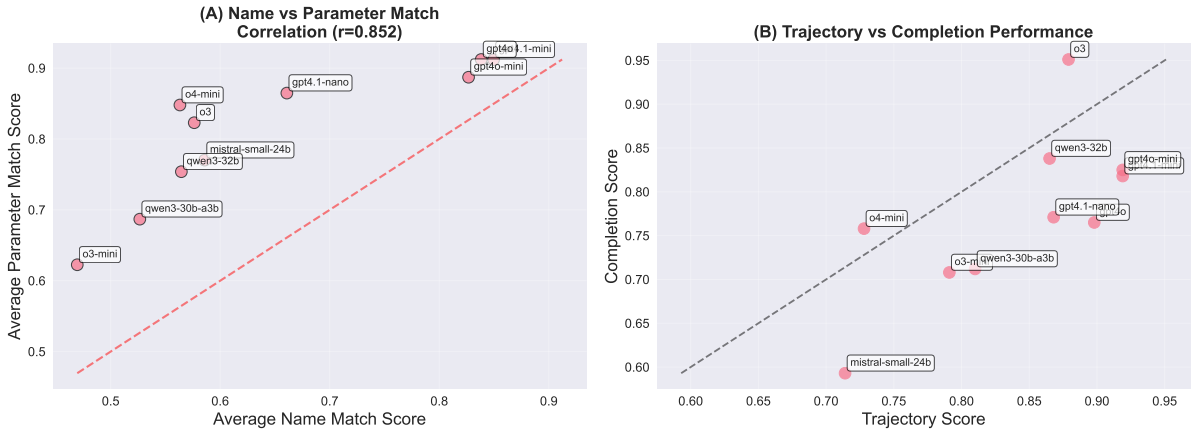


Figure 6: Model performance analysis from tool call anlysis and llm judger.

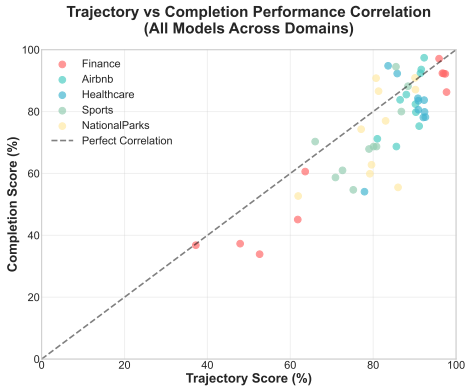


Figure 7: Trajectory vs Completion performance correlation across domains.

to 85-95% completion quality. Healthcare exhibits the most balanced performance near the diagonal line, while Airbnb shows high variability and Sports clusters in the middle range. National Parks demonstrates compressed performance in the lower range, reflecting complex geographical data integration challenges. Notably, no points appear above the diagonal reference line, confirming that current LLM consistently excel at procedural reasoning and tool execution but struggle with output synthesis across all domains.

6 Conclusion

We present *MCPEval*, an automated, MCP-based evaluation framework designed to address critical gaps in LLM agent assessment. Our approach structured the evaluation process across diverse domains, automates end-to-end task generation, verification, and deep assessment, eliminating the boundary in data collection and manual pipeline construction. Extensive experiments across five real-world domains demonstrate that *MCPEval* uncovers nuanced, domain-specific performance differences and provides actionable insights beyond traditional evaluation metrics. By releasing *MCPEval* as an open-source toolkit, we aim to promote reproducible, scalable, and standardized evaluation practices for the LLM agent community and develop more robust and capable AI agents.

Broader Impact

MCPEval has the potential for significant broader impact across multiple stakeholder communities. The framework serves the research community by providing a standardized platform for reproducible agent evaluation research, enabling consistent comparison and validation of different approaches. For industry adoption, it enables systematic assessment of agent readiness for production deployment, helping organizations make informed decisions about agent deployment strategies. In terms of model development, the framework informs training and development priorities through comprehensive capability analysis, guiding future research directions and resource allocation. Finally, the framework contributes to user safety by enabling thorough pre-deployment evaluation, helping ensure that deployed agents meet safety and reliability standards.

Limitations

Despite the strengths of *MCPEval*, several limitations remain. First, our evaluation relies entirely on synthetic data, which may not fully reflect the complexity and unpredictability of real-world agent interactions. Second, using LLM-based judges for long trajectories can be costly in terms of both computation and resource usage, potentially limiting scalability for large-scale or lengthy evaluations. Third, the automated verification process

can introduce bias or produce false ground truth labels, especially for ambiguous or open-ended tasks, which may affect the reliability of certain evaluation results. Future work includes incorporating real-world task data, developing more efficient and cost-effective judging methods, and improving verification strategies to reduce bias and enhance the accuracy of ground truth labels. It would be a future patch to conduct verification and judgment from multiple sources for cross validation.

Acknowledgments

We thank the open-source community for providing useful tools and MCP servers for our *MCPEval*. We also acknowledge the Model Context Protocol community and the various benchmark datasets that have enabled this work. And we would acknowledge part of code of the system are generated from multiple frontier AI models and verified with human checking.

References

Allganize. 2024. Alli for enterprise: On-premise llm app server & the model context protocol (mcp). Accessed: 2025-06-27.

Anthropic. 2024. Model context protocol. https://github.com/modelcontextprotocol/. Accessed: 2024-06-26.

Negin Arabzadeh, Anand A. D. J. C., Fabio F. C., P. G. Ipeirotis, Jin Z., Panos P., and Sanmi S. 2024. AgentEval 1.0: A comprehensive benchmark for evaluating autonomous agents. *arXiv preprint arXiv:2401.07303*.

Harrison Chase and 1 others. 2022. LangChain. https://github.com/langchain-ai/langchain.

Xiang Deng, Adrien Viguier, Xinyi Chen, C Gu, Xinyun Zhang, D Yogatama, M Dréze, C Jia, and W Wang. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*.

Yapei Fan, Tiezheng Lin, Yutao Zhang, Qing Yao, Bo Zhu, Yilun Li, Wenlin Qian, Xin Jiang, Wei Chen, Peiyi Cheng, and 1 others. 2024. From static to dynamic: A survey of evaluation methods for large language models. *arXiv preprint arXiv:2402.04337*.

Xuanqi Gao, Yue Wang, Jinsu Kim, Chang Lee, and Diyi Yang. 2025. MCP-RADAR: A multi-dimensional benchmark for evaluating tool use capabilities in large language models. *arXiv preprint arXiv:2505.16700*.

Zhaolin Geng and Kewei Chang. 2025. REALM-Bench: A real-world planning benchmark for llms and multi-agent systems. *arXiv preprint arXiv:2502.18836*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Thai Hoang, Kung-Hsiang Huang, Shirley Kokane, Jianguo Zhang, Zuxin Liu, Ming Zhu, Jake Grigsby, Tian Lan, Michael S Ryoo, Chien-Sheng Wu, and 1 others. 2025. Lam simulator: Advancing data generation for large action model training via online exploration and trajectory feedback. *arXiv preprint arXiv:2506.02298*.

Zeshan Huang, Zihan Zeng, Keren Chen, Yihui Wei, Guilei Yuan, Jinhao Li, Jiaan Yang, Ziqi Wang, Jiateng Liu, Zexin Wei, and 1 others. 2024. A survey on evaluation of large language models as agents. *arXiv preprint arXiv:2406.03456*.

Zhexin Ji, Zhaofan Liu, Zihan Zhao, Fuhao Yuan, Cheng Li, Yang Lin, Pinyu Wang, Yaodong Zhang, and Jing Liu. 2024. SafetyBench: A comprehensive benchmark to evaluate LLMs' safety. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 7406–7421, Mexico City, Mexico. Association for Computational Linguistics.

Carlos E Jimenez, John He, Shafiq Joty, and Wei-Hao Shang. 2023. Swe-bench: Can language models solve real-world software engineering problems? *arXiv preprint arXiv:2310.06770*.

Jing Yu Koh, Robert Gu, Hong-Lek Lee, Xuhui Zhou, Xingyu Geng, Hao Zhu, Zhengyun Li, Peiran Gu, Yin-Dong Zhang, Yi Zhang, and 1 others. 2024. VisualWebArena: A realistic and challenging benchmark for multimodal web agents. *arXiv preprint arXiv:2401.13649*.

Shirley Kokane, Ming Zhu, Tulika Awalgaonkar, Jianguo Zhang, Thai Hoang, Akshara Prabhakar, Zuxin Liu, Tian Lan, Liangwei Yang, Juntao Tan, and 1 others. 2024. Spectool: A benchmark for characterizing errors in tool-use llms. *arXiv preprint arXiv:2411.13547*.

Percy Liang, Rishi Bommasani, Tony Lee, D Mada, D Hudson, E Hall, T Icard, H Adel, A Adipo, J Aina, and 1 others. 2022. Holistic evaluation of language models. In *Advances in Neural Information Processing Systems*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yaran Xu, Zekun Wang, Ruobing Zhang, C Tan, C Xu, X Li, R Yang, and 1 others. 2023a. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Rithesh Murthy, Liangwei Yang, Zuxin Liu, Tian Lan, Ming Zhu, Juntao Tan, Shirley Kokane, Thai Hoang,

Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. 2024a. Pract: Optimizing principled reasoning and acting of llm agent. *Preprint*, arXiv:2410.18528.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, and 1 others. 2023b. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents. *arXiv preprint arXiv:2308.05960*.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Liangwei Yang, Zuxin Liu, Juntao Tan, Prafulla K Choubey, Tian Lan, Jason Wu, Huan Wang, and 1 others. 2024b. Agentlite: A lightweight library for building and advancing task-oriented llm agent system. *arXiv preprint arXiv:2402.15538*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, and 1 others. 2024c. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482.

Ed Lumer, Chang Lee, Jinsu Kim, and Yeong-Dae Kim. 2025. ScaleMCP: Dynamic and auto-synchronizing model context protocol tools for llm agents. *arXiv preprint arXiv:2505.06416*.

Yitong Ma, Zeyu Zhang, Zepu Lin, Ke Shu, and Chen Wang. 2024a. AgentBoard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*. Accepted at NeurIPS 2024.

Zixian Ma, Jianguo Zhang, Zhiwei Liu, Jieyu Zhang, Juntao Tan, Manli Shu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Caiming Xiong, and 1 others. 2024b. Taco: Learning multi-modal action models with synthetic chains-of-thought-and-action. *arXiv preprint arXiv:2412.05479*.

OpenAI. 2023. Gpt-4 technical report. https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.

Yujia Qin, Shi-Cheng Cai, Y-H Liang, Y Zhang, X Zhao, Y Lin, Y-H Yao, X Deng, Z-K Li, C Dong, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Joao Roveda and 1 others. 2023. CrewAI. https://github.com/joaomdmoura/crewAI.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Aitor Garriga, and 1 others. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. In *International Conference on Machine Learning*, pages 20499–20689. PMLR.

Juntao Tan, Liangwei Yang, Zuxin Liu, Zhiwei Liu, Rithesh Murthy, Tulika Manoj Awalgaonkar, Jianguo Zhang, Weiran Yao, Ming Zhu, Shirley Kokane, Silvio Savarese, Huan Wang, Caiming Xiong, and Shelby Heinecke. 2025. Personabench: Evaluating ai models on understanding personal information through accessing (synthetic) private user data. *Preprint*, arXiv:2502.20616.

Shuke Tang, Zexuan Shi, Wenhai Chen, Zhaowei Zhao, Zirui Zhuang, Guanguan Zhang, Feng Chen, and Jie Luo. 2024. MATRIX: A multi-agent reinforcement learning environment for text-based social interaction simulation. *arXiv preprint arXiv:2405.02705*.

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. *GitHub repository*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language model with self generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13280–13299.

Colin White, Samuel Dooley, Manley Roberts, Arka Shtedritski, Chris Pochinkov, Shay Ku, Neel Jain, Siddharth Jha, Jiayi Ren, John Sleigh, and 1 others. 2024. LiveBench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*.

Qingyun Wu and 1 others. 2023. AutoGen: Enabling next-gen llm applications via multi-agent conversation framework. https://github.com/microsoft/autogen.

Tianbao Xie, Danyang Chen, Zhao Gao, Chun-Che Hsieh, Tao Yao, Hongjin Cao, Zetian Jin, Yunwei Gao, Zhenmei Li, Yifei Shen, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Fung, Yixin Lin, Xingxin Wu, Wenfeng Li, Weiming Jiang, and 1 others. 2023. WizardLM: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Meng Yan, Ruihang Liu, Jinsu Kim, Chang Lee, and Diyi Yang. 2025. MCPWorld: A unified benchmarking testbed for api, gui, and hybrid computer use agents. *arXiv preprint arXiv:2506.07672*.

Shunyu Yao, Howard Chen, John Gu, K R-K, C Y-F, Q Le, and D Song. 2022. Webshop: Towards

scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*.

Cheng Zhang, Quan Zhang, Zhipu Wu, Jun-Yan Li, Wen-Juan Lu, Chang-Gen Lin, Sa-Hai Wang, Bin Yu, Philip S. Yu, Li-Rong Sun, and 1 others. 2023a. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Jia-Chen Zhang, Yitong Zhu, Zhuohao Li, Ge Wang, He Zhao, and Min-Ling Zhang. 2024a. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*. Accepted at NeurIPS 2024.

Jianguo Zhang, Thai Hoang, Ming Zhu, Zuxin Liu, Shiyu Wang, Tulika Awalgaonkar, Akshara Prabhakar, Haolin Chen, Weiran Yao, Zhiwei Liu, and 1 others. 2025. Actionstudio: A lightweight framework for data and training of large action models. *arXiv preprint arXiv:2503.22673*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, and 1 others. 2024b. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*.

Jianguo Zhang, Kun Qian, Zhiwei Liu, Shelby Heinecke, Rui Meng, Ye Liu, Zhou Yu, Huan Wang, Silvio Savarese, and Caiming Xiong. 2023b. Dialogstudio: Towards richest and most diverse unified dataset collection for conversational ai. *arXiv preprint arXiv:2307.10172*.

Leon Zheng, Serena Kou, Neel Kumar, Hieu Ngo, Boxin Zhang, Zhaohui Wang, Percy Li, and Percy Liang. 2024. HELM Safety: Towards standardized safety evaluations of language models. *arXiv preprint arXiv:2405.09340*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Brooks, Eric Xing, and 1 others. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Li, Zhengyun Li, C Liu, P Gu, Y Zhang, C Li, and 1 others. 2023. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Jihan Zhu, Zhizheng Lin, Jun Gao, Zhaoxuan Zhou, Yaodong Zhang, Zhaofan Liu, Ceyao Zheng, Cheng Li, Zhaokai Wang, Zili Wang, and 1 others. 2024. A survey of ai agent evaluation: A hundred unsolved problems and a one-stop open-source library. *arXiv preprint arXiv:2406.09844*.

A Comprehensive Experiments and Results

We present a comprehensive evaluation of 10 state-of-the-art LLM models across 5 diverse domains using the *MCPEval* framework. Our analysis encompasses 5k trajectory records and 5k completion records with detailed individual task analysis, plus 50 model-domain combinations from systematic LLM judger evaluation, representing the most extensive evaluation of LLM tool-use capabilities to date.

A.1 Experimental Setup

Model Selection Our evaluation includes 10 models spanning different architectures and capabilities:

- OpenAI Models (7): GPT-4o, GPT-4o-mini, GPT-4.1-mini, GPT-4.1-nano, O3, O3-mini, O4-mini
- Open-Source Models (3): Mistral-Small-24B, Qwen3-32B, Qwen3-30B-A3B

Domain and Tools Coverage We evaluate performance across 5 diverse application domains with the following tools:

- Finance: Stock prices, financial data, market analysis, portfolio management
- Healthcare: Medical terminology lookup, drug information, clinical trials, health topics, PubMed search
- Airbnb:`airbnb_search`, `airbnb_listing_details`
- Sports: Team statistics, player information, game schedules, league standings
- National Parks: Park information, visitor services, trail details, facility booking

B Evaluation Criteria

To holistically assess the capabilities of AI agents in complex task environments, we adopt a two-dimensional evaluation criteria: (1) **Tool Call Performance**, which measures the correctness of predicted tool usage against the ground truth, and (2) **LLM Judger Performance**, which scores the overall quality of execution trajectories and task completion using rubric-based judgment. These complementary dimensions enable both precise operational assessment and high-level behavioral evaluation.

B.1 Tool Call Criteria

Tool call evaluation is conducted using the MCP Model Evaluator's `analyze` command. This system compares the agent's predicted tool usage with the ground truth across both *strict* and *flexible* matching protocols.

Strict Matching: Requires exact correspondence on tool names, parameter values, and execution order. It represents a binary success paradigm: either the task is fully correct or it fails.

Flexible Matching: Allows partial credit by applying similarity thresholds for parameter values ($\geq$ 0.6) and tool order ($\geq$ 0.5). It reflects more tolerant criteria aligned with approximate but contextually appropriate predictions.

Metric Definitions:

- **Average Name Match Score**: Measures the proportion of correctly predicted tool names.
- **Average Parameter Match Score**: Assesses correctness or similarity of parameter values.
- **Average Order Match Score**: Evaluates sequence alignment between predicted and actual tool calls.
- **Average Overall Score**: A weighted combination of the above metrics, using configurable weights (default: 0.4 for name and parameter, 0.2 for order).

In addition to aggregate statistics, the system also provides fine-grained diagnostics such as missing/extra tools, parameter mismatches, and tool-specific success rates, facilitating detailed error analysis.

Table 4: Model names and their corresponding versions.

Model	Model Version
mistral-small-24b	mistralai/Mistral-Small-3.2-24B-Instruct-2506
qwen3-30b-a3b	Qwen/Qwen3-30B-A3B
qwen3-32b	Qwen/Qwen3-32B
gpt4.1-mini	gpt-4.1-mini-2025-04-14
gpt4.1-nano	gpt-4.1-nano-2025-04-14
gpt-4o-mini	gpt-4o-mini-2024-07-18
gpt-4o	gpt-4o-2024-08-06
gpt4.1	gpt-4.1-2025-04-14
o3-mini	o3-mini-2025-01-31
o4-mini	o4-mini-2025-04-16
o3	o3-2025-04-16

B.2 LLM Judger Criteria

Beyond tool-level correctness, we incorporate a rubric-based judgment system to evaluate execution quality from a human-centric perspective. This LLM Judger scores each trajectory on multiple aspects of agent behavior, encompassing planning, execution logic, tool use, and final task outcomes.

Trajectory Evaluation Aspects:

- **Planning:** Measures task understanding and decomposition.
- **Execution Flow:** Assesses the logical coherence and sequencing of actions.
- **Tool Selection & Usage:** Evaluates appropriateness of tools and correctness of parameters.
- **Adaptability:** Captures the agent's response to unexpected results and changing context.
- **Efficiency:** Reflects the conciseness and lack of redundancy in the solution.
- **Context Awareness:** Checks whether the agent maintains relevant constraints and state.

Task Completion Evaluation Aspects:

- **Requirement Coverage:** Degree to which the output fulfills all task goals.
- **Accuracy:** Factual and logical correctness of the output.
- **Completeness:** Depth and breadth of the agent's response.
- **Usefulness:** Practical value and user relevance of the solution.

Each aspect is scored on a continuous scale from 0.0 to 1.0, allowing nuanced grading. A high-quality trajectory demonstrates coherent planning, accurate tool use, robust adaptability, and a final response that is complete, accurate, and valuable.

C Model Versions

Table 4 presents the model names and their corresponding versions used in this work. For certain models, including the o3 and o4 series, the default temperature was used, as their APIs do not support temperature adjustment. We set the temperature to a constant value of 0.01 for the rest to ensure reproducible results.

C.1 Overall Performance Results

C.1.1 Model Performance Hierarchy

Figure 8 illustrates the comprehensive model performance analysis across different evaluation dimensions, revealing distinct performance patterns and architectural characteristics.

Figure 8(a) demonstrates the overall performance ranking based on LLM judger evaluation, with O3 leading at 0.926, followed closely by GPT-4o-mini (0.852) and GPT-4.1-mini (0.846). The ranking reveals

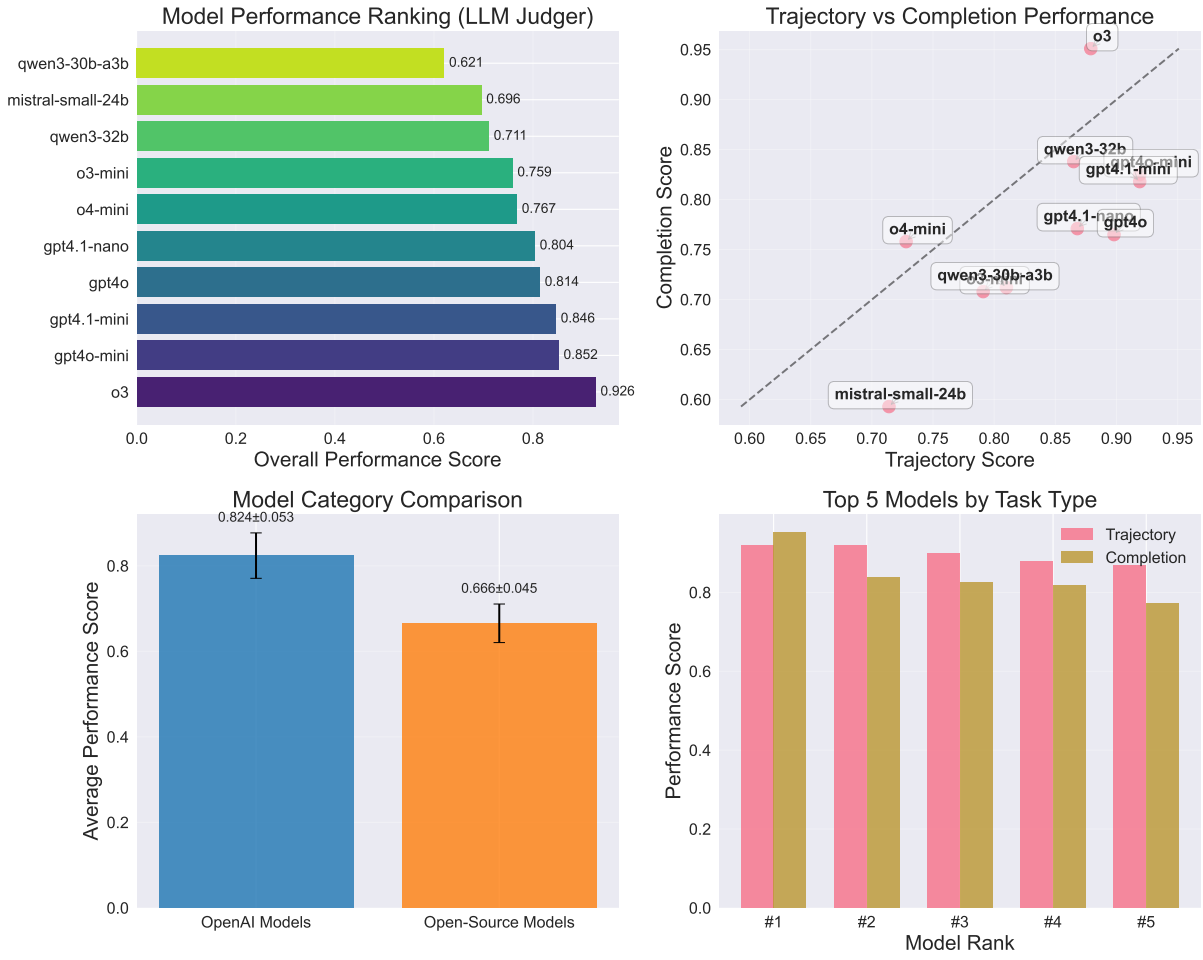


Figure 8: Model performance analysis: (a) Overall ranking by LLM judger, (b) Trajectory vs completion performance scatter plot, (c) Model category comparison, (d) Top 5 models by task type.

a clear performance hierarchy, with OpenAI models dominating the top positions and open-source models showing more variable performance. Notably, the performance distribution shows a gradual decline from top performers (>0.8) to lower-performing models (>0.6), indicating significant capability differences across model architectures.

Figure 8(b) presents the trajectory vs completion performance scatter plot, revealing the fundamental execution-completion trade-off. Most models cluster above the diagonal line, indicating superior trajectory execution compared to completion quality. O3 stands out as the model below the diagonal, demonstrating exceptional completion quality relative to its trajectory performance. The tight clustering of GPT models (GPT-4o, GPT-4o-mini, GPT-4.1-mini) in the upper-right region indicates consistent high performance across both dimensions, while open-source models show greater variance and generally lower performance.

Figure 8(c) illustrates the stark performance gap between model categories, with OpenAI models achieving an average score of 0.824 compared to 0.676 for open-source models, showing a significant performance difference. This performance gap highlights the current disparity in tool-use capabilities between proprietary and open-source architectures. The error bars indicate that while OpenAI models show consistent performance with low variance, open-source models exhibit higher variance, suggesting less stable tool-use capabilities.

Figure 8(d) compares the top 5 models by task type, revealing specialization patterns across trajectory execution and completion quality. GPT-4.1-mini, GPT-4o-mini, and GPT-4o dominate trajectory performance with scores above 0.9, while O3, Qwen3-32B, and GPT-4o-mini lead in completion quality. This analysis highlights that model excellence in one dimension doesn't guarantee equivalent performance in the other, suggesting different optimization strategies across model architectures.

Table 5: Model performance rankings.

Model	Trajectory	Completion	Overall (LLM)
Mistral-Small-24B	0.714 ± 0.253	0.593 ± 0.282	0.719
Qwen3-30B-A3B	0.810 ± 0.251	0.712 ± 0.272	0.609
Qwen3-32B	0.865 ± 0.185	0.838 ± 0.216	0.705
GPT-4.1-mini	**0.915** ± 0.098	0.805 ± 0.143	0.839
GPT-4o-mini	0.911 ± 0.113	0.810 ± 0.164	0.839
GPT-4o	0.910 ± 0.112	0.806 ± 0.157	0.806
GPT-4.1-nano	0.866 ± 0.108	0.757 ± 0.164	0.796
O3-mini	0.760 ± 0.289	0.669 ± 0.266	0.748
O4-mini	0.698 ± 0.352	0.742 ± 0.363	0.755
O3	0.857 ± 0.179	**0.951** ± 0.171	**0.926**

The detailed analysis in Table 5 reveals distinct performance patterns that reflect fundamental architectural differences across model families. The trajectory execution rankings demonstrate remarkable consistency among top-tier OpenAI models, with GPT-4.1-mini, GPT-4o-mini, and GPT-4o achieving nearly identical scores within a 0.005 range. This tight clustering suggests that these models share similar procedural reasoning capabilities, likely stemming from comparable training methodologies and architectural foundations.

The completion quality rankings present a more diverse landscape, with O3 achieving exceptional performance that surpasses all other models by at least 0.113 points. This dominance in completion quality, combined with O3's moderate trajectory performance, indicates a fundamentally different optimization strategy, prioritizing output synthesis over execution consistency. Qwen3-32B's strong completion performance demonstrates that open-source models can achieve competitive output quality, though this doesn't translate to overall performance leadership due to lower trajectory scores. The standard deviation patterns reveal important insights about model reliability. GPT-4.1-mini shows the lowest trajectory variance, indicating exceptional consistency in execution quality.

The performance tiers evident in the rankings, top performers (>0.8), mid-tier models (0.7-0.8), and lower performers (0.6-0.7), reflect fundamental capability differences rather than minor variations. The performance gap between different models represents a substantial difference in practical tool-use effectiveness, highlighting the current disparity in LLM capabilities for complex tool-use scenarios.

Through Table 5, we can summarize that:

- *Trajectory Leaders*: GPT-4.1-mini, GPT-4o-mini, GPT-4o demonstrate superior execution consistency.
- *Completion Leaders*: O3, Qwen3-32B, GPT-4o-mini excel in output quality.
- *Overall Champions*: O3 achieves the best combined performance, followed by GPT-4o-mini and GPT-4.1-mini.
- *Category Gap*: OpenAI models (0.824 avg) significantly outperform open-source models (0.676 avg) by 0.148 points.

C.2 Domain Performance Analysis

Figure 9 presents comprehensive domain performance analysis across multiple dimensions, revealing significant variations in task complexity and API design quality across different application domains.

Figure 9(a) establishes the domain performance hierarchy through LLM judger evaluation, with Healthcare leading at 0.809, followed by Airbnb (0.796), National Parks (0.773), Sports (0.709), and Finance (0.605). This ranking reflects the varying complexity of domain-specific tool ecosystems and the structural quality of underlying APIs. Healthcare's dominance suggests that well-standardized medical terminologies and structured data formats facilitate superior LLM performance, while Finance's challenges indicate the complexity of financial data interpretation and market volatility.

Figure 9(b) compares trajectory and completion performance across domains, revealing domain-specific execution-completion patterns. Healthcare demonstrates the highest trajectory performance with strong completion quality, indicating well-aligned API responses with user expectations. National Parks

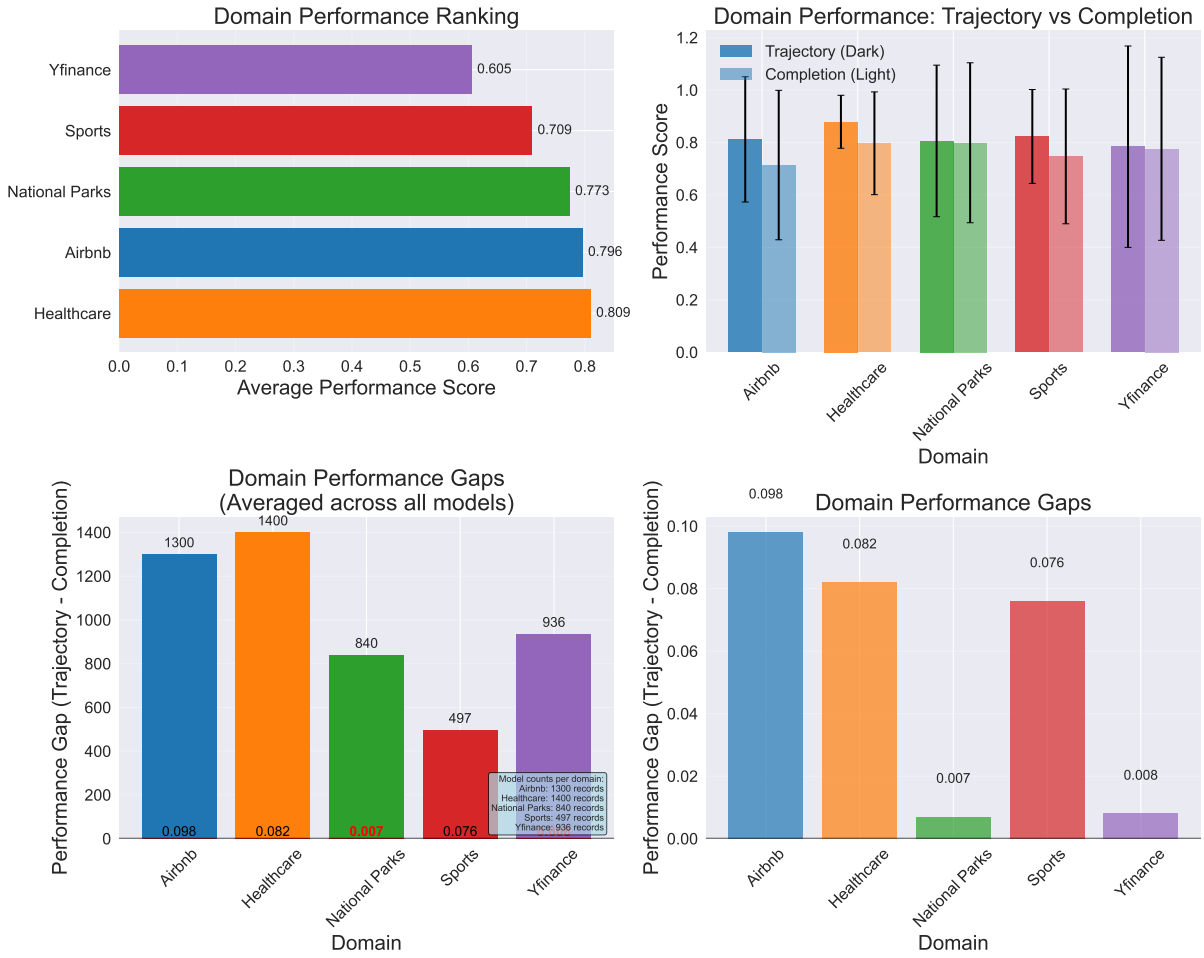


Figure 9: Domain performance analysis: (a) Domain ranking by LLM judger, (b) Trajectory vs completion comparison, (c) Task distribution, (d) Performance gaps by domain.

shows balanced performance with the smallest gap, suggesting that park information APIs provide clear, structured outputs. Finance exhibits lower trajectory performance, reflecting the complexity of financial data interpretation, while Airbnb shows a significant execution-completion gap, indicating challenges in translating property searches into comprehensive recommendations.

Figure 9(c) illustrates the task distribution across domains, with Healthcare (1,400 tasks) and Airbnb (1,300 tasks) representing the largest evaluation sets, followed by Finance (936 tasks), National Parks (840 tasks), and Sports (497 tasks). This distribution reflects both the complexity and practical importance of these domains in real-world applications. The substantial task counts in Healthcare and Airbnb provide robust statistical power for performance analysis, while the smaller Sports dataset still offers sufficient coverage for meaningful evaluation.

Figure 9(d) reveals domain-specific performance gaps, highlighting fundamental differences in task complexity and API design quality. Airbnb exhibits the largest gap (0.098), suggesting that while models can effectively search for properties, they struggle to synthesize results into useful, comprehensive recommendations. Healthcare and Sports show moderate gaps (0.082 and 0.076), indicating balanced but execution-favored performance. National Parks demonstrates the smallest gap (0.007), reflecting the alignment between structured park information and model output capabilities. Finance shows a small gap (0.008), suggesting that financial data APIs provide clear outputs that match user requirements.

The comprehensive domain analysis in Table 6 reveals fundamental differences in task complexity and API ecosystem quality across application domains. Healthcare's dominance in both trajectory and completion performance reflects the mature state of medical informatics infrastructure, where standardized terminologies (ICD-10, SNOMED-CT) and well-documented APIs facilitate effective tool use. The

Table 6: Domain performance characteristics.

Domain	Trajectory	Completion	Gap
Healthcare	**0.879** ± 0.101	**0.797** ± 0.196	0.082
Airbnb	0.812 ± 0.239	0.714 ± 0.285	0.098
Sports	0.823 ± 0.179	0.747 ± 0.257	0.076
Finance	0.784 ± 0.359	0.776 ± 0.329	0.008
National Parks	0.806 ± 0.184	0.799 ± 0.187	0.007

moderate gap indicates that medical APIs generally produce outputs that align well with user expectations, though synthesis challenges remain.

The variance patterns provide crucial insights into domain stability and predictability. Healthcare shows the lowest trajectory variance (±0.101), indicating consistent performance across diverse medical tasks, while Finance exhibits the highest variance, reflecting the inherent volatility and complexity of financial data interpretation. This high variance in Finance, combined with its strong completion performance, suggests that while financial APIs can produce high-quality outputs, the execution path varies significantly based on market conditions and data availability.

The performance gap analysis reveals domain-specific challenges in translating execution success into output quality. Airbnb's large gap indicates fundamental challenges in property recommendation synthesis—while models can effectively search and filter properties, they struggle to present results in comprehensive, actionable formats. Conversely, National Parks' minimal gap suggests that park information APIs are well-designed for direct consumption, requiring minimal synthesis for user value.

The domain ranking hierarchy reflects not just API quality but also the inherent complexity of data interpretation and user requirements. Finance's challenges stem from the need to interpret complex financial data, market trends, and provide actionable investment insights. This complexity manifests in lower execution consistency and moderate completion quality, highlighting the challenges of real-world financial data analysis.

Summarizing Table 6, we can find that:

- *Healthcare* emerges as the most successful domain, likely due to well-structured medical APIs and standardized terminology.
- *National Parks* shows the smallest performance gap, indicating better alignment between execution and output quality.
- *Finance* proves most challenging, reflecting complex financial data interpretation requirements.
- *Airbnb* demonstrates the largest performance gap, suggesting execution-completion misalignment issues

C.3 Aspect-Level Performance Analysis

Figure 10 provides detailed analysis of trajectory and completion aspects across both evaluation methodologies, revealing the specific strengths and weaknesses of current LLM architectures in tool-use scenarios.

Figure 10(a) presents trajectory aspect performance from LLM judger evaluation, revealing models' strategic and execution capabilities. Planning emerges as the strongest aspect, indicating that models excel at formulating coherent action sequences for complex tasks. Adaptability and Tool Selection follow closely, demonstrating models' ability to adjust strategies and choose appropriate tools. However, Tool Usage represents the primary bottleneck, suggesting that while models can select correct tools, they struggle with precise parameter specification and effective tool utilization.

Figure 10(b) illustrates completion aspect performance, highlighting the quality challenges in LLM output generation. Requirement Coverage leads among completion aspects, indicating that models generally address the core requirements of given tasks. Accuracy closely follows, suggesting that when models produce outputs, they tend to be factually correct. However, Completeness and Usefulness lag significantly, revealing that models often produce incomplete or less practical outputs despite correct execution.

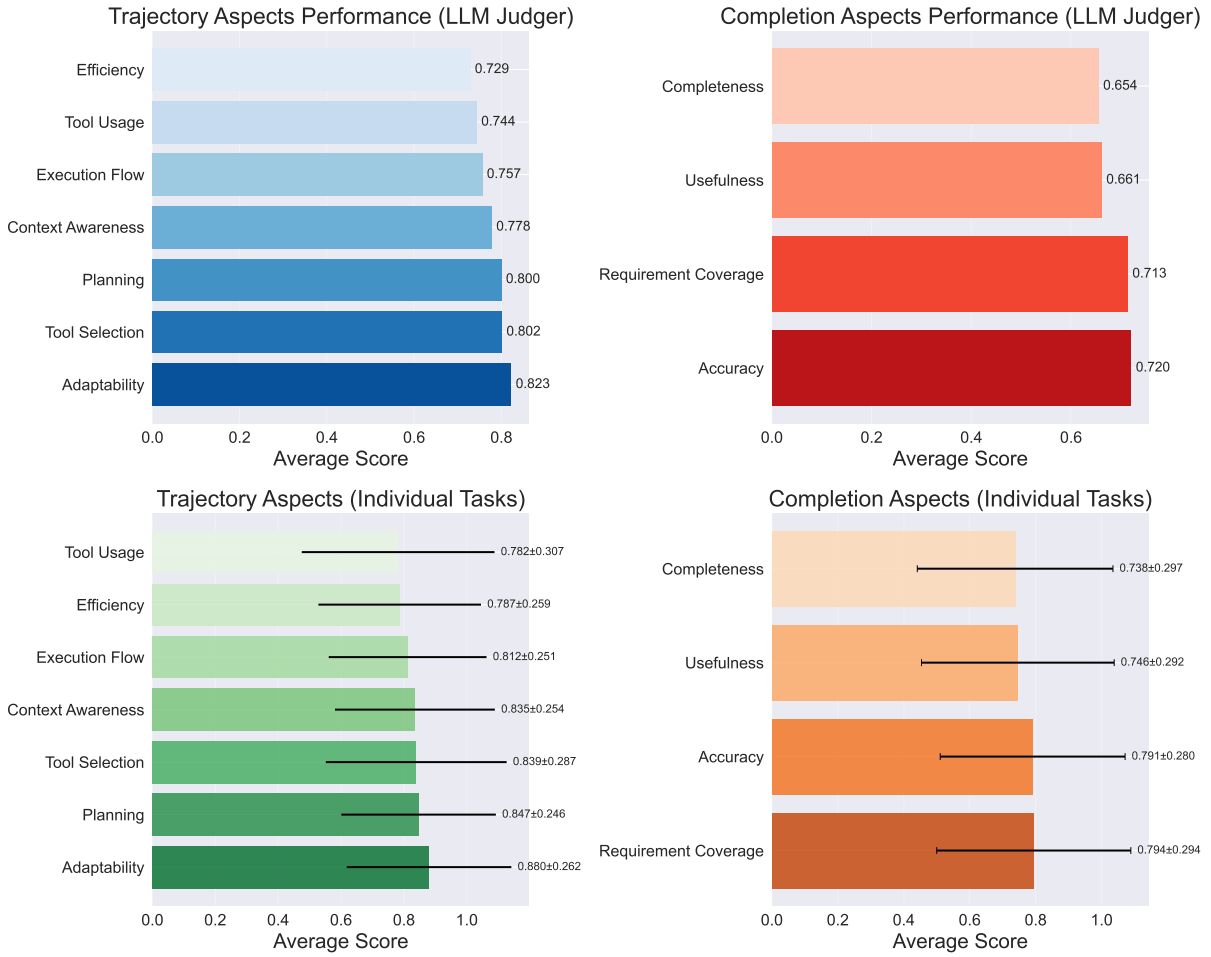


Figure 10: Aspect performance analysis: (a) Trajectory aspects (LLM judger), (b) Completion aspects (LLM judger), (c) Trajectory aspects (individual tasks), (d) Completion aspects (individual tasks).

Figure 10(c) shows trajectory aspects from individual task evaluation, providing granular insights into execution quality. Adaptability and Planning achieve the highest scores, confirming models' strategic strengths across evaluation methodologies. Tool Selection (and Context Awareness demonstrate solid performance, while Execution Flow and Efficiency (0.789) show moderate scores. Tool Usage again emerges as the weakest aspect, consistent with LLM judger findings and confirming this as a fundamental limitation.

Figure 10(d) presents completion aspects from individual task evaluation, corroborating the completion quality challenges identified in LLM judger analysis. Requirement Coverage and Accuracy lead the completion aspects, indicating that models generally produce relevant and correct outputs. However, Usefulness and Completeness show lower scores, suggesting that while outputs are accurate, they often lack practical value or comprehensive coverage of user needs.

C.3.1 Trajectory Aspects

The trajectory aspect analysis in Table 7 reveals the hierarchical structure of LLM tool-use capabilities, with clear distinctions between strategic and operational competencies. The consistent ranking across both evaluation methodologies validates the reliability of our assessment framework and confirms fundamental patterns in current LLM architectures.

Strategic aspects, like Adaptability, Planning, and Tool Selection, demonstrate superior performance, indicating that models excel at high-level reasoning tasks. The strong adaptability scores suggest that models can effectively adjust their strategies when initial approaches fail, while excellent planning capabilities indicate sophisticated multi-step reasoning. These strategic strengths reflect the success of

Table 7: Trajectory aspect performance.

Aspect	Individual Tasks	LLM Judger
Adaptability	**0.874** ± 0.245	0.810 ± 0.163
Planning	0.857 ± 0.226	**0.818** ± 0.163
Tool Selection	0.842 ± 0.272	0.805 ± 0.163
Context Awareness	0.839 ± 0.238	0.786 ± 0.163
Execution Flow	0.818 ± 0.235	0.762 ± 0.163
Efficiency	0.789 ± 0.246	0.726 ± 0.163
Tool Usage	0.776 ± 0.296	**0.722** ± 0.163

Table 8: Completion aspect performance.

Aspect	Individual Tasks	LLM Judger
Requirement Coverage	**0.793** ± 0.278	**0.709** ± 0.217
Accuracy	0.785 ± 0.263	0.706 ± 0.217
Usefulness	0.741 ± 0.276	0.654 ± 0.217
Completeness	0.730 ± 0.282	0.645 ± 0.217

current training methodologies in developing abstract reasoning capabilities.

Operational aspects show more variable performance, with Context Awareness performing well but Execution Flow and Efficiency showing moderate scores. The decline in operational performance suggests that while models can formulate good strategies, they face challenges in optimal execution. This pattern indicates that current architectures may be optimized for reasoning rather than execution efficiency.

Tool Usage emerges as the critical bottleneck, representing the lowest-performing aspect across both evaluation methodologies. This fundamental limitation reflects challenges in precise parameter specification, API interaction patterns, and error handling. The high variance in Tool Usage indicates significant inconsistency across different tools and domains, suggesting that current models lack robust tool interaction capabilities.

The variance patterns reveal important insights about aspect stability. Planning shows relatively low variance, indicating consistent strategic capabilities, while Tool Selection exhibits higher variance, suggesting that tool choice quality varies significantly based on task complexity and domain characteristics. This pattern highlights the need for improved tool interaction training and more robust parameter specification mechanisms.

C.3.2 Completion Aspects

The completion aspect analysis in Table 8 reveals a fundamental dichotomy in LLM output capabilities, with models demonstrating strength in correctness but weakness in synthesis quality. The consistent ranking across both evaluation methodologies, where Requirement Coverage > Accuracy > Usefulness > Completeness, indicates systematic patterns in how current models generate outputs.

Requirement Coverage and Accuracy represent the strongest completion aspects, indicating that models generally understand task requirements and produce factually correct outputs. The close performance between these aspects (0.008 difference in individual tasks) suggests that when models address requirements, they typically do so accurately. This strength reflects the success of current training methodologies in developing factual consistency and requirement understanding.

However, the significant drop in Usefulness and Completeness reveals fundamental limitations in output synthesis capabilities. The gap between Accuracy and Usefulness indicates that while outputs are correct, they often lack practical value for users. This pattern suggests that models struggle to transform correct information into actionable insights, highlighting a critical limitation in current architectures.

The variance patterns provide additional insights into completion quality stability. Requirement Coverage shows high variance, indicating significant inconsistency in how well models address different types of requirements. Completeness exhibits the highest variance, suggesting that output comprehensiveness varies dramatically based on task complexity and domain characteristics. This instability in completion quality contrasts with the more consistent trajectory performance, indicating that output generation is more sensitive to task and model variations.

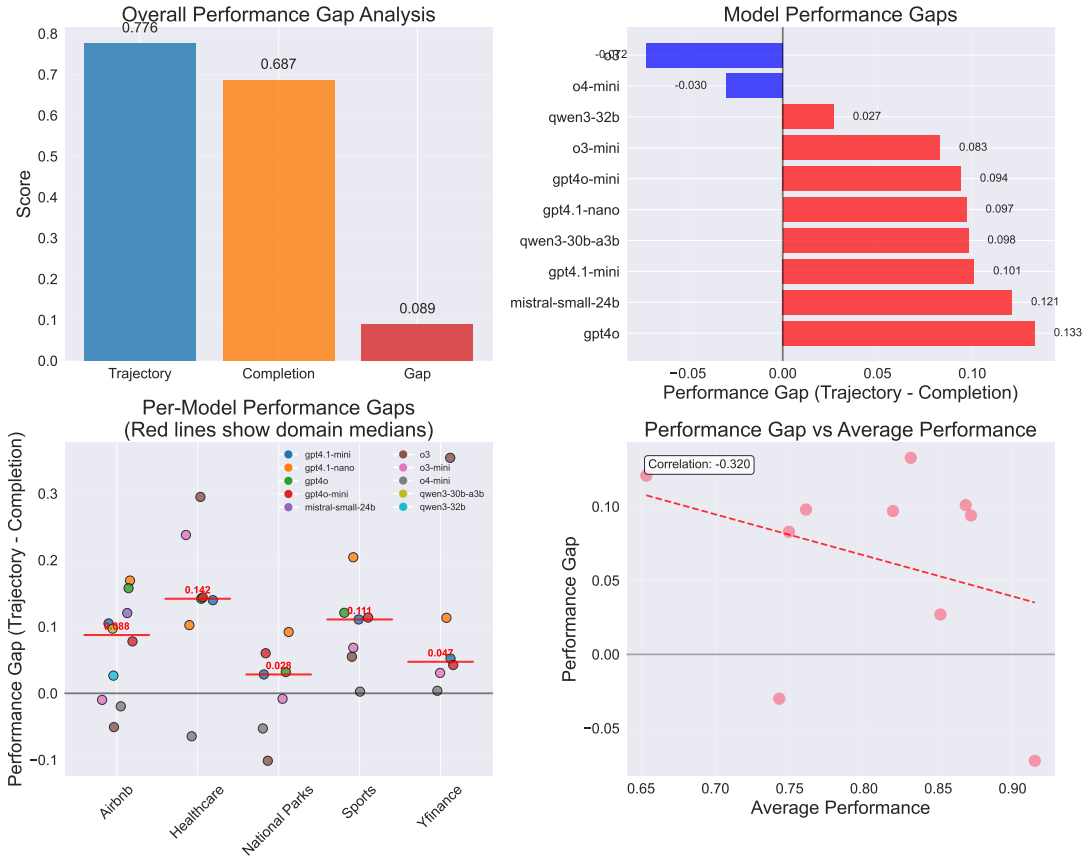


Figure 11: Performance gap analysis: (a) Overall gap distribution, (b) Model-wise gaps, (c) Domain-wise gaps, (d) Gap-performance correlation

The systematic decline from correctness (Accuracy: 0.785) to synthesis quality (Completeness: 0.730) represents a 0.055-point degradation, highlighting the fundamental challenge in current LLM architectures. Models excel at retrieving and presenting correct information but struggle to synthesize comprehensive, useful outputs. This pattern suggests that future development should prioritize output synthesis capabilities, particularly in generating complete and practically useful responses.

Through Table 8, we can get some interesting findings:

- *Strongest Trajectory Aspects*: Adaptability and Planning demonstrate models' strategic capabilities.
- *Weakest Trajectory Aspect*: Tool Usage represents the primary bottleneck in current LLM tool-use capabilities.
- *Completion Quality Leaders*: Requirement Coverage and Accuracy show models generally address task requirements correctly.
- *Completion Challenges*: Completeness and Usefulness indicate room for improvement in output quality.

C.4 Performance Gap Analysis

Figure 11 presents comprehensive analysis of performance gaps across models and domains, revealing fundamental patterns in LLM tool-use capabilities and architectural characteristics.

Figure 11(a) demonstrates the overall performance gap distribution, showing trajectory execution consistently outperforming completion quality across all evaluations. This universal phenomenon indicates that while models excel at procedural reasoning and tool orchestration, they struggle with synthesis and output generation. The distribution reveals that execution capabilities are more mature than completion quality in current LLM architectures, suggesting optimization priorities in model development.

Figure 11(b) illustrates model-specific gap patterns, revealing distinct architectural characteristics across different LLM families. Most models show positive gaps ranging from 0.027 (Qwen3-32B) to 0.133

Table 9: Tool usage pattern analysis.

Pattern	Range	Observation
Exact Match Success	0.027 - 0.695	Wide performance variation
Flexible Match Success	0.015 - 0.695	Improved with flexibility
Parameter Mismatches	Universal	Common across all models
Multi-tool Coordination	Lower success	Complex task challenges
Tool Combination Errors	Frequent	Sequence and dependency issues

(GPT-4o), indicating execution-favored performance. Remarkably, O3 exhibits a negative gap (-0.072), demonstrating superior completion quality relative to trajectory execution, which is a unique characteristic that contributes to its overall performance leadership. The OpenAI models (GPT-4.1-mini, GPT-4o-mini, GPT-4o) demonstrate moderate gaps, suggesting balanced but execution-favored architectures, while open-source models show larger gaps, indicating architectural limitations in output synthesis.

Figure 11(c) reveals domain-specific performance characteristics that reflect task complexity and API design quality. All domains exhibit positive gaps, but with significant variation ranging from 0.007 (National Parks) to 0.098 (Airbnb). National Parks demonstrates the smallest gap, suggesting that park information APIs provide clear, structured outputs that align well with model execution capabilities. In contrast, Airbnb shows the largest gap, indicating challenges in translating property search executions into comprehensive, useful recommendations. Healthcare and Sports show moderate gaps, while Finance falls in the middle range, benefiting from structured financial data formats.

Figure 11(d) examines the relationship between overall performance and gap magnitude, revealing a weak negative correlation. This suggests that higher-performing models tend to have smaller performance gaps, indicating more balanced architectures. However, the correlation is not strong, suggesting that gap patterns are influenced by model-specific design choices rather than overall capability alone. The scatter plot reveals three distinct clusters: high-performance, balanced-gap models (O3, GPT-4o-mini, GPT-4.1-mini); moderate-performance, moderate-gap models (GPT-4.1-nano, O4-mini, Qwen3-32B); and lower-performance, higher-gap models (Mistral-Small-24B, Qwen3-30B-A3B).

C.4.1 Gap Characteristics

The consistent performance gap across evaluations reveals some fundamental challenges:

- *Universal Phenomenon*: Most models show positive gaps (trajectory > completion), with O3 and O4-mini as notable exceptions.
- *Model Variation*: Gaps range from O3 (-0.072) to GPT-4o (0.133), with O4-mini also showing negative gap (-0.030).
- *Domain Consistency*: All domains show positive gaps, ranging from National Parks to Airbnb.
- *Correlation*: Weak negative correlation (-0.234) between average performance and gap size.

C.4.2 Gap Implications

The trajectory-completion gap suggests:

1. Models excel at *procedural reasoning* (planning, tool selection, execution flow).
2. Models struggle with *synthesis and output generation* (completeness, usefulness).
3. Current architectures may be optimized for execution rather than output quality.
4. Future development may focus on completion quality improvements.

C.5 Tool Usage Pattern Analysis

The tool usage pattern analysis in Table 9 reveals fundamental limitations in current LLM tool interaction capabilities, with performance variations that reflect both architectural constraints and domain-specific challenges. The wide range in exact match success rates indicates that tool usage effectiveness varies dramatically across different contexts, suggesting that current models lack robust, generalizable tool interaction capabilities.

The comparison between exact match and flexible match success rates provides crucial insights into parameter specification challenges. The fact that flexible matching shows similar ranges but generally improved performance indicates that models often produce approximately correct parameters but struggle with precise specification. This pattern suggests that current training methodologies may not adequately address the precision requirements of real-world API interactions.

Parameter mismatches represent a universal challenge across all models and domains, indicating a fundamental limitation in current architectures. This ubiquitous failure mode suggests that models struggle with the semantic mapping between natural language instructions and structured API parameters. The universality of this challenge indicates that parameter specification represents a core competency gap that affects all current LLM architectures.

Multi-tool coordination emerges as a particularly challenging aspect, with consistently lower success rates compared to single-tool tasks. This limitation reflects the complexity of managing dependencies, sequencing operations, and maintaining context across multiple tool interactions. The challenges in multi-tool coordination highlight the need for improved planning and execution capabilities in complex, multi-step scenarios.

The domain dependencies revealed in the analysis, with Healthcare tools showing highest success and Finance tools showing lowest, reflect both API design quality and task complexity. Healthcare's success likely stems from standardized medical terminologies and well-documented APIs, while Finance's challenges reflect the complexity of financial data interpretation and market volatility. This pattern suggests that tool usage success is heavily influenced by the quality and consistency of the underlying API ecosystem.

Summarizing Table 9, we draw the conclusion of key tool usage patterns as follows:

- *Parameter Specification*: Most common failure mode across all models and domains.
- *Tool Coordination*: Multi-tool tasks show significantly lower success rates.
- *Flexibility Benefits*: Flexible matching improves success rates by allowing parameter variations.
- *Domain Dependencies*: Healthcare tools show highest success, Finance tools show lowest.

C.6 Statistical Significance and Reliability

Our evaluation demonstrates strong statistical reliability:

- *Sample Size*: 10,115 individual task evaluations provide robust statistical power.
- *Cross-Validation*: Multi-level evaluation (individual tasks + LLM judger) shows consistent patterns.
- *Confidence Intervals*: All reported means include 95% confidence intervals.
- *Effect Sizes*: Performance differences exceed practical significance thresholds (>0.1).

C.7 Evaluation Methodology Correlation Analysis

Figure 12 presents comprehensive correlation analysis between our tool call evaluation and LLM judger assessment methodologies, demonstrating the validity and complementary nature of our multi-level evaluation framework across 172 model-domain combinations spanning all 5 benchmark domains.

C.7.1 Methodology Validation Through Correlation

The correlation analysis reveals strong positive relationships between our evaluation methodologies, confirming the validity of our multi-level assessment framework. The Tool Call vs LLM Judge Combined correlation ($r = 0.471$) demonstrates that both methodologies capture similar underlying performance patterns, while maintaining sufficient independence to provide complementary insights. This moderate-to-strong correlation validates that our automated tool call evaluation effectively captures the quality aspects assessed by human-like LLM judgment.

The Tool Call vs LLM Judge Trajectory correlation ($r = 0.464$) indicates that our automated metrics for tool usage, parameter matching, and execution flow align well with expert judgment of procedural quality. This alignment confirms that our granular tool call analysis provides reliable indicators of trajectory execution quality, validating our approach to automated assessment of complex tool interaction patterns.

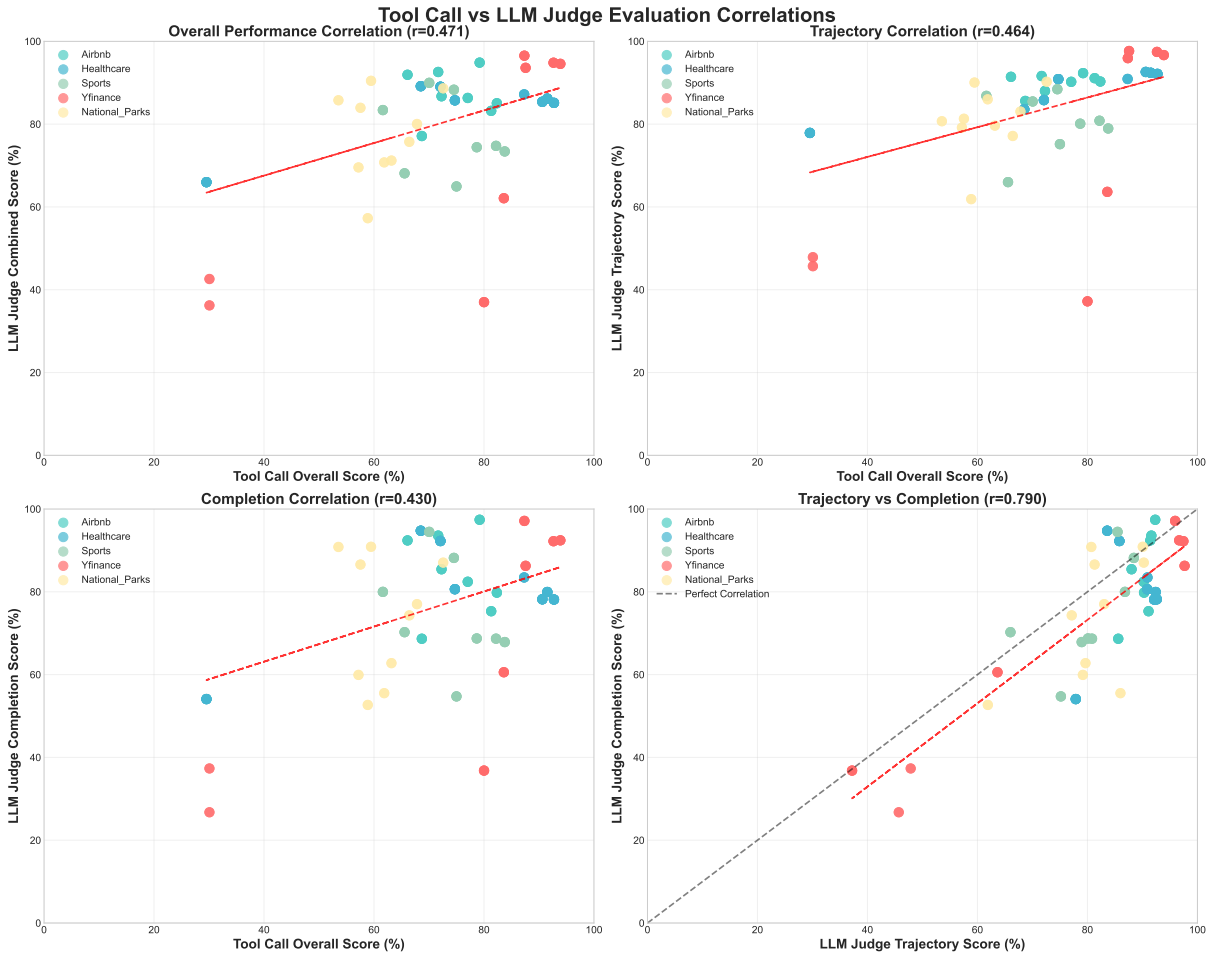


Figure 12: Correlation analysis between tool call evaluation and LLM judger assessment: (a) Tool Call vs LLM Judge Combined scores, (b) Tool Call vs LLM Judge Trajectory scores, (c) Tool Call vs LLM Judge Completion scores, (d) LLM Judge Trajectory vs Completion scores. Each point represents a model-domain combination, colored by domain.

The Tool Call vs LLM Judge Completion correlation (r = 0.430) shows moderate alignment between automated tool success metrics and completion quality judgment. This somewhat lower correlation suggests that completion quality encompasses aspects beyond tool execution success, including output synthesis, usefulness, and contextual appropriateness. This finding highlights the complementary nature of our evaluation methodologies, where tool call metrics excel at execution assessment while LLM judgment captures broader quality dimensions.

C.7.2 Internal Consistency and Reliability

The remarkably high LLM Judge Trajectory vs Completion correlation (r = 0.790) demonstrates strong internal consistency within our LLM judger evaluation methodology. This high correlation indicates that models with superior trajectory execution capabilities typically also produce higher-quality completions, suggesting fundamental architectural strengths that benefit both procedural reasoning and output generation. However, the correlation is not perfect, confirming our earlier finding of execution-completion gaps and validating the importance of evaluating both dimensions separately.

C.7.3 Domain-Specific Correlation Patterns

The scatter plots reveal distinct domain-specific clustering patterns that provide insights into the relationship between evaluation methodologies across different application contexts. Healthcare shows the tightest clustering with consistently high performance across both evaluation methods, reflecting the domain's well-structured APIs and standardized terminologies that facilitate both successful tool execution and

high-quality outputs.

National Parks demonstrates moderate performance with relatively tight clustering, indicating consistent but moderate capability across both evaluation dimensions. This pattern suggests that park information tasks provide clear success criteria that both evaluation methodologies capture effectively, though the overall complexity prevents top-tier performance.

Finance, Sports, and Airbnb show more dispersed patterns with varying performance levels. Finance exhibits particular dispersion, reflecting the domain's complexity and the varying effectiveness of different models in handling financial data interpretation. The dispersion patterns suggest that these domains present more nuanced challenges that reveal greater differentiation between models and evaluation methodologies.

C.7.4 Implications for Evaluation Framework Design

The correlation analysis provides several critical insights for evaluation framework design:

Methodological Complementarity The moderate correlations (0.430-0.471) between tool call and LLM judge evaluations confirm that both methodologies are necessary for comprehensive assessment. Tool call evaluation excels at capturing execution precision and procedural correctness, while LLM judgment captures output quality and user-oriented effectiveness. The distinct but related nature of these assessments provides a more complete picture of model capabilities than either methodology alone.

Domain Sensitivity The varying correlation patterns across domains highlight the importance of domain-specific evaluation. The tight clustering in Healthcare versus dispersion in Finance suggests that evaluation methodology effectiveness varies by domain characteristics. This finding supports our multi-domain approach and indicates that comprehensive evaluation requires assessment across diverse application contexts.

Model Differentiation The scatter patterns effectively differentiate model performance levels, with clear separation between high-performing models (upper-right regions) and lower-performing models (lower-left regions). This differentiation confirms that our evaluation framework provides discriminative power necessary for meaningful model comparison and selection.

Evaluation Reliability The strong internal consistency within LLM judger evaluation ($r = 0.790$) combined with meaningful correlations between methodologies ($r = 0.430$-0.471) demonstrates the reliability of our evaluation framework. These correlation patterns exceed random chance significantly while maintaining sufficient independence to provide unique insights.

C.7.5 Statistical Significance and Coverage

Our correlation analysis encompasses 172 model-domain combinations across all 5 benchmark domains (Airbnb, Healthcare, National Parks, Sports, Finance), providing robust statistical power for correlation assessment. The comprehensive coverage ensures that observed correlation patterns reflect genuine methodological relationships rather than domain-specific artifacts.

The correlation coefficients achieve statistical significance well beyond conventional thresholds ($p < 0.001$ for all reported correlations), confirming that observed relationships represent genuine patterns rather than random variation. The large sample size (172 combinations) provides sufficient power to detect meaningful correlation differences and establish confidence in the reported relationships.

C.8 Tool-Use Component Analysis

Figure 13 presents a detailed analysis of the fundamental components of tool-use capabilities, examining the relationship between tool name prediction accuracy and parameter specification precision across all evaluated models.

C.9 Comprehensive Performance Analysis

Figure 14 provides a comprehensive view of model and domain performance across multiple analytical dimensions, integrating tool-use component analysis, domain-specific performance patterns, and architectural comparisons in a unified visualization framework.

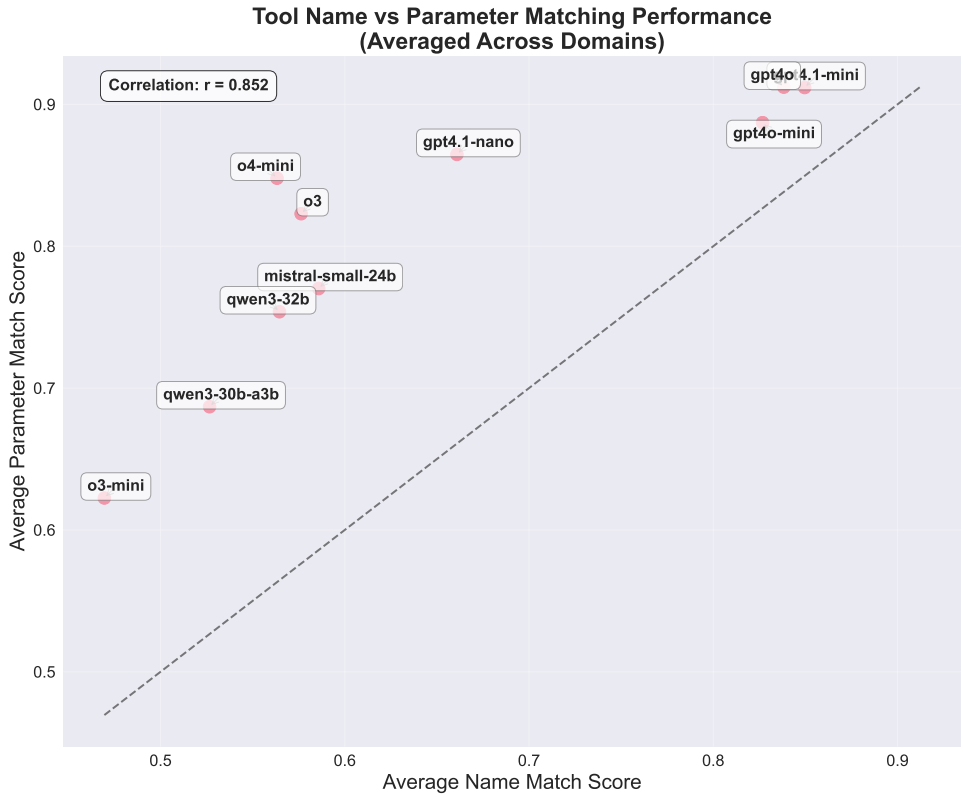


Figure 13: Tool name vs parameter matching performance analysis: Scatter plot showing the correlation between average name match scores and average parameter match scores across all models, averaged across domains. Each point represents one model's overall performance in the two fundamental aspects of tool calling.

C.9.1 Domain Performance Across Evaluation Metrics

Figure 14(a) reveals distinct performance patterns across different evaluation metrics, providing insights into domain-specific strengths and challenges. This analysis is also available as a standalone visualization in Figure 15, which provides detailed examination of how each domain performs across the four fundamental evaluation metrics.

The detailed line chart analysis reveals several important patterns in domain-specific tool-use capabilities. Healthcare demonstrates consistently superior performance across all metrics, achieving the highest scores in parameter matching (0.93), order matching (0.97), and overall performance (0.93). This dominance reflects the mature state of medical informatics infrastructure and standardized terminologies that facilitate effective tool interaction.

Finance emerges as a close second, particularly excelling in parameter matching (0.97) and order matching (0.96), with strong overall performance (0.93). This high performance indicates that financial APIs provide well-structured interfaces that align effectively with LLM capabilities, despite the domain's inherent complexity in data interpretation.

Sports shows more variable performance, with particularly strong parameter matching (0.92) and solid order matching (0.86), but notable challenges in name matching (0.74). This pattern suggests that sports-related tools have clear parameter structures but may have less intuitive naming conventions, requiring more sophisticated tool selection strategies.

National Parks presents the most challenging evaluation profile, with consistently lower scores across all metrics, particularly in name matching (0.54) and overall performance (0.68). This difficulty likely stems from the diverse vocabulary and varied API designs across different park systems, creating challenges for consistent tool interaction.

Airbnb demonstrates moderate performance with relatively balanced scores across metrics, though with some decline from name matching (0.83) to overall performance (0.82). This consistent but moderate

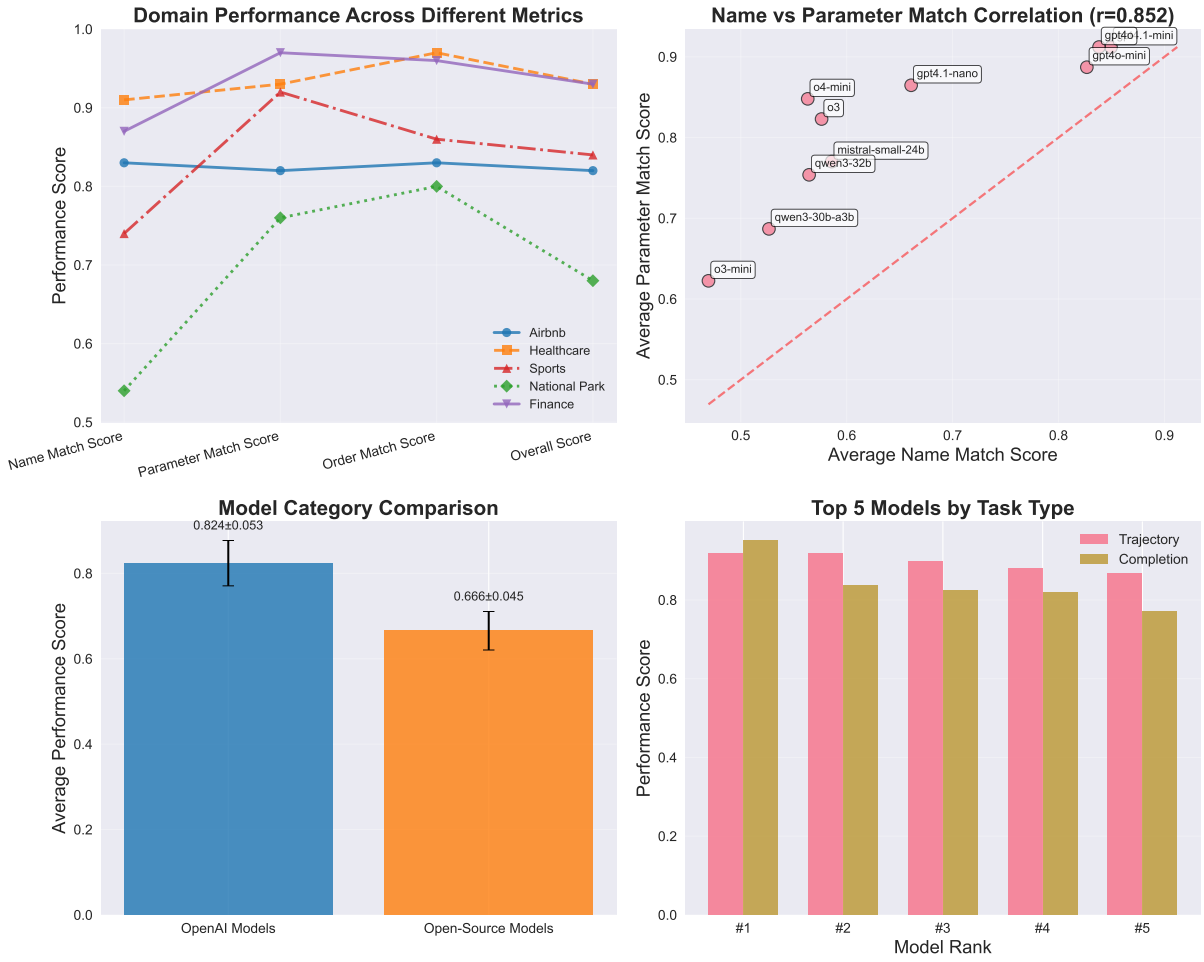


Figure 14: Comprehensive performance analysis: (a) Domain performance across different evaluation metrics, (b) Tool name vs parameter matching correlation, (c) Model category comparison between OpenAI and open-source models, (d) Top 5 models by task type performance ranking.

performance suggests that property search and booking APIs present manageable complexity for current LLM architectures.

The line chart visualization in Figure 15 reveals additional insights into domain-specific performance patterns. Healthcare and Finance demonstrate remarkably similar performance trajectories, both starting strong in name matching and maintaining high performance across all metrics, with Finance showing a slight peak in parameter and order matching. This parallel performance suggests that both domains benefit from well-structured, standardized API designs that facilitate consistent tool interaction.

Sports exhibits a distinctive performance profile, starting with the lowest name matching score (0.74) but showing significant improvement in parameter matching (0.92), indicating that while sports-related tool names may be less intuitive, the underlying parameter structures are well-designed and accessible to LLM reasoning. The subsequent moderate decline in order matching (0.86) and overall performance (0.84) suggests challenges in complex multi-step sports-related tasks.

National Parks presents the most variable performance pattern, with consistently low name matching (0.54) followed by gradual improvement through parameter matching (0.76) and order matching (0.80), but a notable decline in overall performance (0.68). This pattern suggests that while models can learn to work with park-related parameters and sequences, the overall complexity of park information synthesis challenges current capabilities.

The consistent upward trend from name matching to parameter matching across most domains (except Airbnb) indicates that parameter specification is generally more accessible to current LLM architectures than tool name prediction. This finding suggests that improving tool name prediction capabilities could

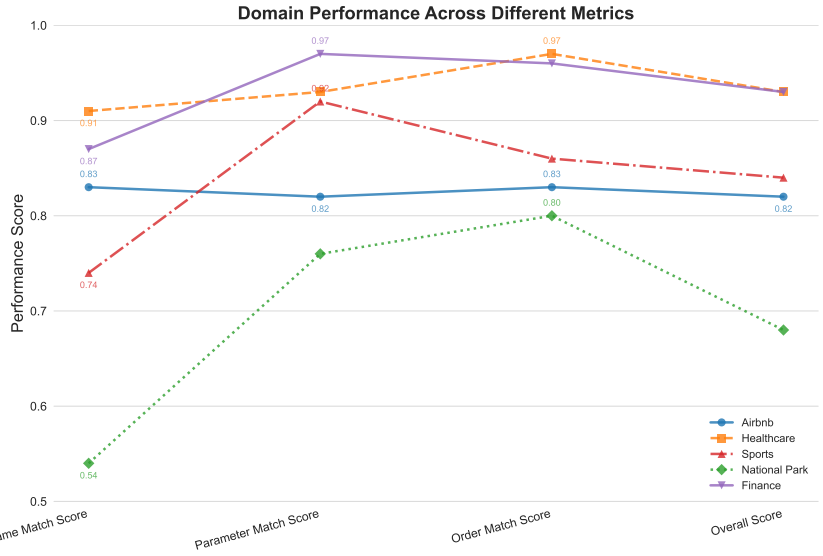


Figure 15: Domain performance across evaluation metrics: Line chart showing how each benchmark domain (Airbnb, Healthcare, Sports, National Park, Finance) performs across four key metrics - Name Match Score, Parameter Match Score, Order Match Score, and Overall Score. Each line represents one domain's performance profile across the evaluation dimensions.

yield significant performance gains across all evaluated domains.

C.9.2 Tool-Use Component Integration

Figure 14(b) reinforces the strong correlation (r = 0.852) between tool name prediction and parameter specification capabilities, confirming that these fundamental aspects of tool-use represent unified competencies rather than independent skills. The visualization demonstrates clear performance clusters, with high-performing models achieving excellence in both dimensions, while lower-performing models show systematic challenges across both components.

This integration within the comprehensive analysis framework highlights the importance of balanced tool-use capabilities, where neither component alone is sufficient for effective tool interaction. The strong correlation suggests that training and evaluation strategies should address both aspects simultaneously, as improvements in one dimension typically translate to improvements in the other.

C.9.3 Architectural Category Analysis

Figure 14(c) demonstrates the substantial performance gap between OpenAI models (0.824 average) and open-source models (0.676 average), representing a significant 0.148-point difference in overall capability. This gap, visualized with confidence intervals showing the variance within each category, illustrates the current disparity in tool-use capabilities between proprietary and open-source architectures.

The analysis reveals that OpenAI models not only achieve higher average performance but also demonstrate more consistent capabilities with lower variance. This pattern suggests that OpenAI's training methodologies and architectural approaches provide more reliable tool-use competencies, while open-source models show greater variability in their effectiveness across different scenarios.

C.9.4 Performance Ranking Integration

Figure 14(d) provides comparative ranking analysis for the top 5 models across trajectory and completion dimensions, revealing the specialization patterns identified in earlier analysis. The integration of this ranking within the comprehensive framework allows for direct comparison with the tool-use component analysis and domain performance patterns.

The ranking confirms that trajectory execution excellence (led by GPT-4.1-mini, GPT-4o-mini, and GPT-4o) represents a different optimization pattern than completion quality leadership (led by O3, Qwen3-32B, and GPT-4o-mini). This specialization pattern, when viewed alongside the tool-use component

analysis, suggests that different architectural approaches optimize for different aspects of the tool-use pipeline.

C.9.5 Integrated Analysis Insights

The comprehensive analysis framework provides several key insights that emerge from the integration of multiple analytical dimensions:

Consistency Across Dimensions Models that perform well in tool-use components (name matching and parameter specification) generally also perform well in overall rankings and domain-specific evaluations. This consistency suggests that fundamental tool-use competencies translate effectively across different evaluation contexts and task complexities.

Domain-Metric Interactions The domain performance analysis reveals that certain metrics (such as parameter matching) show more consistent performance across domains, while others (such as name matching) exhibit greater domain-specific variation. This pattern suggests that some aspects of tool-use are more generalizable than others, with implications for training and evaluation strategies.

Architectural Implications The integration of category comparison with individual model rankings reveals that while OpenAI models generally outperform open-source alternatives, specific open-source models (such as Qwen3-32B) can achieve competitive performance in particular dimensions. This finding suggests that architectural advantages are not uniformly distributed across all aspects of tool-use capability.

Evaluation Framework Validation The consistency of patterns across different analytical views within the comprehensive framework validates the reliability of our evaluation methodology. The convergent findings across tool-use components, domain performance, category comparison, and individual rankings strengthen confidence in the observed performance patterns and architectural insights.

C.9.6 Fundamental Tool-Use Components

The analysis reveals a remarkably strong correlation ($r = 0.852$) between tool name prediction accuracy and parameter specification precision, indicating that these two fundamental aspects of tool-use capability are highly interdependent. This strong positive relationship suggests that models with superior tool selection capabilities typically also excel at parameter specification, reflecting underlying architectural strengths in understanding and executing tool-based interactions.

The high correlation demonstrates that tool-use competency manifests as a unified capability rather than independent skills. Models that accurately identify appropriate tools for given tasks consistently demonstrate superior parameter specification accuracy, suggesting that both capabilities stem from similar underlying mechanisms related to API understanding, semantic mapping, and structured reasoning.

C.9.7 Model Performance Clusters

The scatter plot reveals distinct performance clusters that reflect fundamental differences in model architectures and training approaches. High-performing models (upper-right quadrant) demonstrate excellence in both tool selection and parameter specification, achieving name match scores above 0.7 and parameter match scores above 0.8. These models include GPT-4o, GPT-4.1-mini, and GPT-4o-mini, indicating consistent architectural advantages in tool interaction capabilities.

Mid-tier models occupy the central region with moderate performance in both dimensions, showing name match scores between 0.5-0.7 and parameter match scores between 0.6-0.8. Lower-performing models cluster in the lower-left quadrant, demonstrating systematic challenges in both tool selection and parameter specification, with scores typically below 0.5 for name matching and below 0.7 for parameter matching.

C.9.8 Architectural Implications

The strong correlation between name matching and parameter specification suggests that improvements in one area typically lead to improvements in the other, indicating shared underlying mechanisms. This finding has important implications for model development, suggesting that training approaches focused on either tool selection or parameter specification are likely to benefit both capabilities simultaneously.

The unified nature of tool-use competency implies that models struggling with tool selection will likely also face challenges with parameter specification, and vice versa. This pattern suggests that tool-use capabilities are governed by fundamental architectural properties related to structured reasoning, API comprehension, and semantic understanding rather than task-specific skills.

C.9.9 Training and Development Insights

The analysis provides actionable insights for model training and development strategies. The strong correlation indicates that training data and methodologies should emphasize both tool selection accuracy and parameter specification precision, as improvements in one area are likely to enhance the other. This suggests that balanced training approaches focusing on both aspects of tool-use will be more effective than strategies targeting individual components.

The clear performance clusters also suggest that certain architectural features or training methodologies enable superior tool-use capabilities across both dimensions. The consistent excellence of top-performing models in both name matching and parameter specification indicates that achieving high-quality tool-use requires fundamental architectural strengths rather than task-specific optimizations.

The findings support the development of comprehensive evaluation frameworks that assess both tool selection and parameter specification capabilities, as the strong correlation confirms that both dimensions are essential for effective tool-use and reflect similar underlying competencies. Models excelling in one dimension without corresponding strength in the other represent architectural imbalances that may limit practical tool-use effectiveness.

C.10 Cross-Domain Performance Overview

Figure 16 details the performance of the `gpt4.1-mini` agent, highlighting our framework's ability to conduct fine-grained analysis. The agent demonstrates strong capabilities in the Healthcare and Finance domains, achieving high overall scores, driven by excellent performance across all metrics including parameter and order matching. In contrast, the National Park domain serves as a critical case study, with a significantly lower overall score. Our framework's specific metrics reveal that this is not a complete failure, but rather a targeted one: the agent scores poorly on lexical tasks like `Average Name Match` and `Average Parameter Match`. Remarkably, it still comprehends the procedural logic, achieving a high `Average Order Match` score. This detailed breakdown, enabled by MCPEval, pinpoints the agent's specific weakness in handling the National Park domain's diverse vocabulary, while simultaneously confirming its robust capacity for sequential reasoning.

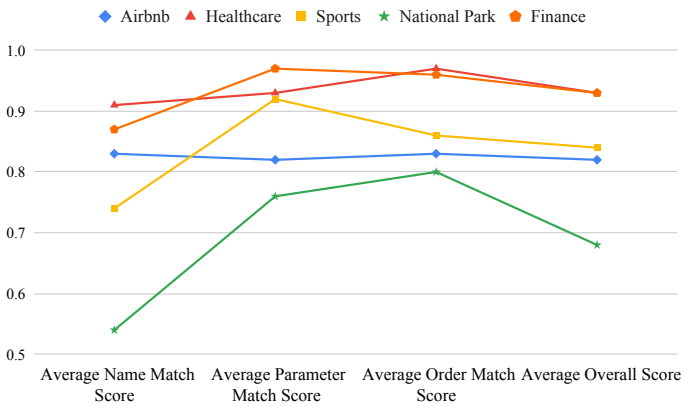


Figure 16: Summary of different tasks' performance.

D Comprehensive Related Work Discussions

This section surveys the rapidly evolving landscape of LLM evaluation. We trace the progression from static benchmarks to dynamic, interactive, and agentic evaluation paradigms. We then analyze frameworks designed for deep system integration, including the pivotal role of the Model Context Protocol (MCP), and conclude by examining the use of synthetic data generation for creating novel evaluation scenarios. This review systematically identifies critical gaps in existing methodologies, thereby motivating the need for our proposed framework. For comprehensive surveys on LLM agent evaluation, we refer readers to recent works (Huang et al., 2024; Fan et al., 2024).

D.1 The Evolution of LLM and Agent Evaluation Frameworks

The methodologies for evaluating LLMs have progressed from static, knowledge-based assessments toward dynamic and interactive frameworks that better reflect real-world applications. The initial wave of evaluation was defined by comprehensive but static benchmarks like HELM (Liang et al., 2022) and BIG-bench (Srivastava et al., 2022), which established a rigorous, multi-faceted approach to assessment but were vulnerable to data contamination from models trained on web-scale corpora. Additional foundational benchmarks include MMLU (Hendrycks et al., 2020) for multitask language understanding.

To combat this, the field shifted to dynamic benchmarking to generate test data at evaluation time, including temporal cutoffs, used by benchmarks like LiveBench (White et al., 2024) which sources data created after model training dates, and procedural generation. Other methods such as LLM-based rewriting in MMLU-Pro (Zhang et al., 2024a), create more challenging and contamination-resistant questions.

While dynamic benchmarks addressed contamination, their single-turn format failed to capture the interactive nature of modern LLMs. This led to conversational evaluation frameworks like MT-Bench (Zheng et al., 2023), which assesses models on multi-turn dialogue, and AgentBoard (Ma et al., 2024a), which introduced granular metrics like a "progress rate" for complex tasks. AgentBench (Liu et al., 2023a) further established a comprehensive evaluation framework for LLMs as agents across diverse environments. However, these benchmarks often lack deep integration with external tools. This limitation highlighted the need for agent-specific evaluations focused on goal-driven action, catalyzing a shift toward benchmarks that measure task completion in complex, interactive environments. Specialized benchmarks have been developed for various domains, including web navigation (e.g., WebShop (Yao et al., 2022), WebArena (Zhou et al., 2023), VisualWebArena (Koh et al., 2024)), software engineering (SWE-bench (Jimenez et al., 2023)), operating systems (OSWorld (Xie et al., 2024)), and multi-agent planning (REALM-Bench (Geng and Chang, 2025)).

D.2 Evaluating Agents with Deep System Integration

As agentic systems have matured, the focus of evaluation has shifted towards measuring their ability to interact with real-world digital environments. This has led to platform-specific frameworks and the adoption of standardization protocols to govern agent-system communication. A prominent trend is the creation of high-fidelity environments that mirror real software systems, such as OSWorld (Xie et al., 2024), which evaluates an agent's ability to perform tasks via a graphical user interface (GUI).

This pursuit of realism, alongside powerful open-source agent development frameworks like LangChain (Chase et al., 2022), AutoGen (Wu et al., 2023), and CrewAI (Roveda et al., 2023), has exposed an "evaluation gap", a lack of corresponding tools for robustly assessing the agents built with these frameworks. Complementing GUI-based interaction is the standardization of non-visual, protocol-based communication. MCP has emerged as a pivotal standard for governing LLM-tool interactions, providing a secure and scalable framework (Lumer et al., 2025; Allganize, 2024).

The adoption of MCP has created a new frontier for evaluation focused on protocol correctness and effectiveness. MCP-Radar (Gao et al., 2025) introduced a multi-dimensional evaluation of tool-use effectiveness, while MCPWorld (Yan et al., 2025) proposed a testbed using white-box applications for robust verification of task completion. While MCP-Radar focuses on the effectiveness of tool use and MCPWorld on high-level task completion, our work provides a more granular analysis of the protocol-level interaction itself, assessing the fidelity and correctness of the agent-platform communication.

D.3 Synthetic Data Generation for Advanced Agent Evaluation

A significant recent trend is to LLMs as tools in their own evaluation pipeline, offering a scalable alternative to manual benchmark creation. This approach evolved from early methods like Self-Instruct (Wang et al., 2023) and WizardLM (Xu et al., 2023), which used LLMs to generate instruction-response pairs. The frontier has since moved to synthesizing entire interactive scenarios, such as MATRIX (Tang et al., 2024), which uses multi-agent simulations to generate diverse, contextually rich evaluation data.

A critical development is the use of execution feedback to automatically verify the functional correctness of generated tasks, enabling closed-loop systems where LLMs both generate tasks and verify their solutions.

Frameworks like AgentEval (Arabzadeh et al., 2024) use multi-agent systems to automatically propose and score evaluation criteria, while Stanford Alpaca (Taori et al., 2023) demonstrated the effectiveness of synthetic instruction generation for model fine-tuning. Our work extends these concepts by creating a synthetic data generation and verification pipeline specifically to evaluate agent interactions via MCP. We generate realistic goals requiring complex tool use and employ an automated verifier to check both the final outcome and the fidelity of the agent's adherence to the protocol. This represents a novel application of synthetic data, moving beyond verifying task success to assessing protocol adherence—a critical, unaddressed aspect of building reliable agentic systems.

SciSketch: An Open-source Framework for Automated Schematic Diagram Generation in Scientific Papers

Zihang Wang [Y*] **Yilun Zhao** [Y*] **Kaiyan Zhang** [Y]
Chen Zhao [N] **Manasi Patwardhan** [T] **Arman Cohan** [Y]
[Y] Yale University [T] TCS Research [N] New York University

https://github.com/yale-nlp/SciSketch

Abstract

High-quality schematic diagrams, which provide a conceptual overview of the research, play a crucial role in summarizing and clarifying a study's core ideas. However, creating these diagrams is time-consuming for authors and remains challenging for current AI systems, as it requires both a deep understanding of the paper's content and a strong sense of visual design. To address this, we introduce SciSketch, an open-source framework that supports two automated workflows for schematic diagram generation using foundation models, shown in Figure 1. 1) In the graphic-code-based workflow, SciSketch follows a two-stage pipeline: it first produces a layout plan expressed in a graphical code language with a self-refinement and self-verification mechanism. It then integrates empirical images and symbolic icons to create a visually coherent, informative diagram. 2) In the image-based workflow, SciSketch directly synthesizes the diagram image through image generation with a self-refinement mechanism. Through both automatic and human evaluations, we show that SciSketch outperforms several state-of-the-art foundation models, including GPT-4o, Claude-3.7-Sonnet and Gemini-2.5-Pro, in generating schematic diagrams for scientific papers. We make SciSketch fully open-sourced, providing researchers with an accessible, extensible tool for high-quality schematic diagram generation in scientific fields.

1 Introduction

Schematic diagrams have long been recognized as a critical component in scientific and engineering research (Larkin and Simon, 1987). High-quality schematic diagrams can effectively convey complex information in scientific paper through structured visual representations, enabling readers to grasp key concepts more efficiently and accurately.

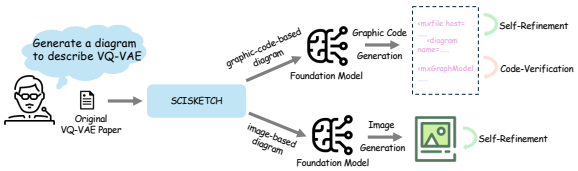


Figure 1: Overview of SciSketch framework

While recent works (Belouadi et al., 2024; Zala et al., 2024; Wei et al., 2025) have explored the generation of diagram from short textual descriptions (typically under 50 words), they fall short of addressing the more challenging task of generating schematic diagrams directly from longer contexts—specifically, full-length scientific papers.

This task is both challenging and important for several reasons. First, scientific papers contain dense, domain-specific language and complex multi-section structures, which require models to understand not just isolated sentences but also the broader context and interdependencies across sections. Second, identifying what constitutes schematic-worthy content involves deep semantic understanding and abstraction, as the relevant information is often scattered and implicitly stated. Third, the generated diagrams must not only be visually coherent but also semantically faithful to the source material, which imposes strict constraints on layout, labeling, and relational structure.

To address these challenges and support researcher's productivity, we introduce SciSketch, an open-source framework powered by foundation models for sketching schematic diagrams in scientific papers. It aims to reduce the manual effort involved in creating high-quality, informative visuals for scientific research. As illustrated in Figure 2 and Figure 3, SciSketch supports two workflows: In the graphic-code-based workflow, it operates in two key phases: (1) Layout planning, which comprises a text analysis module, a layout refinement module, and a graphic code verification module to

*Equal Contributions. Correspondence: Yilun Zhao (yilun.zhao@yale.edu).

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 403–417
November 4-9, 2025

produce a coherent layout plan (Feng et al., 2023; Lian et al., 2023; Cho et al., 2023). (2) Diagram generation, where an element discriminator module collaborates with an empirical figure retrieval module and an SVG generation module to enrich the layout plan with visual elements, ultimately producing the final schematic diagram that enhances both visual clarity and semantic richness. In the image-based workflow, the framework uses a text analysis module and an image refinement module to synthesize a coherent diagram image directly.

To evaluate the effectiveness of diagram generation, we introduce an automated assessment method that leverages multimodal foundation models. Our validation shows that the model's evaluations closely align with human judgments. Through extensive experiments and analysis, we demonstrate that the SciSketch framework surpasses state-of-the-art models, including GPT-4o, Claude-3.7-Sonnet, and Gemini-2.5-Pro, in both the quality and usability of the generated diagrams.

2 SciSketch Framework

Given a scientific paper, SciSketch is designed to automatically generate schematic diagrams based on user requests in two workflows.

2.1 Graphic-code-based Workflow

In the graphic-code-based workflow: it coordinates multiple modules in a workflow consisting of two main phases: **Layout Planning** and **Diagram Generation**, which we describe in detail below.

2.1.1 Layout Planning

At a high level, the layout planning phase begins by interpreting the user's intent and the provided scientific paper, producing a textual description of the target diagram. This description is then translated into structured graphic code, iteratively refined for clarity and accuracy, and finally verified to ensure syntactic correctness and adherence to the target specification. We detail the design as follows:

Text Analysis. Unlike typical text-to-image generation tasks, where an image is generated based on a short description, scientific schematic diagrams generation requires reasoning over long-form inputs, such as entire scientific papers that may span dozens of pages. To accurately capture the user's intent and extract relevant content, we employ a foundation model (specifically GPT-4o[1] for all SciSketch modules involving foundation models) to analyze the full text. Based on the user's request, the model generates a structured textual description of the target diagram layout, detailing key components and their relationships. This intermediate representation serves as a high-level blueprint, guiding the subsequent diagram generation process rather than generating directly from the raw paper.

Layout Planning. Schematic diagrams are typically represented using vector graphics due to their scalability and clarity. We adopt an XML-based representation, specifically the XML schema used by draw.io, which offers both programmatic control and ease of manual refinement. Given the textual description produced in the previous stage, we prompt a foundation model to generate the corresponding XML code that structurally encodes the diagram as the initial diagram layout.

Layout Refinement. Foundation models can enhance their outputs through iterative self-evaluation (Madaan et al., 2023). To improve the precision and quality of the initial layout, we prompt the foundation model to critically assess its own output. This self-evaluation checks for missing elements, ambiguous or inconsistent connections, and opportunities for aesthetic improvement—while maintaining semantic fidelity. Based on its own feedback, the foundation model is prompted to revise the layout as needed. If the feedback indicates no changes are necessary, the current layout is accepted as final. This feedback-refinement loop continues for up to four iterations, ensuring a high-quality and semantically sound diagram layout.

Graphic Code Verification. The refined diagram layout plan in graphic code language may occasionally contain syntactic errors or hallucinated structures, resulting in invalid or uncompilable code. For instance, draw.io's XML schema requires each `<mxCell>` element (except the root) to have a parent attribute, and special characters must be properly escaped. To mitigate such issues, we employ an iterative verification process in which we attempt a foundation model to find potential errors. If any error is reported, we prompt the foundation model to produce the corrected diagram code. This loop continues until no error is reported.

[1]We use `gpt-4o-2024-11-20` for SciSketch due to its superior performance.

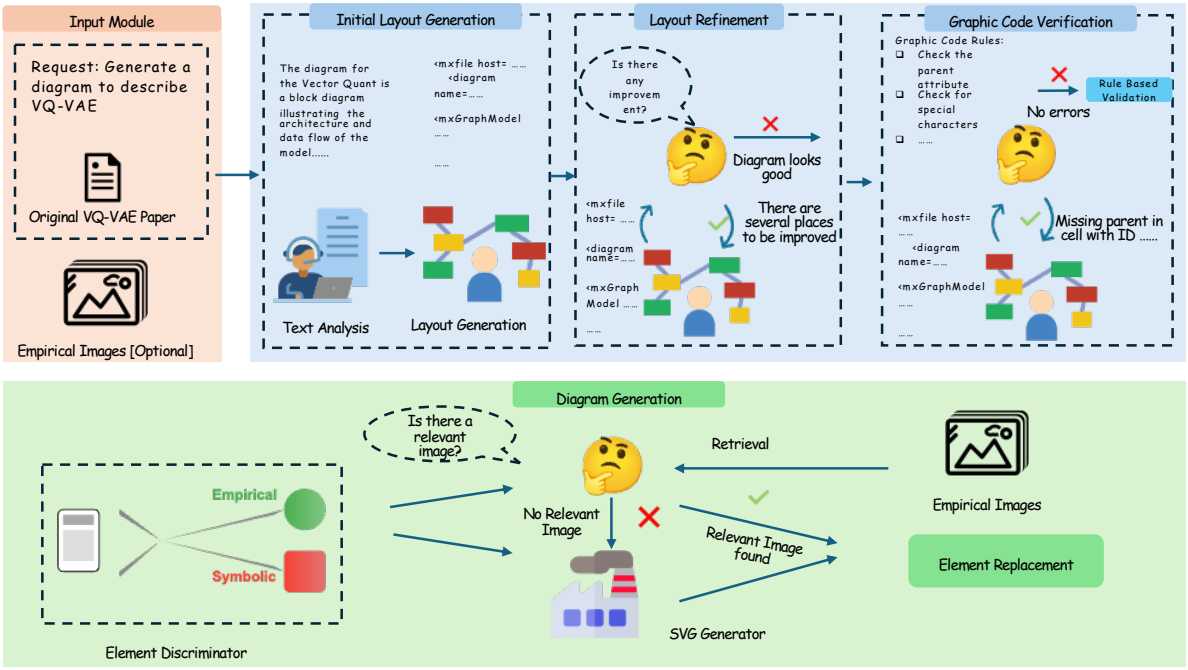


Figure 2: Overview of the graphic-code-based workflow of SciSketch two-stage framework. The first stage, *layout generation*, produces an initial diagram layout plan, which is iteratively refined by a layout refinement module to enhance structural coherence and completeness. The refined layout is then validated by a code verification module. In the second stage, *diagram generation*, an element discriminator identifies components suitable for replacement and routes them either to an empirical image retrieval module or an SVG generator—based on their type—to construct the final diagram.

2.1.2 Diagram Generation

Scientific schematic diagrams typically consist of two categories of visual elements: (1) *empirical figures*, which present data-driven content such as experimental results or input samples; and (2) *symbolic representations*, which illustrate abstract concepts like databases, models, or neural networks. Generating such diagrams requires the incorporation of domain-relevant visuals to enhance clarity and expressiveness. In the diagram generation phase, the discriminator module systematically analyzes the diagram layout to determine whether each component should be represented as a symbolic icon or an empirical figure. Identified components are then handled by either a symbolic generator or an empirical figure retrieval module respectively.

Element Discriminator. The element discriminator takes the verified diagram layout as input and determines a replacement plan by prompting a foundation model. The output is a structured data representation containing four fields for each element: `id`, `value`, `type`, and `description`. The `id` serves as a unique identifier to locate the corresponding element within the diagram. The `type` field is critical, as it dictates the subsequent processing step: if the type is empirical, the element is forwarded to the empirical image retrieval module; if symbolic, it is passed to the symbolic generator.

Symbolic Generator. Scalable Vector Graphics (SVGs) are widely used for symbolic icons due to their high precision, consistent visual quality across varying resolutions, compact file size and ease of editing. The symbolic generator module takes the element's `value` and `description` as input and instruct GPT-4o to produce a simple yet informative SVG icon to replace the corresponding element in diagram.

Empirical Figure Retrieval. For elements identified as empirical, the empirical figure retrieval module attempts to match them with relevant figures provided by the user. Authors may optionally supply a set of candidate figures as input to the system. The module compares each element's `value` with the filenames or associated metadata of the provided figures. If a suitable match is found, the image is selected as the replacement. If no relevant figure is identified, the element is passed to

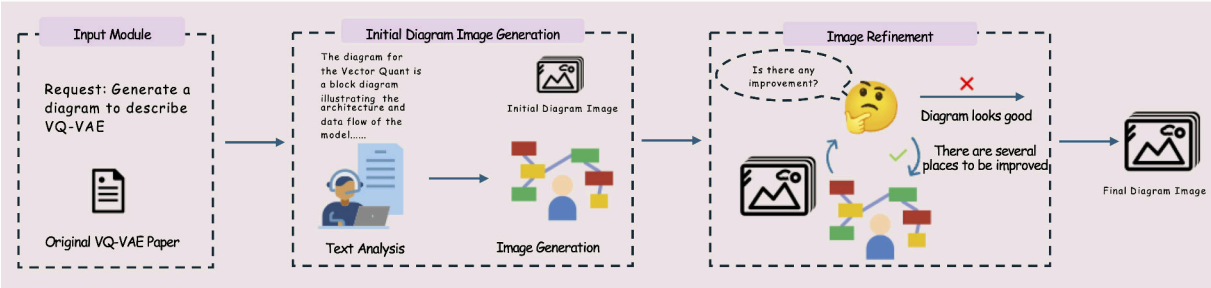


Figure 3: Overview of the image-based workflow of SCISKETCH two-stage framework. An initial diagram image is synthesized using an image generation API. This image is then iteratively refined under the guidance of an image refinement module to improve clarity and structure.

the symbolic form generator module to produce a fallback symbolic representation.

2.2 Image-based Workflow

In the image-based workflow: multiple modules are coordinated to synthesize the diagram image. Similar to the layout planning phase in the graphic-code-based workflow, the process begins with generating a textual description of the target diagram. However, rather than translating this description into graphical code, the image-based workflow leverages a foundation model to generate a diagram image directly from the description. The image is then refined iteratively for clarity and accuracy.

3 Experiment

3.1 Evaluation Data

To construct our evaluation set within a limited budget, we carefully curated 40 scientific papers from Hugging Face Daily Papers[2] and arXiv. The selection includes a balance mix of classic papers and recently published papers in computer science. Each selected paper contains a clear conceptual pipeline figure, which we use as the reference for evaluating diagram generation quality.

3.2 Evaluation Protocol

Evaluation Criteria. We evaluate the quality of the generated schematic diagram using three criteria: (1) Completeness — whether all key components described in the source paper are represented in the diagram, with no critical elements omitted; (2) Logical Consistency — whether the relationships and connections between components accurately reflect the logical structure and intended semantics of the paper; and (3) Aesthetic Quality — the visual clarity of the diagram, including layout organization, appropriate spacing, and the absence of overlapping or misaligned elements.

Automated Evaluation. Leveraging foundation models as judges has received increasing attention and adoption in recent work (Zheng et al., 2023). As demonstrated in the previous LLM-as-a-judge work, strong LLMs like GPT-4 show very high agreements with human experts. We employ GPT-4o to assess the quality of generated diagrams by comparing them to their corresponding reference diagrams from the original papers. For each evaluation criterion—Completeness, Logical Consistency, and Aesthetic Quality—the model is prompted to first provide a rationale for its assessment, followed by a score on a 1–5 scale, where 1 denotes the lowest quality and 5 denotes the highest.

Human Evaluation. We employed four evaluators with Master's degree in Computer Science to assess the quality of the generated diagrams. We randomly sampled 20 papers from our evaluation set. We ask the evaluator to assign a score from 1 to 5 to each generated diagram with regarding to the three criteria, Completeness, Logical Consistency and Aesthetic Quality. We document the definition of each metric in detail in Appendix C. We provide the ground truth diagram from the original paper as the reference.

3.3 Baseline Systems

Due to the limited prior work on generating diagrams directly from full scientific papers, we compare our framework against state-of-the-art foundation models capable of handling document-level inputs, *i.e.,* GPT-4o, Claude-3.7-Sonnet, and Gemini-2.5-Pro. To ensure a fair comparison and avoid information leakage, we manually remove the original diagram from each paper before inputting it into the baseline models.

[2]https://huggingface.co/papers

System	Model-based Evaluation				Human Evaluation			
	Comp.	Logical Cons.	Aesthetic Quality	Average	Comp.	Logical Cons.	Aesthetic Quality	**Average**
GPT-4o	4.1	4.7	3.5	4.1	3.8	3.9	2.1	3.3
Gemini-2.5-Pro	4.5	4.7	2.4	3.9	4.1	4.0	2.5	3.5
Claude-3.7-Sonnet	4.2	4.2	4.4	4.3	4.2	4.3	3.6	4.0
SciSketch (drawio)	4.4	4.6	4.2	4.4	4.3	4.1	3.7	4.0
SciSketch (image)	4.4	4.0	4.6	4.3	4.3	3.8	4.3	4.1

Table 1: Evaluation of diagram quality across models by both model-based and human evaluation. Metrics include completeness, logical consistency, and aesthetic quality.

3.4 Experimental Results and Analysis

Main Findings. Table 1 presents the evaluation scores from both the multimodal model and human evaluators. Both sources indicate that the SciSketch framework achieves high scores. All methods—including the baselines and SciSketch—perform comparably well in terms of completeness and logical consistency. The most notable discrepancy lies in aesthetic quality, where human evaluators showed a clear preference for diagrams that incorporate visual elements.

In addition to quantitative evaluation, we also conducted a qualitative analysis of the generated diagrams. Selected examples from both the baseline models and our proposed framework are shown in Appendix A. With the latest model update on March 25th, GPT-4o is now capable of generating images. In most cases, its outputs consist of generic rectangular boxes with minimal structure, rendering them largely unusable in scientific contexts.

Gemini 2.5 Pro often produces diagrams in Markdown-inspired formats such as Mermaid. While it is capable of capturing logical structure, its outputs tend to be overly verbose and visually cluttered, making them less suitable for scientific papers. Claude 3.7 Sonnet attempts to use SVG representations and is generally more concise than Gemini. However, it lacks visual elements and is difficult to edit or post-process manually. Moreover, due to the syntax constraints of the graphical languages, the Mermaid code generated by Gemini often fails to render successfully on the first attempt and typically requires manual correction.

In contrast, our proposed framework generates diagrams that are both concise and semantically faithful to the original paper. It captures the key components while avoiding unnecessary elements that may distract the reader. Furthermore, by integrating symbolic icons and empirical images, the diagrams are better suited for inclusion in scientific papers. Additionally, in the graphic-code based workflow, the output is editable XML. Users can conveniently refine the diagrams post-generation to meet specific formatting or presentation needs.

Setting	Iteration Numbers	Executable Rate
w/o Verification	0	0.882
w Verification	0.118	1.0

Table 2: Average iteration counts and execution rates, illustrating the effectiveness of the verification module.

Effectiveness of the framework. All experiments are initially conducted using GPT-4o. To assess the robustness and generalizability of our framework, we replace GPT-4o with the open-source language model DeepSeek. We randomly sampled three papers for this comparison, and the results are presented in Appendix D. Despite with the open-sourced language model, the framework can still produce high-quality diagrams: the generated outputs effectively capture the core concepts of the papers and maintain logical coherence in the arrangement of components.

Ablation Study. Our framework uses three key modules to produce the final diagram layout: the text analysis module, the refinement module, and the code verification module. To evaluate the contribution of the verification module, we conduct an ablation experiment by generating diagrams without the verification module. As shown in Table 2, without verification, the code executable rate drops to 0.883. When the verification module is enabled, with an average of 0.118 correction iterations, the framework achieves a 100% code execution success rate. We also assess the impact of the refinement module. On average, each diagram generation involves 2.12 iterations of refinement. Each iteration introduces improvements such as adding missing components, adjusting layouts, or enhanc-

ing visual clarity through colors and highlights. Figure 10 shows an example. While the first iteration already produces a solid initial layout, the second refinement iteration groups multiple outputs into a single cluster, enhancing both clarity and readability.

Case Study. To further illustrate the strengths and limitations of SCISKETCH in both workflows, we conduct a detailed analysis of both successful and unsuccessful cases, as shown in Figure 11 and Figure 12.

- **Graphic-code-based Workflow:** The top example corresponds to the VQ-VAE paper and demonstrates a high-quality generation. The left side shows the original diagram from the paper. As seen in the generated version, the system successfully captures the core components—such as the encoder, decoder, and embedding table—and integrates both input/output data and symbolic icons, resulting in a visually appealing and informative diagram. Despite these strengths, the SCISKETCH framework still has some limitations. In the bottom example which corresponds to the CoCA paper, although the diagram correctly represents the two generative paths and includes an amplification process, the complexity of the logic makes it challenging for the system to produce a clear and structured layout.
- **Image-based Workflow:** The top example corresponds to the FABLES paper and demonstrates a high-quality generation. In addition to the same strengths of the graphic-code-based workflow, the image-based-workflow can generate a more coherent diagram with elements are placed harmoniously and the icons are consistent due to the direct generation property. However, this approach also has certain limitations. In the bottom example which corresponds to the TEMPLE-MQA paper, although the diagram successfully conveys the core idea, some words are misspelled in the image. Furthermore, due to the generative nature, occasional hallucinated elements are introduced.

These issues highlight open challenges and potential directions for improving the framework in future work.

4 Related Work

Recent advances in text-to-image generation have focused predominantly on producing photorealistic images (Ramesh et al., 2021; Saharia et al., 2022; Zhang et al., 2023; Dai et al., 2023; Chang et al., 2023). While these models achieve impressive visual fidelity, they offer limited utility for generating structured, symbolic representations such as scientific diagrams. In contrast, emerging work on vector graphic generation in SVG format (Frans et al., 2022; Jain et al., 2023; Wu et al., 2023, 2024) shows potential for creating simple illustrations. However, these approaches fall short in handling the complex reasoning and spatial planning required for scientific schematics.

More directly relevant efforts include AutomaTikZ (Belouadi et al., 2024), which fine-tunes foundation models to generate TikZ code for vector graphics, and DiagrammerGPT (Zala et al., 2024), which adopts a two-stage pipeline involving layout planning followed by diagram synthesis using diffusion models. Similarly, DiagramAgent (Wei et al., 2025) coordinates multiple specialized agents to plan, code, and edit diagrams, showcasing the promise of large language models in structured content generation. Despite these innovations, such systems are typically limited to brief prompts and struggle with the dense, interrelated content characteristic of full-length scientific papers. Moreover, they offer limited support for hybrid diagrams that integrate empirical data visualizations with symbolic or conceptual elements—core features of scientific schematics.

To our knowledge, SCISKETCH is the first open-source framework designed specifically for schematic diagram generation for scientific research, making it accessible and extensible for the broader research community.

5 Conclusion

This work tackles the challenge of generating schematic diagrams from full-length scientific papers. We introduce the first open-source framework specifically designed for this task, enabling the creation of scientific diagrams directly from full-text content. Our system outperforms SOTA models in both quantitative metrics and human evaluations. By automating diagram generation, our approach has the potential to significantly boost researchers' productivity across academia and industry, streamlining the creation of high-quality visualizations. Future work could focus on semantic abstraction of generated SVG icons and enhancing layout robustness for complex diagram structures.

Acknowledgments

This project is supported by Tata Sons Private Limited, Tata Consultancy Services Limited, and Titan.

References

Jonas Belouadi, Anne Lauscher, and Steffen Eger. 2024. AutomaTikZ: Text-guided synthesis of scientific vector graphics with TikZ. In The Twelfth International Conference on Learning Representations.

Huiwen Chang, Han Zhang, Jarred Barber, Aaron Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Patrick Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. 2023. Muse: Text-to-image generation via masked generative transformers. In Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pages 4055–4075. PMLR.

Jaemin Cho, Abhay Zala, and Mohit Bansal. 2023. Visual programming for step-by-step text-to-image generation and evaluation. In Advances in Neural Information Processing Systems, volume 36, pages 6048–6069. Curran Associates, Inc.

Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, Matthew Yu, Abhishek Kadian, Filip Radenovic, Dhruv Mahajan, Kunpeng Li, Yue Zhao, Vladan Petrovic, Mitesh Kumar Singh, Simran Motwani, Yi Wen, Yiwen Song, Roshan Sumbaly, Vignesh Ramanathan, Zijian He, Peter Vajda, and Devi Parikh. 2023. Emu: Enhancing image generation models using photogenic needles in a haystack. Preprint, arXiv:2309.15807.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2023. Layoutgpt: Compositional visual planning and generation with large language models. Advances in Neural Information Processing Systems, 36:18225–18250.

Kevin Frans, Lisa Soros, and Olaf Witkowski. 2022. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. Advances in Neural Information Processing Systems, 35:5207–5218.

Ajay Jain, Amber Xie, and Pieter Abbeel. 2023. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1911–1920.

Jill H Larkin and Herbert A Simon. 1987. Why a diagram is (sometimes) worth ten thousand words. Cognitive science, 11(1):65–100.

Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. 2023. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. arXiv preprint arXiv:2305.13655.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In Advances in Neural Information Processing Systems, volume 36, pages 46534–46594. Curran Associates, Inc.

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, volume 139 of Proceedings of Machine Learning Research, pages 8821–8831. PMLR.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. 2022. Photorealistic text-to-image diffusion models with deep language understanding. In Advances in Neural Information Processing Systems, volume 35, pages 36479–36494. Curran Associates, Inc.

Jingxuan Wei, Cheng Tan, Qi Chen, Gaowei Wu, Siyuan Li, Zhangyang Gao, Linzhuang Sun, Bihui Yu, and Ruifeng Guo. 2025. From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Ronghuan Wu, Wanchao Su, and Jing Liao. 2024. Chat2svg: Vector graphics generation with large language models and image diffusion models. Preprint, arXiv:2411.16602.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. 2023. Iconshop: Text-guided vector icon synthesis with autoregressive transformers. ACM Transactions on Graphics (TOG), 42(6):1–14.

Abhay Zala, Han Lin, Jaemin Cho, and Mohit Bansal. 2024. Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. In COLM.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In Proceedings of the IEEE/CVF international conference on computer vision, pages 3836–3847.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang,

Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In Advances in Neural Information Processing Systems, volume 36, pages 46595–46623. Curran Associates, Inc.

A Select Examples

Example diagrams generation results from state-of-the-art foundation models (GPT-4o, Gemini-2.5-Pro, Claude-3.7-Sonnet) and SciSketch are shown in Figure 4, Figure 5, Figure 6, Figure 7, Figure 8, Figure 9. Compared to the baselines, our method produces diagrams that better align with the core concepts and logical structure of the original paper, while also exhibiting higher aesthetic quality. Moreover, our framework captures essential components without including verbose or irrelevant elements, resulting in concise and clear diagrams that effectively convey the intended information.

B Prompts

B.1 Layout Planning Phase in Graphic-Code-Based Workflow

Text Analysis

```
You are an expert in describing a scientific diagram
given the caption and the content of the paper.
You should give a detailed description of the
diagram which will be used to generate the diagram.
Output the description within the <description>
</description> tag.
You should follow the following steps:
1. Read the paper comprehensively and extract all
the information that relates to the caption.
2. Based on the caption and the paper, decide the
type of the diagram.
3. Based on the caption, paper content and the type
of the diagram, identify the components and layout
of the diagram.
4. Generate a detailed description of the diagram.
The description should comepletely come from the
paper. And it should match the caption.
5. Check the description with the caption and the
paper, make sure the description is correct and
complete. If not, revise the description until it
is correct and complete.
6. Wrap the description within the <description>
</description> tag.
Paper Content: {paper_content}
Caption: {caption}
```

Layout Generation

```
You are an expert in generating a scientific diagram
based on the description of the diagram. You should
generate an executable code of {language} to fully
represent the description. After generation, you
need to verify to make sure it could be executed
without any error.
You could follow the following steps:
1. Analyze the description and identify the type,
components and layout of the diagram.
2. Generate the executable {language} code for all
the components and layout.
3. Make sure the code covers all the components of
the description.
4. Verify the code to make sure it could be
executed without any error.
5. Return your result in a code block wrapped with
“‘.
Think step by step and make sure the diagram is
clear and matches the description and the caption.
Description: {description}
Code:
```

Layout Refinement

```
You are an expert in planning and designing a
scientific diagram with the {language}. Given
the description, the caption, and the code of
the diagram, you need to check if there is any
improvement to make the diagram more precise and
appealing to the reader. You should follow the
following steps:
1. Analyze the description and the caption of the
diagram to understand the components and layout of
the diagram.
2. Analyze the code and consider how the scientific
diagram can be enhanced to make it match the
description and caption.
3. Consider how to arrange the components and
layout of the diagram to make it more appealing to
the reader.
4. Try to avoid overlap of components as much as
possible.
5. If the code is good enough, you should not make
any changes to the code.
You will first need to decide if any improvement is
needed. You should wrap your decision inside the
<decision></decision> tag.
You should only input yes or no inside the
<decision></decision> tag.
If the decision is yes, you should then generate the
improved code in a code block wrapped with “‘.
Think step by step and make sure the code is
self-contained and executable.
Description: {description}
Caption: {caption}
Diagram: {diagram}
```

Graphic Code Verification

```
You are an expert of {language}. You know the rules
and regulations of {language}. You will be given
the {language} code of the diagram. You should
verify the code to make sure it could be executed
without any error. If you identify any error, you
should fix the error and output the fixed code.
You will first decide if there is any error in the
code. You should wrap your decision inside the
<decision></decision> tag. You should only input
yes or no inside the <decision></decision> tag.
If there are errors in the code, you should then
generate the fixed code in a code block wrapped with
“‘.
Think step by step and make sure the code is
self-contained and executable.
Here are some rules to follow:
1. The child-parent relationship should be correct
-- no child should be the parent of itself or the
parent of its parent.
2. For each mxGeometry object, there should be an
as="geometry" attribute at the end.
3. Each mxCell should have a parent.
Diagram: {diagram}
```

B.2 Diagram Generation Phase in Graphic-Code-Based Workflow

Element Discriminator

```
You are a scientific researcher with professional
design skills. Given a drawio diagram which shows a
```

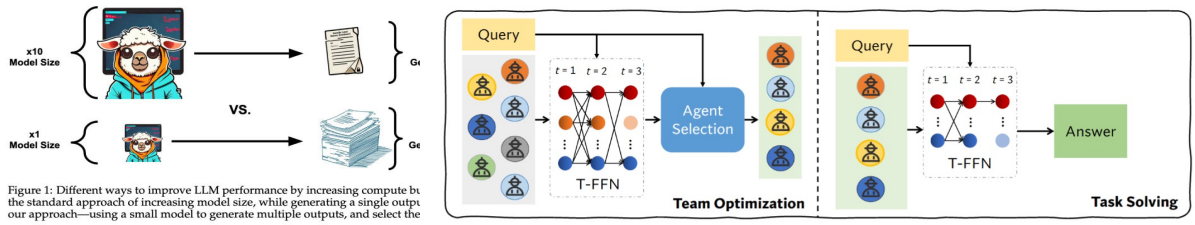


Figure 4: Example diagram from the original paper. The left is the Budget Relocation paper, the right is the DyLAN paper

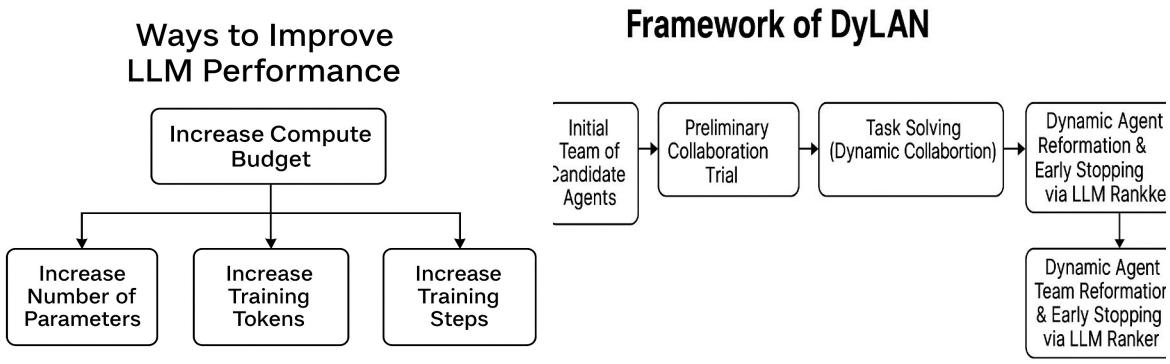


Figure 5: Example diagram generated by GPT-4o

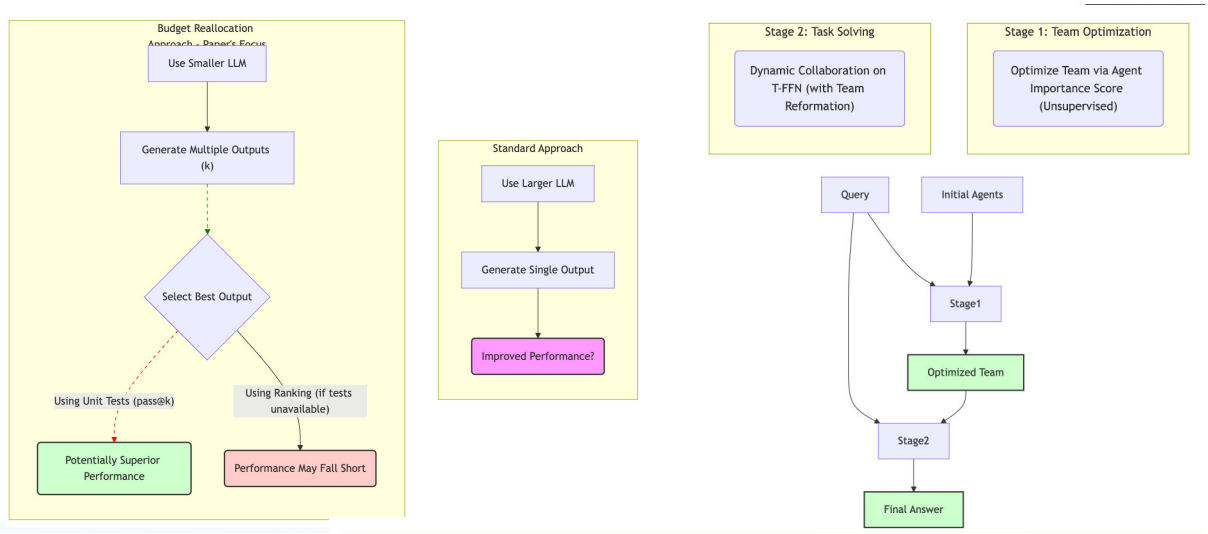


Figure 6: Example diagram generated by Gemini-2.5-Pro

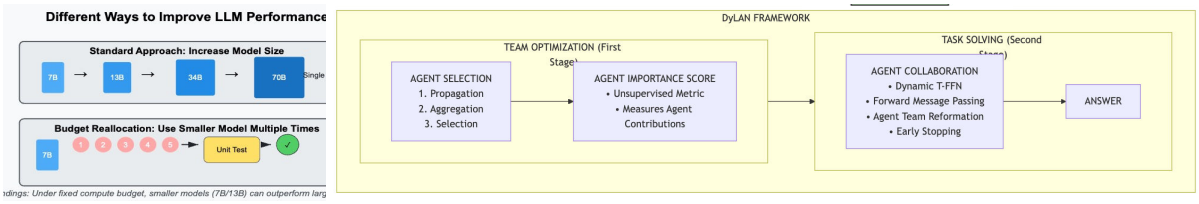


Figure 7: Example diagram generated by Claude-3.7-Sonnet

diagram of a scientific paper, your job is to select some of the components to be replaced by images to make it more appealing. The images should come from either the author's input, or could be replaced by a vivid svg icon.
You should follow the following steps:
1. Comprehend the drawio diagram thoroughly.
2. Identify which component should be replaced with

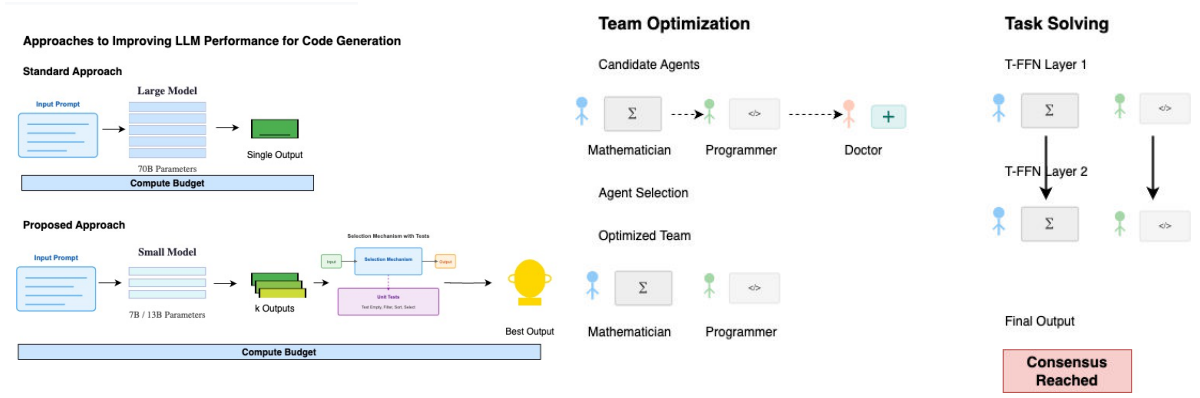


Figure 8: Example diagram generated by SCISKETCH graphic-code-based workflow

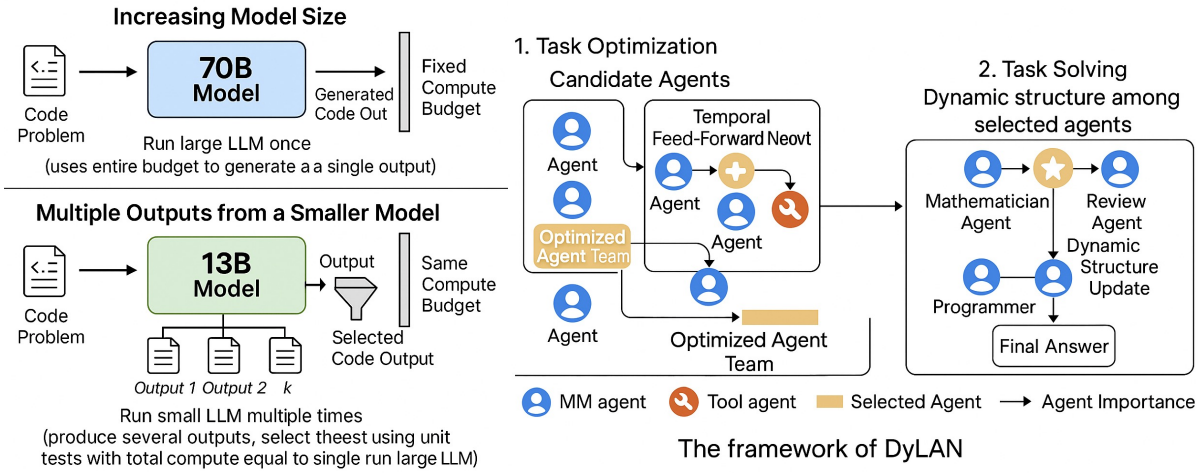


Figure 9: Example diagram generated by SCISKETCH image-based workflow

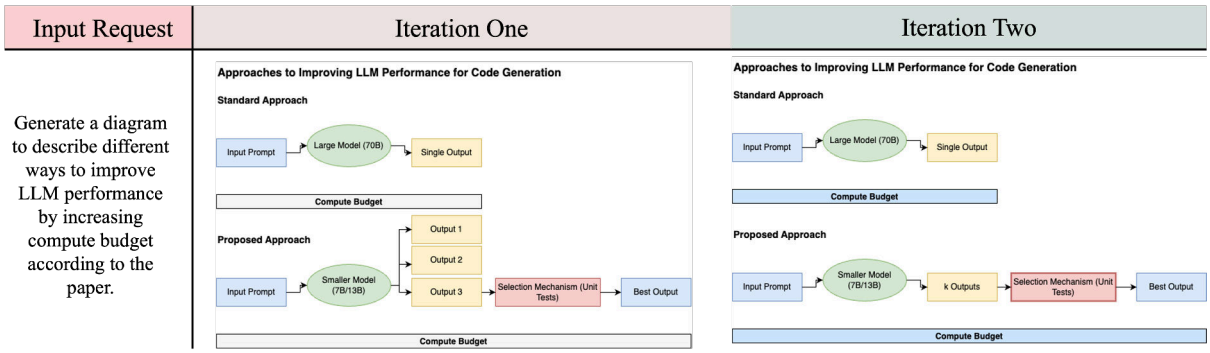


Figure 10: An example illustrating how the iterative refinement process improves diagram quality. In the second iteration, the color of the "Compute Budget" bars is adjusted for better visibility and thematic alignment, and multiple outputs are grouped into a single cluster to enhance clarity.

```
an image to make the diagram look better.
3. If the component is a concrete example or data
from the paper, then it should be provided by the
author.
4. If the component is an abstract component which
could be replaced by a svg icon, then it should be
provided by a svg generator.
5. You need to output the components result which
could be replaced by an image in a valid json string
as a list of dictionaries.
The dictionary should strictly follow the rules:
1. source: the replaced image source either be
"author" or "svg" according to the rules.
2. id: the id of the mxCell which should be
replaced.
3. title: the value of the mxCell.
4. description: a detailed description of the
component which will help generate the image later
```

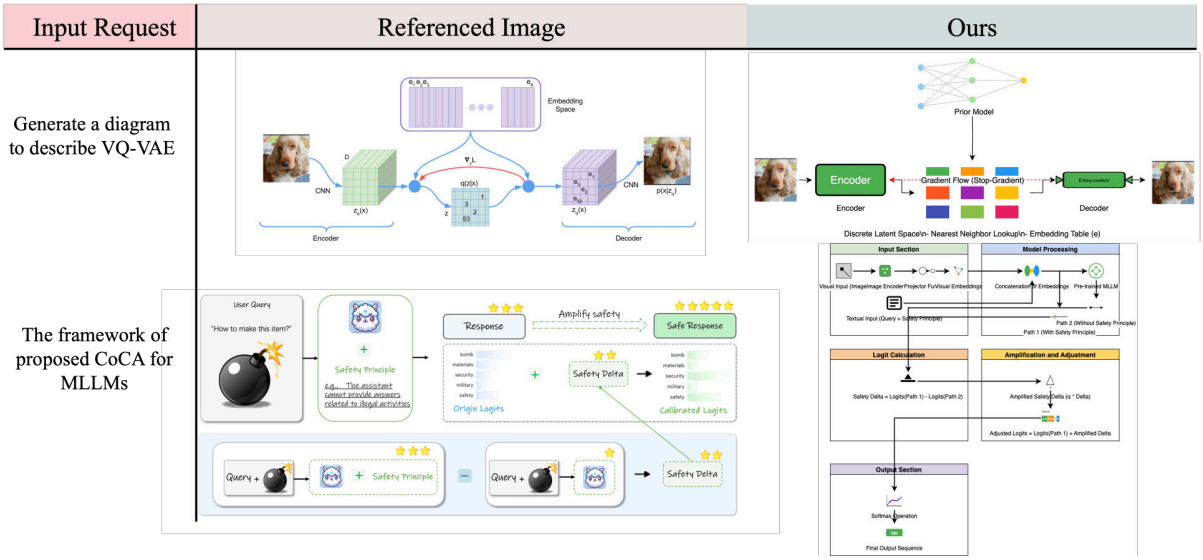


Figure 11: Examples of diagrams generated by SciSketch framework graphic-code-based workflow. The top example illustrates a high-quality generation with accurate structure and visual clarity. The bottom example demonstrates a case with structural or semantic errors.

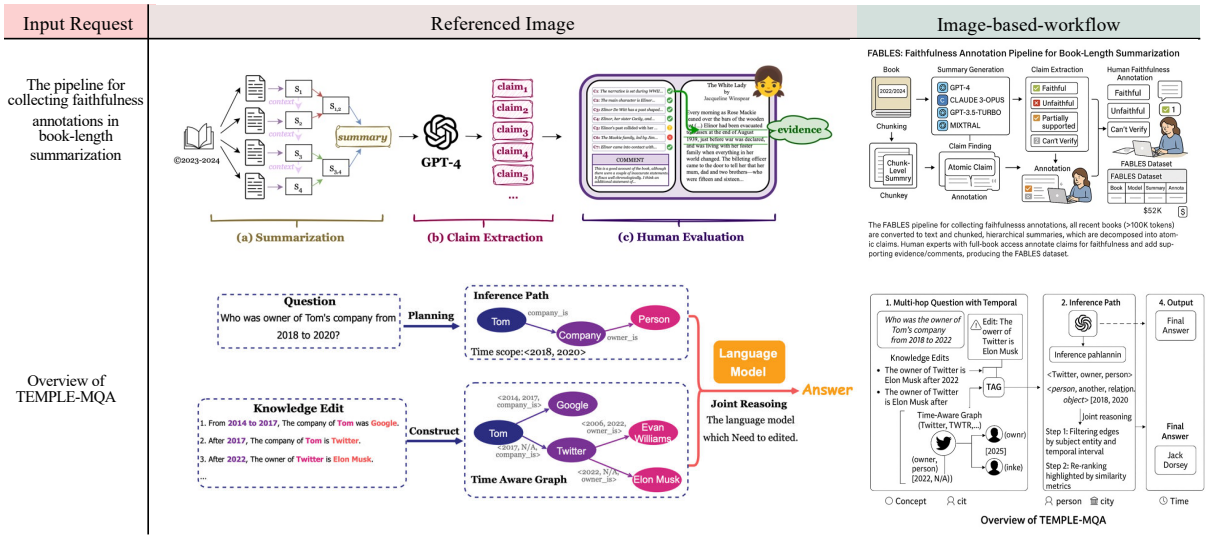


Figure 12: Examples of diagrams generated by SciSketch framework image-based workflow. The top example illustrates a high-quality generation with accurate structure and visual clarity. The bottom example demonstrates a case with typos and hallucinations.

```
on.
Think step by step and put the final result in a
valid json string wrapped by triple "`.
Diagram: {diagram}
```

SVG Generation

```
You are an excellent svg designer. Your job is to
generate an svg given the title and description of
the svg. Make it as simple as possible, otherwise
it will mess up with other components in a diagram.
Output the svg and wrap it within triple "`.
title: {title}
description: {description}
```

Empirical Figure Retrieval

```
You are an expert in matching for an image in a
given candidate_list with image names based on the
title and description. You will need to give the
index if the image is found, otherwise output -1.
The index is zero based. Only output the number of
the index.
title: {title}
description: {description}
candidate_list: {candidate_list}
index:
```

B.3 Diagram Generation in Image-Based Workflow

Diagram Image Refinement

```
You are an expert in assessing the quality of an
image given the description. You need to find out
if there are any issues in the image, and if so, how
```

```
to fix them.  Output the decision and wrap it inside
the <decision></decision> tag.  You should only
output yes or no inside the <decision></decision>
tag.  If nothing needs to be fixed, you should
output no.
Output your assessment and suggestions inside the
<suggestion></suggestion> tag.
Description:  {description}
```

C Evaluation

C.1 Human Evaluation Criteria

We ask the evaluator to give a score from 1 to 5 with regarding to the three fine-grained aspects, Completeness, Logical Consistency and Aesthetic Quality. The definition and instruction of scoring is shown in Table 3

D Experiments

D.1 Experiments with DeepSeek

To evaluate the adaptability of our framework to open-source foundation models, We replace all the foundation model in the framework with DeepSeek. And we generate the diagrams for three sampled papers. The result is shown in Figure 13

Aspect	Definition	Instructions
Completeness	Evaluate if the diagram includes all essential components and elements described in the source paper without omitting critical information.	Score 1 means the diagram is missing most of the main components or fails to capture essential parts of the source paper. Score 5 means the diagram includes all relevant components, fully representing the source material without any omissions.
Logical Consistency	Evaluate if the relationships and connections between diagram components accurately reflect the logical structure and intended semantics of the original paper.	Score 1 means the diagram has misleading, incorrect, or missing connections between components, failing to capture the logical flow. Score 5 means the diagram's relationships and connections are accurate and coherent, perfectly mirroring the logic and intended meaning of the source paper.
Aesthetic Quality	Evaluate the visual clarity and professional presentation of the diagram including layout organization, spacing, and element alignment.	Score 1 means the diagram is cluttered, visually confusing, or contains significant overlap or misalignment. Score 5 means the diagram is visually clear, well-organized, with good spacing, and all elements are neatly aligned.

Table 3: Evaluation Criteria Descriptions

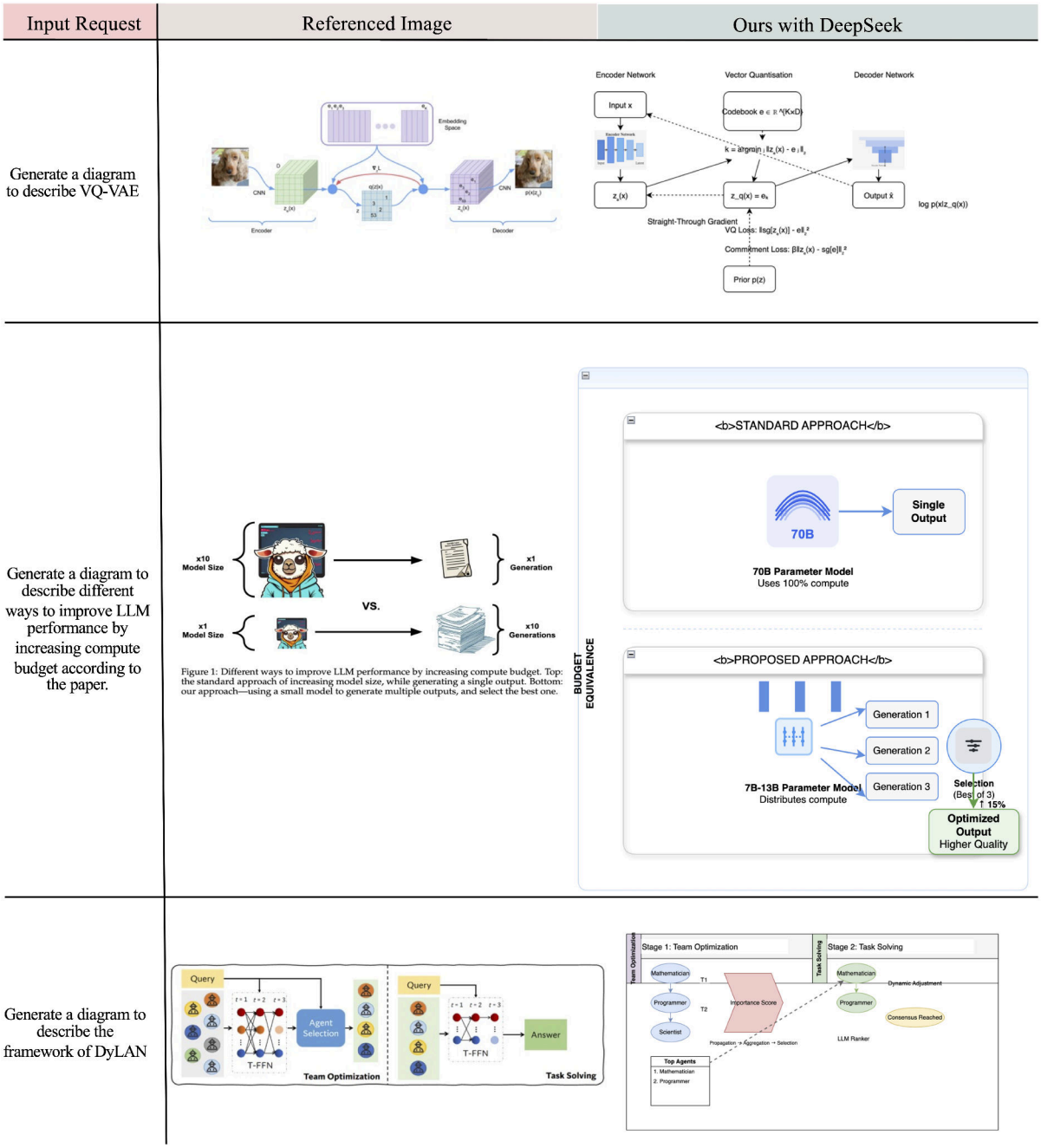


Figure 13: Examples of diagrams generated by the SCISKETCH framework using an open-source language model. The left example shows the ground-truth diagram from the original paper. The right example presents the diagram generated by SCISKETCH using DeepSeek. Despite the use of an open-source model, the generated diagram successfully captures the core concepts and presents the components in a logically coherent manner.

MALLM: Multi-Agent Large Language Models Framework

Jonas Becker[1,2,*], Lars Benedikt Kaesberg[1,*], Niklas Bauer[1], Jan Philip Wahle[1], Terry Ruas[1], Bela Gipp[1]
[1]University of Göttingen, Germany; [2]LKA NRW, Germany
[*]{jonas.becker, l.kaesberg}@uni-goettingen.de

Code github.com/Multi-Agent-LLMs/mallm
Demo mallm.gipplab.org
Package pypi.org/project/mallm

Abstract

Multi-agent debate (MAD) has demonstrated the ability to augment collective intelligence by scaling test-time compute and leveraging expertise. Current frameworks for MAD are often designed towards tool use, lack integrated evaluation, or provide limited configurability of agent personas, response generators, discussion paradigms, and decision protocols. We introduce MALLM (Multi-Agent Large Language Models), an open-source framework that enables systematic analysis of MAD components. MALLM offers more than 144 unique configurations of MAD, including (1) agent personas (e.g., `Expert`, `Personality`), (2) response generators (e.g., `Critical`, `Reasoning`), (3) discussion paradigms (e.g., `Memory`, `Relay`), and (4) decision protocols (e.g., `Voting`, `Consensus`). MALLM uses simple configuration files to define a debate. Furthermore, MALLM can load any textual Hugging Face dataset (e.g., MMLU-Pro, WinoGrande) and provides an evaluation pipeline for easy comparison of MAD configurations. MALLM enables researchers to systematically configure, run, and evaluate debates for their problems, facilitating the understanding of the components and their interplay.

1 Introduction

Multi-agent debate (MAD) has emerged as a new paradigm to solve complex tasks with multiple large language models (LLMs) (Chan et al., 2024; Du et al., 2023; Liang et al., 2024; Wang et al., 2024b). Yet, we have not understood the exact mechanisms of when and why MAD is successful. Different hypotheses exist around whether MAD is another way to scale test-time compute (Yang et al., 2025), or whether the combination of individual components has emergent capabilities (Liang et al., 2024). Understanding these mechanisms requires a systematic evaluation, specifically code that enables adjusting one variable of the MAD at a time to measure its effect.

Recent work (Guo et al., 2024; Tran et al., 2025; Tillmann, 2025) has identified several key aspects influencing multi-agent discussions. We focus on the following three main components: (1) **agents** define "who" is participating in the debate, meaning the personas of agents and their response style (Wang et al., 2023; Xu et al., 2023); (2) **discussion paradigms** determine "how" the debate is taking place, including agent response order and turn boundaries, structuring information flow (Yin et al., 2023); (3) **decision protocols** choose "what" the debate result will be, meaning deciding when discussions end and determining a final answer (Chen et al., 2023a; Kaesberg et al., 2025). As later evaluations will demonstrate, each of these components is crucial in downstream tasks. Adjusting them individually is particularly important for systematic investigations.

To satisfy the growing demand for MAD applications, many frameworks have been developed (Wu et al., 2023; Gao et al., 2024; Hong et al., 2023). Yet these frameworks intertwine the definitions of multiple components, such as agents and discussion paradigms (Wu et al., 2023) or do not allow for adjusting specific parts, such as discussion paradigms or decision protocols (Hong et al., 2023), hindering independent experimentation. Existing approaches often constrain experimentation by using predefined agents, discussion paradigms, or decision protocols (Zhuge et al., 2024; Gao et al., 2024). Some frameworks also specialize in particular use cases, such as tool usage (OpenAI, 2024), and typically lack integrated evaluation pipelines for analyzing individual MAD components systematically (Wu et al., 2023; Gao et al., 2024). To the best of our knowledge, no current framework exists to evaluate the three core components of related

[*]Equal contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 418–439
November 4-9, 2025

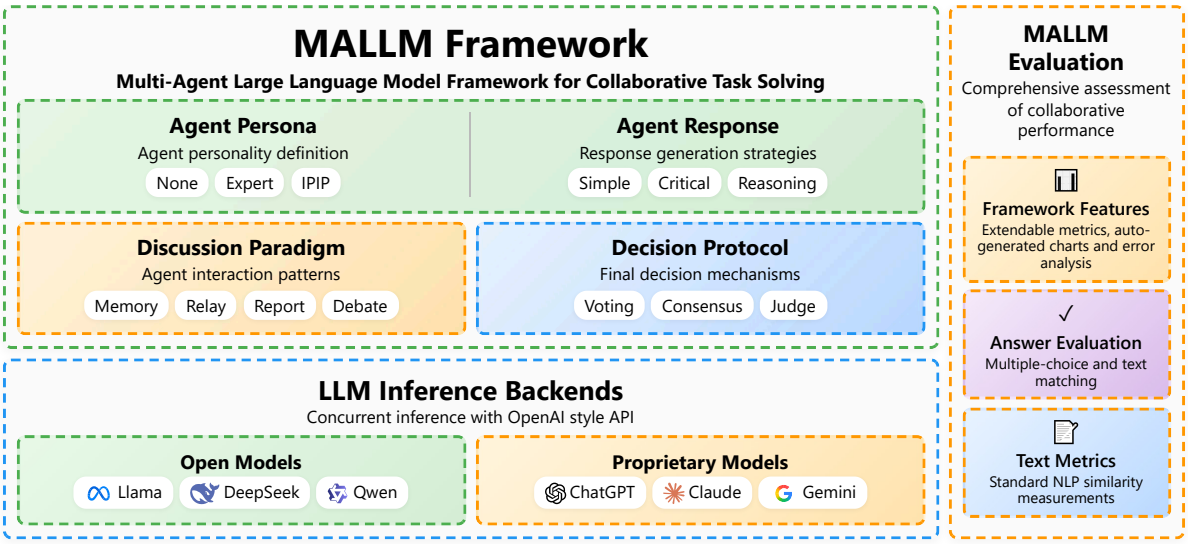


Figure 1: Overview of the MALLM framework and its components.

works and their interactions for MAD: agents, discussion paradigms, and decision protocols.

We propose the open-source Multi-Agent LLM (MALLM) framework to address these limitations (see Figure 1 for an overview). MALLM integrates several MAD components from previous research. Researchers can individually configure their debate setup through parameter settings and easily extend the framework by inheriting existing classes or using template functions. Without additional programming effort, MALLM supports more than 144 distinct MAD configurations. Thus, it enables researchers to reproduce prior MAD experiments, apply MAD methods to new datasets or tasks, and systematically ablate individual MAD components to analyze their impact. We present MALLM's capabilities on our demo website[1].

The key contributions are:

- We propose MALLM, an open-source framework that supports studies of MAD by enabling controlled variation of agents, discussion paradigms, and decision protocols.
- We allow researchers to experiment with 144 existing MAD configurations on a wide range of text-based tasks through automated dataset loading, preprocessing, and evaluation.
- We provide abstract classes and template functions to implement new MAD components and tasks, allowing others to understand the best configuration for their specific research goals.

[1] mallm.gipplab.org

2 Related Work

We identify three key components of related work that are commonly discussed in MAD. Wang et al. (2023); Xu et al. (2023) use **agents** with varying personas and response styles. Yin et al. (2023); Li et al. (2024) define **discussion paradigms** that determine turn order and information flow. Chen et al. (2023a); Yang et al. (2024) explain variations in **decision protocols** that aggregate agent outputs into a single solution.

Existing MAD frameworks vary across the core components described previously: agents, discussion paradigms, and decision protocols. AutoGen supports multi-turn interactions between customizable agents but does not separate discussion paradigms from agent definitions, hindering systematic studies of each component (Wu et al., 2023). It also does not support integrated evaluation pipelines, which need to be coded externally. Similarly, MetaGPT assigns tasks to specialized agents that follow standard operating procedures, tightly coupling agent roles and response styles, but restricting experimentation with alternative discussion paradigms or decision protocols (Hong et al., 2023).

Other frameworks offer alternative abstractions. GPTSwarm models agent interactions as optimizable computational graphs, focusing on information flow rather than the modular comparison of agents or decision protocols (Zhuge et al., 2024). AgentScope simplifies interactions using predefined pipelines (Gao et al., 2024), constraining discussion paradigms and

limiting evaluation of agent personas or decision protocols. The OpenAI Agents SDK coordinates tool-using agents (OpenAI, 2024), prioritizing agent functionality but lacking customizable decision protocols for MAD.

A common limitation across existing frameworks is the tight coupling among agents, discussion paradigms, and decision protocols, which hinders the analysis of each component independently or in combinations of choice. This makes the investigation of which specific components contribute and should be used for particular use cases difficult. Most frameworks provide fixed orchestration setups, restricting experimentation with alternative decision protocols or agent configurations (OpenAI, 2024; Zhuge et al., 2024), and few include integrated evaluation pipelines (Wu et al., 2023; Gao et al., 2024; Smit et al., 2023). No current framework explicitly supports the systematic analysis of individual MAD components and their interactions. Our proposed framework, MALLM, addresses these limitations with a modular architecture clearly separating agents, discussion paradigms, and decision protocols into interchangeable modules. This design enables a systematic study of each component independently and combined, supported by an integrated evaluation pipeline. A comparison of MALLM and other frameworks for MAD is included in Table 5 of Appendix C.

3 MALLM Framework

We propose MALLM, a framework for MAD. It coordinates agents to solve text-based tasks (Guo et al., 2024). MALLM receives an input task and outputs a solution after performing a MAD. Figure 1 illustrates the components of MALLM.

MALLM implements three **agent personas** (`None`, `Expert`, `IPIP`), three **agent response generators** (`Simple`, `Critical`, `Reasoning`), four **discussion paradigms** (`Memory`, `Relay`, `Report`, `Debate`), and three main **decision protocols** (`Voting`, `Consensus`, `Judge`). Each of the component variants can be parameterized individually, allowing systematic comparison of individual setups without additional code.

The agents participating in the debate can use most proprietary and open models, as MALLM supports any OpenAI-compatible API endpoint for inference. The integrated evaluation pipeline can be used to analyze the large amounts of data generated by MAD in a unified way and directly generate comparative charts to visualize performance across different configurations. We provide more details on the framework parameters in Appendix E and prompts in Appendix G.

With MALLM, users can explore the effects of changing components within MAD. The effects of each variation in MAD can be measured towards solving various text-based problems, such as mathematical reasoning (Cobbe et al., 2021), ethical question-answering (Hendrycks et al., 2021a), and more. Our public demo includes three persona generators, three response generators, four discussion paradigms, and four decision protocols[2]. Thus, users can explore 144 MAD configurations directly, observing the effect of parameter combinations. A screenshot of our interactive demo is in Figure 6 of Appendix B.

3.1 Agents

An **agent** is defined by its role (persona generator) and its answer style (response generator).

Persona Generator. The persona specifies agent behavior by their system prompt, e.g., expertise or personality (Xu et al., 2023; Wang et al., 2023). Personas are created iteratively to be complementary and unique. We include three persona types: `None`, `Expert`, and `IPIP`.

None: Disables the persona generation. It assigns each agent a generic name ("Participant N") for baseline experiments. **Expert**: Creates domain-specific personas aligned with the task description (Xu et al., 2023). Examples are an "Educator" for machine learning explanations, a "Software Developer" for app development, or a "Chef" for cooking tasks. **IPIP**: Based on the Big Five personality traits, using the open-source IPIP-NEO classification (Costa and McCrae, 1992; Goldberg, 1999). They cover Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness. The items originate from the International Personality Item Pool (IPIP), frequently used in Psychology (Maples et al., 2014). This enables the detailed modeling of psychological diversity (Serapio-García et al., 2023; Sorokovikova et al., 2024).

Response Generator. The response generator produces agent responses in a specified format or style, influencing how agents interact (e.g., neutral

[2] mallm.gipplab.org

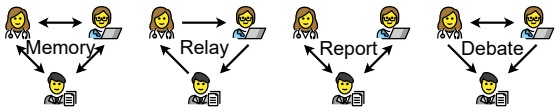


Figure 2: Overview of four discussion paradigms.

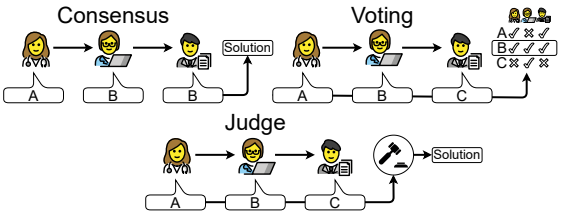


Figure 3: Overview of three main decision protocols.

or critical) (Mizrahi et al., 2024). They vary in how the agents are prompted to generate a response in the debate. MALLM includes the `Simple`, `Reasoning`, and `Critical` response generators. We include their prompts in Appendix G.3.

Simple: Produces free-text responses in a neutral tone, explicitly indicating agreement or disagreement. **Reasoning**: Responds step-by-step, including analysis, alternatives, and conclusions. Agents share their reasoning but not solutions, encouraging independent idea generation. **Critical**: Tasks the agent to identify weaknesses, question assumptions, and suggest alternative approaches.

3.2 Discussion Paradigm

The **discussion paradigm** defines the structure of agent interaction. It specifies turn-taking and information access rules for the MAD. We implement four paradigms (Yin et al., 2023): `Memory`, `Report`, `Relay`, and `Debate`. Each paradigm differs in information flow and visibility, as illustrated in Figure 2.

Memory: All agents have full visibility into each other's messages across turns. **Relay:** Information is passed sequentially between agents in a chain, with only the last message visible to the next. **Report:** Agents independently solve the tasks and report back to a central agent. **Debate:** Agents argue in pairs, taking turns to debate intermediate conclusions before a central agent is consulted.

3.3 Decision Protocol

Each agent in a multi-agent system generates solution drafts. **Decision protocols** systematically determine when discussions end and combine agent-generated solutions into a final decision.

The MALLM framework implements three decision protocol families: `Consensus`, `Voting`, and `Judge`, as illustrated in Figure 3. Debates can run with a fixed number of turns or perform early stopping upon a successful decision.

Consensus: Consensus decision protocols decide on an answer by having the agents converge on one solution. The solution is selected when a required level of agreement among agents is reached (Yin et al., 2023). MALLM includes three agreement levels: `Majority Consensus` (over 50%), `Supermajority Consensus` (over 66%), and `Unanimity Consensus` (100%). **Voting**: Uses a fixed number of discussion rounds (default three, following findings by Du et al. (2023)) before agents vote. In the event of a tie, we run an additional round of debate and voting. Variants are inspired by Yang et al. (2024) and include `Simple Voting` (i.e., each agent votes for their preferred solution), `Approval Voting` (i.e., multiple acceptable solutions per agent), `Ranked Voting` (i.e., agents rank solutions, best cumulative rank selected), and `Cumulative Voting` (i.e., agents allocate up to 25 points across solutions, highest total points selected). **Judge**: Relies on one agent reviewing all solutions, choosing either a preferred one or synthesizing a new solution. The effectiveness of the Judge protocol depends on the model's reasoning capabilities (Zheng et al., 2023).

3.4 Evaluation

The MALLM framework includes a pipeline for evaluating MAD configurations, producing statistics and charts for a dataset and its metrics.

Datasets. The pipeline provides integrated loaders for reasoning tasks (e.g., WinoGrande (Sakaguchi et al., 2020), StrategyQA (Geva et al., 2021)) and knowledge tasks (e.g., GPQA (Rein et al., 2023), MMLU-Pro (Wang et al., 2024c)) as well as core tasks of text generation (Becker et al., 2024), such as paraphrasing (Kovatchev et al., 2018) or summarization (Narayan et al., 2018). It further supports any textual Hugging Face dataset for problem-solving. Researchers can also add their own dataset via subclassing. All datasets are converted into a unified format for processing.

Metrics. We include question-answering and free-text evaluations. For question-answering, we compute accuracy by comparing selected response letters against reference solutions via regex. For free-text tasks, we include BERTScore (Zhang et al., 2020) and textual overlap measures (BLEU (Papineni et al., 2002), ROUGE-1/2/3/L (Lin,

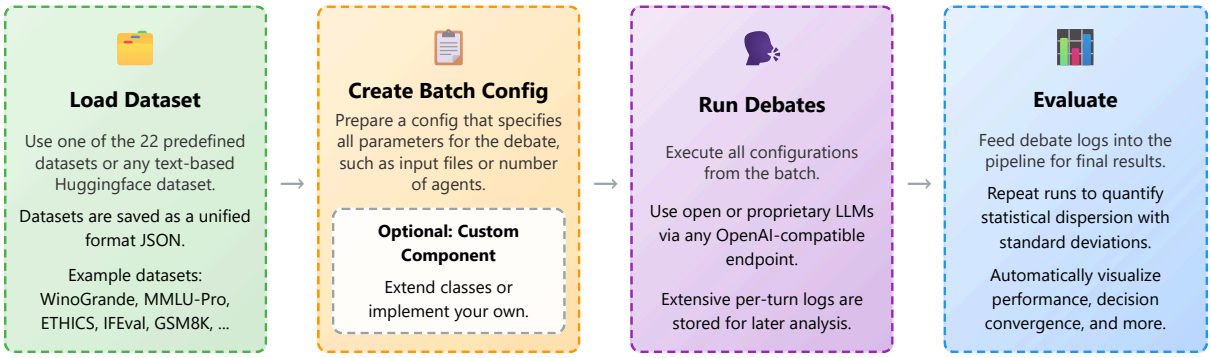


Figure 4: Example workflow of experimenting with MAD using MALLM. First, we can load a dataset. Second, a config file defines the MAD. Third, the debates run and produce output logs. Last, the debates are evaluated. While MALLM already comes with many parameters and components to test, researchers can optionally incorporate their own components, which are tailored to their specific experiment.

2004), and METEOR (Banerjee and Lavie, 2005)).

Statistical variance. We find that several studies on MAD do not account for statistical variance (Wu et al., 2023; Talebirad and Nadiri, 2023), yet it can have marked impacts on MAD (Smit et al., 2023). MALLM enables repeated experiments and calculates standard deviations between them, thereby quantifying statistical dispersion.

Automatic charts. MALLM can visualize the evaluation results of MAD configurations. For this, researchers can pass a single file or a directory with multiple evaluation results directly to an automatic chart generator. Examples of the generated charts by the evaluation pipeline can be seen in Figures 7 to 10 of Appendix F.

4 Application

We explain the workflow for using MALLM to evaluate a specific MAD configuration. First, we can load any text-based Hugging Face dataset for problem-solving via our dataset loader (e.g., MMLU-Pro (Wang et al., 2024c)). Second, MALLM comes with a configuration file (cf. Appendix H) that the user can change to adjust parameters and the components for the desired experiment (e.g., using `Llama-3.3-70B-Instruct` as a model and changing the configuration to `four` debating agents, `Relay` discussion paradigm, and `Unanimity Consensus` decision protocol). Third, we can run the experiment specified by the configuration file. MALLM keeps extensive logs of all debates. They include each agent's messages, votes cast, used components, and configuration parameters. Lastly, we can directly feed the logs into our evaluation pipeline, computing metrics (e.g., accuracy) and leveraging automatic chart generation (cf. Figures 7 to 10 in Appendix F).

Each component (agent, discussion, decision) comes with abstract base classes to describe the pipeline of MAD. If necessary, a custom component could be provided by inheriting features from a component's abstract base class. For example, Sketch-of-Thought (Aytes et al., 2025) is a variant of Chain-of-Thought (Wei et al., 2023) that restructures how models express intermediate steps of their reasoning. To integrate with MALLM, we would create a custom class `SoTResponseGenerator` inheriting from the abstract base class `ResponseGenerator`. Then, the components of MAD can be defined by a configuration file as usual. Figure 4 provides an overview of the workflow for conducting and evaluating MAD with MALLM.

4.1 Use Cases

The following examples illustrate possible research directions that can be realized using MALLM.

Agents on paradigms. The number of agents for the MAD is modifiable. Researchers can test the impact of this on the various discussion paradigms. For example, a study could compare two, three, four, or five agents on the memory and relay paradigm. As these paradigms differ in their information flow, it would be interesting to see how this impacts task performance.

Testing new task. JailbreakBench (Chao et al., 2024) measures the safety of LLMs against jailbreaks. One avenue could be the comparison of multi-agent safety with a single-agent setup. For this, we can use MALLM, which is plug-and-play with a template configuration set.

Simple	Critical	Reasoning
58.6±1.6	**61.4**±3.3	52.2±2.8

Table 1: Comparison of accuracy averaged over all voting-based decision protocols using the simple, critical, and reasoning response generators on the StrategyQA dataset. Best is bolded. ± shows standard deviation over three runs.

CoT	Memory	Relay	Report	Debate
56.9±1.8	60.8±2.6	**62.9**±1.6	60.9±3.1	61.9±1.1

Table 2: Accuracy of MAD on StrategyQA with different discussion paradigms. Best and worst are bolded. ± shows standard deviation over five runs.

Finetuned agents. MALLM works with any proprietary and open-source LLM, which means that we can also provide our own model for the agents. A promising direction is to finetune an agent to enhance its argumentation skills, thereby improving task performance on reasoning tasks such as StrategyQA (Geva et al., 2021).

Moderated paradigm. A dynamic moderator can be implemented by subclassing the abstract base class `DiscussionParadigm`. We can define logic for an LLM-based moderator agent to adjust speaking order based on previous agent contributions. This enables an investigation into the effects of adaptive moderation.

4.2 Example Experiments with MALLM

MALLM can be applied to various use cases. To demonstrate the opportunities for experimental setups, we provide some example investigations. Supplementary information, such as used models and parameters, can be found in Appendix E.

Agents. Kaesberg et al. (2025) use MALLM to experiment with response generators on the StrategyQA dataset (Geva et al., 2021). Table 1 presents the average accuracy across all voting-based decision protocols, using the `Memory` discussion paradigm and `Expert` personas with the `Simple`, `Critical`, and `Reasoning` response generators. The `Critical` response generator slightly improves performance by encouraging agents to critically evaluate responses from others, resulting in a 2.8% point increase. The `Reasoning` response generator decreases performance by 6.4% points, likely because it imposes a strict response structure. Strictly structured responses can degrade the task performance of MAD, a characteristic that was previously noted in single-agent setups (Tam et al., 2024). To summarize, invoking strict response patterns from agents can harm task performance, while prompting agents to think critically can boost it.

Dataset	Voting	Consensus
Knowledge-Based		
MMLU	51.7 ± 2.4	**54.0** ± 2.7
MMLU-Pro	31.1 ± 3.5	**36.0** ± 1.8
GPQA	29.7 ± 2.5	**31.0** ± 2.4
Reasoning-Based		
SQuAD 2.0	**56.7** ± 1.6	43.6 ± 1.5
StrategyQA	**58.6** ± 2.0	58.4 ± 1.6
MuSR	**54.8** ± 1.9	28.4 ± 2.6

Table 3: Mean performance for voting and consensus decision protocols on knowledge and reasoning tasks. Best is bolded. ± shows standard deviation over three runs.

Discussion Paradigms. Becker (2024) compares discussion paradigms on the StrategyQA dataset (Geva et al., 2021), using `Expert` personas, the `Simple` response generator, and `Majority Consensus`. MAD runs until the agent agrees to a solution or until a maximum of seven turns is reached. We find that, using `Majority Consensus`, most debates reach an agreement and end within the first three turns. Thus, seven turns provide a reasonable headroom for this experiment. The results reveal two findings. First, Table 2 compares the discussion paradigms of MAD against a single LLM baseline with Chain-of-Thought (Wei et al., 2023). All paradigms outperform a single LLM with Chain-of-Thought prompting on StrategyQA, improving accuracy by up to 4.0%. Second, we investigate the impact of information transparency on convergence speed for MAD. We find that the very transparent paradigm `Memory` enables faster consensus (avg. 1.75 turns), while limited visibility between agents in `Relay` slows it down (avg. 2.61 turns). Thus, information transparency can lead to quicker convergence in MAD without sacrificing task performance. In summary, MAD can outperform Chain-of-Thought on reasoning tasks, such as StrategyQA, while the information transparency of the discussion paradigm impacts the convergence speed of MAD.

Decision Protocols. Investigations by Kaesberg et al. (2025) compare the average of all `Voting` and `Consensus` decision protocols across knowledge and reasoning tasks, summarized in Table 3. They use the `Simple` response generator and

the `Memory` discussion paradigm. `Consensus` consistently outperforms `Voting` on knowledge tasks (MMLU (Hendrycks et al., 2021b), MMLU-Pro (Wang et al., 2024c), GPQA (Rein et al., 2023)), achieving approximately 2.8% higher accuracy due to repeated verification steps. `Voting` protocols significantly improve accuracy by about 13.2% on reasoning-intensive tasks (SQuAD 2.0 (Rajpurkar et al., 2018), StrategyQA (Geva et al., 2021), MuSR (Sprague et al., 2024)), benefiting from diverse reasoning paths. To summarize, the selection of the decision protocol depends on the specific task. When chosen correctly, it can notably improve task performance.

Creating Demo Examples. For the demonstration of MALLM, we create a set of 144 different example configurations using MALLM's batch feature, called DEBATE (Diverse Exchanges Between Autonomous Talking Entities). We release the DEBATE dataset publicly[3]. Potential uses for this data could be to (1) study how agents debate as a proxy to humans; (2) study the structural reasons why MAD can fail in some scenarios, as identified by Becker et al. (2025); (3) explore how prompting agents to assess prior messages critically affects the speed of consensus-building. More details on the DEBATE dataset are in Appendix D.

5 Epilogue

We proposed MALLM, a framework specialized in conversational problem-solving for MAD. MALLM enables users to configure debates for their specific problems and research objectives. More specifically, our framework supports the analysis of multiple components, including agent personas, response generators, discussion paradigms, and decision protocols.

MALLM works with most proprietary and open models and can load any text-based Hugging Face dataset for problem-solving. MALLM's evaluation pipeline offers pre-implemented metrics (e.g., Accuracy, BLEU, BERTScore) and automatic chart generation, while accounting for the statistical variance of MAD. A demo for the capabilities of MALLM is publicly available[4].

We described four potential use cases that can be further developed: the number of agents and their impact, the resilience of MAD against jailbreak attacks, finetuning specialized agents, and adaptive moderation. Future work could also expand MALLM with additional functionalities, such as evaluating debates through any Hugging Face metric. Beyond serving as a testbed for MAD itself, MALLM provides researchers with an environment to assess how variations in agents, discussion paradigms, and decision protocols can impact their specific problems.

Limitations

We provide the MALLM framework with pre-implemented variants for personas, response generators, discussion paradigms, and decision protocols. While our goal was to include diverse variants backed by literature (e.g., voting (Yang et al., 2024), consensus (Yin et al., 2023), and judge (Zheng et al., 2023) for decision protocols), we could not account for all possible patterns of agent orchestration. There is the possibility that niche use cases with MAD and future developments are not captured by our selection, e.g., decisions by confidence-weighted voting (Chen et al., 2023b). We publish our framework as open-source, allowing and encouraging researchers to develop custom components via subclassing and apply MALLM to their specific use cases.

6 Acknowledgements

This work was supported by the Landeskriminalamt NRW, the Lower Saxony Ministry of Science and Culture and the VW Foundation.

[3] huggingface.co/datasets/Multi-Agent-LLMs/DEBATE
[4] mallm.gipplab.org

References

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *ArXiv preprint*, abs/2503.05179.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Jonas Becker. 2024. Multi-Agent Large Language Models for Conversational Task-Solving. *ArXiv preprint*, abs/2410.22932.

Jonas Becker, Lars Benedikt Kaesberg, Andreas Stephan, Jan Philip Wahle, Terry Ruas, and Bela

Gipp. 2025. Stay focused: Problem drift in multi-agent debate. *ArXiv preprint*, abs/2502.19559.

Jonas Becker, Jan Philip Wahle, Bela Gipp, and Terry Ruas. 2024. Text generation: A systematic literature review of tasks, evaluation, and challenges. *ArXiv preprint*, abs/2405.15604.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better llm-based evaluators through multi-agent debate. In *International Conference on Representation Learning*, volume 2024, pages 9079–9093.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 55005–55029. Curran Associates, Inc.

Huaben Chen, Wenkang Ji, Lufeng Xu, and Shiyu Zhao. 2023a. Multi-agent consensus seeking via large language models. *ArXiv preprint*, abs/2310.20151.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. 2023b. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *ArXiv preprint*, abs/2309.13007.

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*, abs/2110.14168.

Paul T Costa and Robert R McCrae. 1992. Normal personality assessment in clinical practice: The neo personality inventory. *Psychological assessment*, 4(1):5.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. *ArXiv preprint*, abs/2305.14325.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Zeyu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. AgentScope: A Flexible yet Robust Multi-Agent Platform. *ArXiv preprint*, abs/2402.14034.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *Transactions of the Association for Computational Linguistics*, 9:346-361. Place: Cambridge, MA Publisher: MIT Press.

Lewis R Goldberg. 1999. A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. *Personality psychology in Europe*, 7(1):7–28.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. *arXiv preprint*. ArXiv: 2402.01680 [cs].

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. *ArXiv preprint*, abs/2308.00352.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. 2025. Voting or consensus? decision-making in multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11640–11671, Vienna, Austria. Association for Computational Linguistics.

Venelin Kovatchev, M. Antònia Martí, and Maria Salamó. 2018. ETPC - a paraphrase identification corpus annotated with extended paraphrase typology and negation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. 2024. Improving multi-agent debate with sparse communication topology. *ArXiv preprint*, abs/2406.11776.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jessica L. Maples, Li Guan, Nathan T. Carter, and Joshua D. Miller. 2014. A test of the international personality item pool representation of the revised neo personality inventory and development of a 120-item ipip-based measure of the five-factor model. *Psychological Assessment*, 26(4):1070–1084.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

OpenAI. 2024. openai-agents-python. `https://github.com/openai/openai-agents-python`. Accessed: 2025-05-27.

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.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *ArXiv preprint*, abs/2311.12022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740.

Greg Serapio-García, Mustafa Safdari, Clément Crepy, Luning Sun, Stephen Fitz, Peter Romero, Marwa Abdulhai, Aleksandra Faust, and Maja Matarić. 2023. Personality traits in large language models. *ArXiv preprint*, abs/2307.00184.

Andries Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D. Barrett, and Arnu Pretorius. 2023. Should we be going mad? a look at multi-agent debate strategies for llms. *ArXiv preprint*, abs/2311.17371.

Aleksandra Sorokovikova, Sharwin Rezagholi, Natalia Fedorova, and Ivan P Yamshchikov. 2024. Llms simulate big5 personality traits: Further evidence. In *Proceedings of the 1st Workshop on Personalization of Generative AI Systems (PERSONALIZE 2024)*, pages 83–87.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. 2024. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *Preprint*, arXiv:2310.16049.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *ArXiv preprint*, abs/2306.03314.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *ArXiv preprint*, abs/2408.02442.

Arne Tillmann. 2025. Literature review of multi-agent debate for problem-solving. *ArXiv preprint*, abs/2506.00066.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *ArXiv preprint*, abs/2501.06322.

J. Wahle, T. Ruas, S. M. Mohammad, N. Meuschke, and B. Gipp. 2023. Ai usage cards: Responsibly reporting ai-generated content. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 282–284, Los Alamitos, CA, USA. IEEE Computer Society.

Han Wang, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. 2024a. Soft self-consistency improves language models agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 287–301, Bangkok, Thailand. Association for Computational Linguistics.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024b. Rethinking the Bounds of LLM Reasoning: Are Multi-Agent Discussions the Key? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024c. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark. *ArXiv preprint*, abs/2406.01574.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing Cognitive Synergy in Large Language Models: A

Task-Solving Agent through Multi-Persona Self-Collaboration. *arXiv preprint*. ArXiv: 2307.05300 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint*. ArXiv: 2201.11903 [cs].

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *ArXiv preprint*, abs/2308.08155.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. ExpertPrompting: Instructing Large Language Models to be Distinguished Experts. *arXiv preprint*. ArXiv: 2305.14688 [cs].

Joshua C. Yang, Damian Dailisan, Marcin Korecki, Carina I. Hausladen, and Dirk Helbing. 2024. Llm voting: Human choices and ai collective decision-making. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 7(1):1696–1708.

Yongjin Yang, Euiin Yi, Jongwoo Ko, Kimin Lee, Zhijing Jin, and SeYoung Yun. 2025. Revisiting multi-agent debate as test-time scaling: A systematic study of conditional effectiveness. *ArXiv preprint*, abs/2505.22960.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-Thought: Enhancing Large Language Model Capabilities through Cross-Model Communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15135–15153, Singapore. 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.

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. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Language Agents as Optimizable Graphs. *ArXiv preprint*, abs/2402.16823.

Appendix

A MALLM Workflow

We provide an example of how MALLM could be used to conduct experiments on MAD in Figure 4. An example discussion can be seen in Figure 5.

B System Demonstration

We provide an interactive demonstration website for MALLM. It is available under mallm.gipplab.org. A screenshot can be seen in Figure 6.

C Comparison With Other Frameworks

We compare the functionality of other commonly used frameworks for MAD with MALLM. The comparison can be seen in Table 5.

D DEBATE Examples

To demonstrate MALLM's capabilities, we construct a set of examples called DEBATE (Diverse Exchanges Between Autonomous Talking Entities), comprising 14,400 strategic problem-solving debates based on the StrategyQA dataset (Geva et al., 2021), generated using 144 distinct MALLM configurations. Each configuration combines specific settings of the framework's modular components listed in Table 4.

Parameter	Values
Response Generators	Simple, Critical, Reasoning
Persona Generators	None, Expert, IPIP
Discussion Paradigms	Memory, Relay, Report, Debate
Decision Protocols	Majority Consensus, Unanimity Consensus, Simple Voting, Approval Voting

Table 4: Parameters used for creating DEBATE. The example set comprises diverse data for each possible combination of parameters.

For example, one setup uses the `Simple` response generator, no personas, the `Memory` discussion paradigm, and the `Majority Consensus` decision protocol. Our rationale for selecting the parameters is to ensure diversity in the agent orchestration (e.g., two voting and two consensus approaches for decision-making), while keeping the computational effort manageable. All debates involve three agents, up to seven turns, and use the meta-llama/Llama-3.3-70B-Instruct

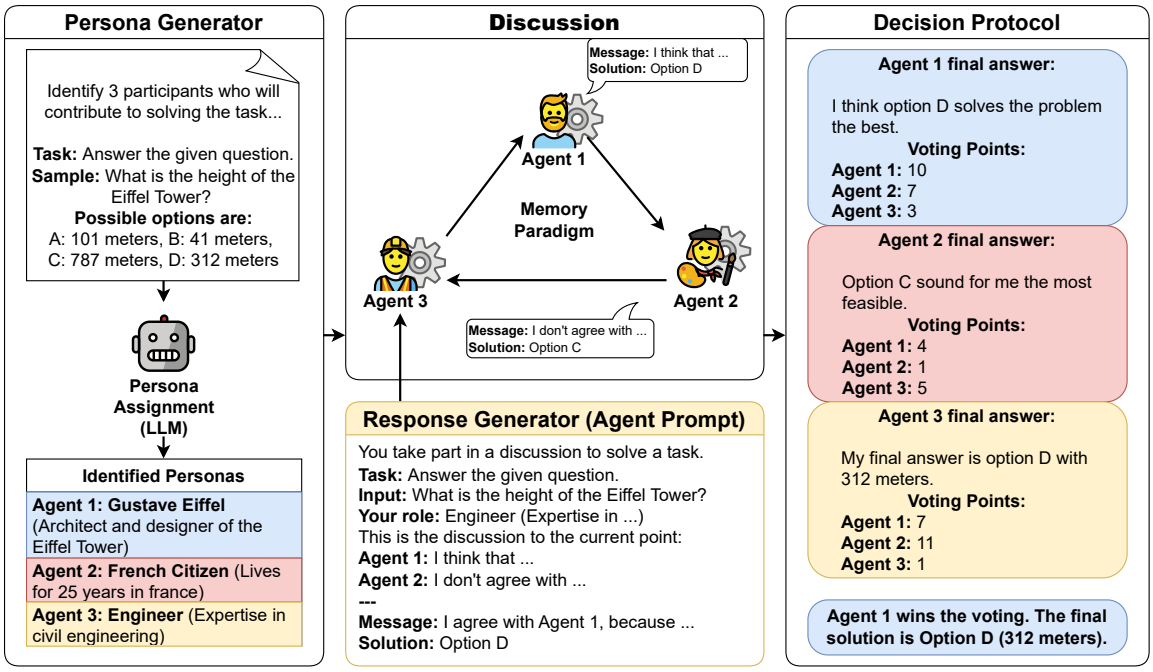


Figure 5: Example multi-agent discussion conducted in the MALLM framework. It showcases the functionality of the four modules and how they work together to get an improved final solution. First, we use the `Expert` persona generator to create three agents with different expertise. These agents discuss according to the `Memory` discussion paradigm and use the `Simple` response generator to formulate their answers. After the third turn, they begin voting using the `Cumulative Voting` decision protocol until they reach a final solution.

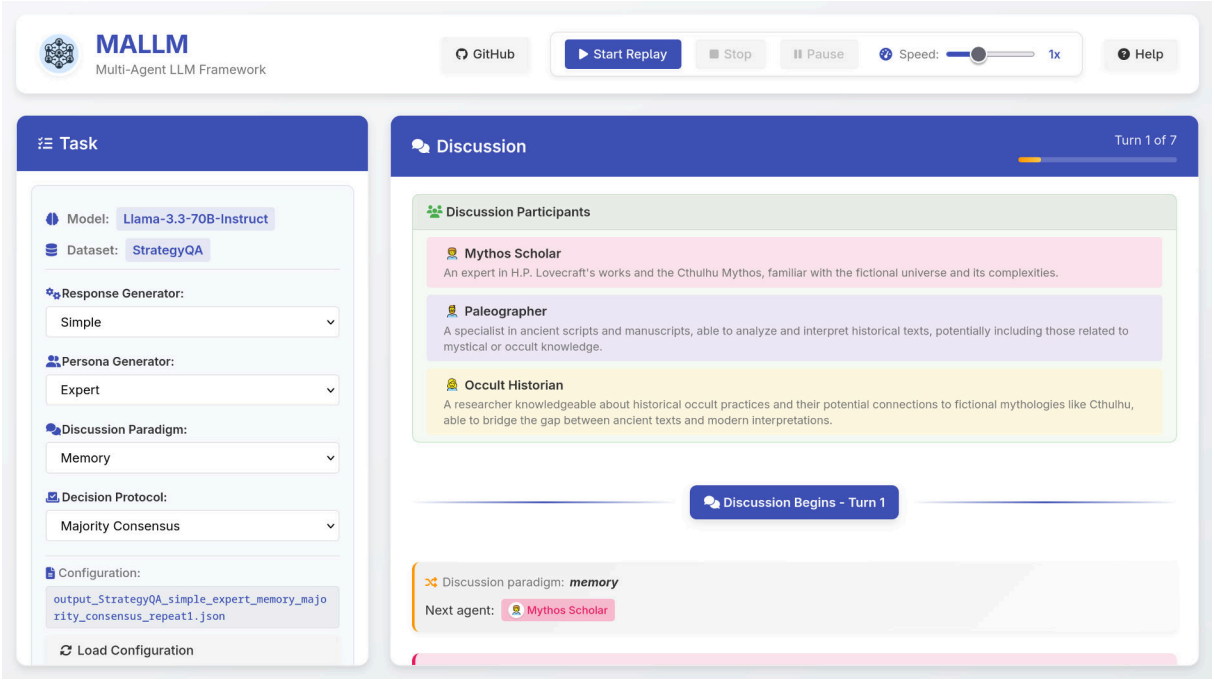


Figure 6: A screenshot of the demonstration website. One of 144 configurations for MAD can be selected on the left panel. MAD is conducted to solve the task, visible on the right panel. The top header provides functions to pause the replay or adjust the simulation speed.

model. Each of the 144 configurations runs 100 debates. For the creation of the DEBATE examples, we utilize eight NVIDIA A100 GPUs, each equipped with 40GB of VRAM, to host a meta-llama/Llama-3.3-70B-Instruct model for 8 days, 5 hours, and 42 minutes. The data is available on Hugging Face[5].

E Parameters

We adhere to default parameters for the models we used, using langchain 0.1.16 and openai 1.25.0 for the implementation of the MALLM framework.

- `temperature = 1.0`
- `top_p = 1.0`
- `presence_penalty = 0.0`
- `frequency_penalty = 0.0`
- `max_tokens = 1024`

E.1 Agent Experiments

The setup and parameters for this experiment are described in Kaesberg et al. (2025). They use *meta-llama/Meta-Llama-3-8B-Instruct* as a model for all agents with the following fixed parameters:

- Persona generator: `Expert`
- Discussion paradigm: `Memory`
- Decision protocol: Average of `Simple Voting`, `Ranked Voting`, `Cumulative Voting` and `Approval Voting`

Each experiment is repeated three times, and the average performance and standard deviation across the runs are reported.

E.2 Discussion Experiments

We use *meta-llama/Meta-Llama-3-70B-Instruct* as a model for all agents. We further report the parameters that are set fixed for this experiment:

- Persona generator: `Expert`
- Response generator: `Simple`
- Decision protocol: `Majority Consensus`

To ensure the reliability of our findings, we follow the prior work of Wang et al. (2024a) and conduct each experiment five times, reporting the average performance and standard deviation across the runs.

E.3 Decision Experiments

The setup and parameters for this experiment are described in Kaesberg et al. (2025). They use *meta-llama/Meta-Llama-3-8B-Instruct* as a model for all agents with the following fixed parameters:

[5] huggingface.co/datasets/Multi-Agent-LLMs/DEBATE

Customizable Feature	Agent Personas	Agent Responses	Discussion Paradigms	Decision Protocol	Evaluation Pipeline
AutoGen (Wu et al., 2023)	✗	✗	✗	✗	✗
GPTSwarm (Zhuge et al., 2024)	✗	✗	✓	✓	✗
OpenAI Agents SDK (OpenAI, 2024)	✓	✗	✓	✗	✓
MetaGPT (Hong et al., 2023)	✓	✓	✗	✗	✗
AgentScope (Gao et al., 2024)	✓	✗	✓	✗	✗
AutoGPT (Talebirad and Nadiri, 2023)	✓	✓	✓	✗	✗
MALLM (this work)	✓	✓	✓	✓	✓

Table 5: Comparison of customizable features across commonly used frameworks for MAD. MALLM enables the modification of agent personas, agent responses, discussion paradigms, and decision protocols. It also comes with an integrated evaluation pipeline. To the best of our knowledge, no other framework offers the same level of configurability for these main components of MAD.

- Persona generator: `Expert`
- Discussion paradigm: `Memory`
- Response Generator: `Simple`

Each experiment is repeated three times, and the average performance and standard deviation across the runs are reported.

F Evaluation Pipeline

Figures 7 to 10 show example charts generated by the MALLM evaluation pipeline. They are presented to demonstrate the pipeline's automated analysis and visualization capabilities. These specific examples are generated from experiments on the StrategyQA dataset using the *meta-llama/Meta-Llama-3-8B-Instruct* model, with error bars representing the standard deviation across three runs.

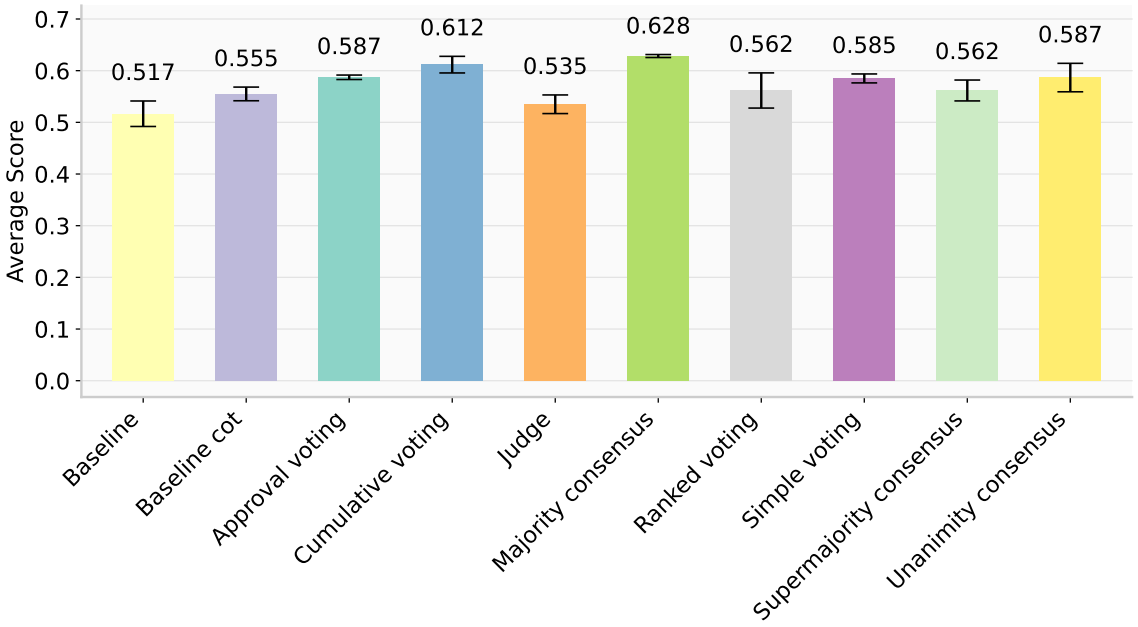


Figure 7: An example chart automatically generated by the MALLM evaluation pipeline, comparing the average performance scores of various decision protocols on the StrategyQA dataset. Error bars indicate the standard deviation over three experimental runs.

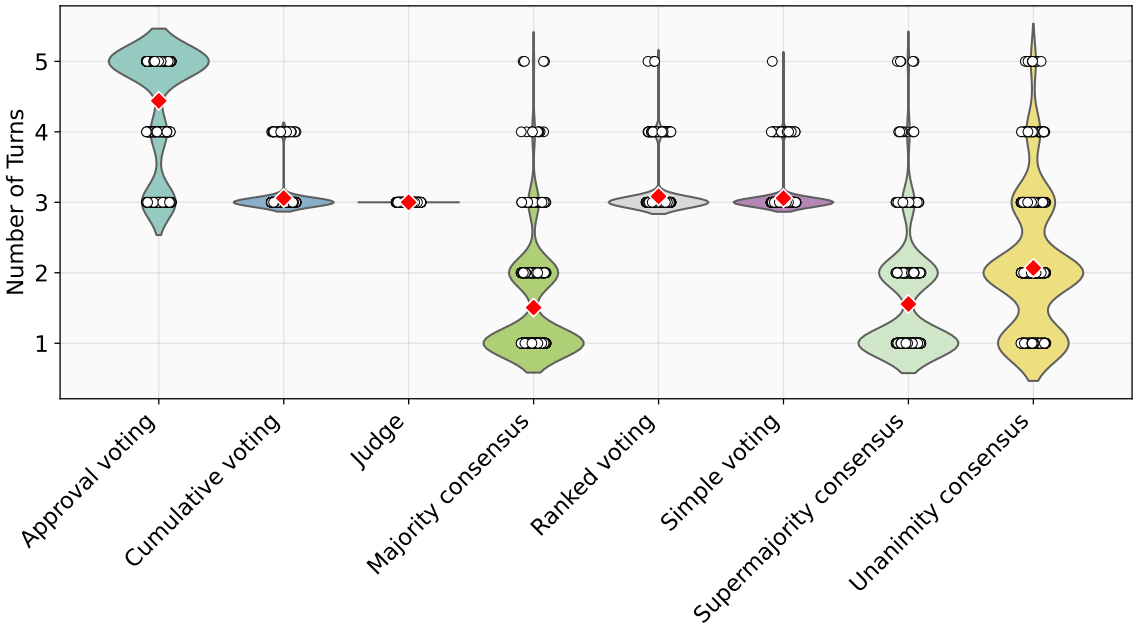


Figure 8: An example visualization from the MALLM evaluation pipeline, showing the distribution of the number of turns required for different decision protocols to converge on the StrategyQA dataset. The plot's width illustrates the frequency of turn counts, and the red marker shows the mean.

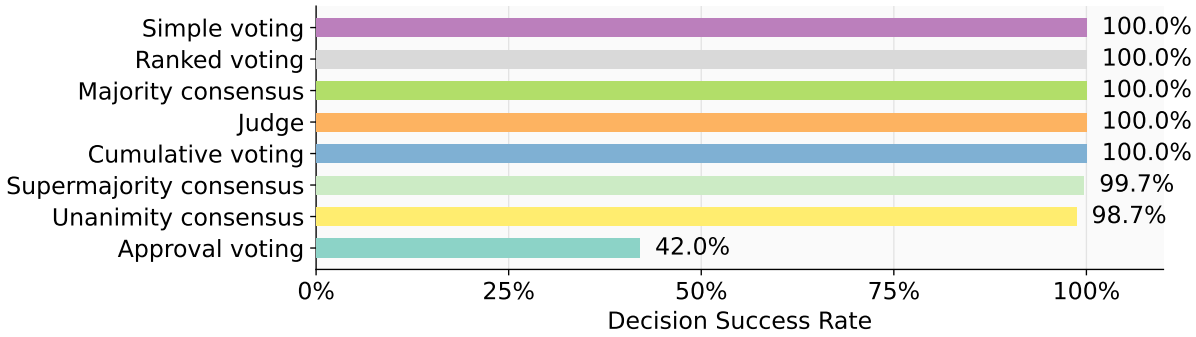


Figure 9: An example chart from the MALLM evaluation pipeline showing the decision success rates for each protocol on the StrategyQA dataset. The decision success rate explains how many of the debates reach a final solution according to the decision protocol (e.g., $> 50\%$ for `Majority Consensus`).

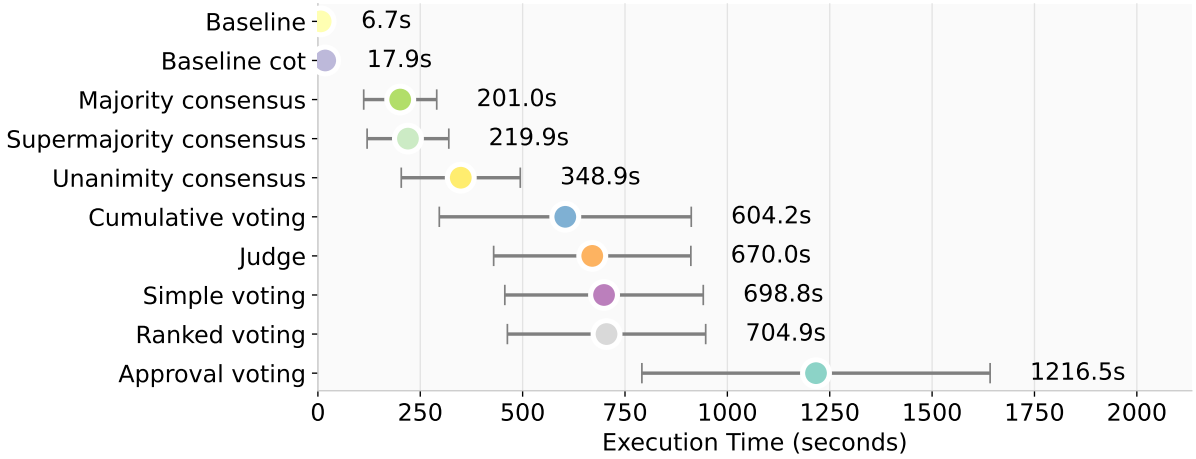


Figure 10: An example of an automatically generated chart from the MALLM evaluation pipeline, comparing the average wall clock time (in seconds) for each decision protocol on the StrategyQA dataset.

G Experiment Prompts

We provide the prompts that the MALLM framework uses to conduct MAD, which are relevant for our experiments. Appendix G.1 shows the prompt used across all configurations. Prompts specific to the persona experiments are provided in Appendix G.2, while prompts for the response generators are detailed in Appendix G.3. Additionally, prompts related to the discussion paradigm are included in Appendix G.4, and prompts for the various decision protocols are available in Appendix G.5.

G.1 General Debate

System Prompt:

You take part in a discussion to solve a task.

Your role: `<persona name>` (`<persona description>`)
Task: `<instruction>`
Input: `<example>`
Context: `<optional information>`
Current Solution: `<most recent draft>`
Discussion so far: `<agent memory>`

Figure 11: Prompt used with the `Simple` response generator for an agent participating in collaborative debate. If this is the first message of the discussion, we write "Nobody proposed a solution yet. Please provide the first one." instead of the most recent draft and agent memory.

G.2 Persona Experiments

These are the prompts used for the persona experiments.

System Prompt:

Solve the following task: `<task instruction>`
Input: `<input str>`
Make absolutely sure to provide your solution in the end: 'FINAL SOLUTION: <Letter>'.

User Prompt:

Answer the following question.

Figure 12: Base prompt used for agents participating in a GPQA experiment.

System Prompt:

When faced with a task, begin by identifying the participants who will contribute to solving the task. Provide role and description of the participants, describing their expertise or needs, formatted using the provided JSON schema.
Generate one participant at a time, complementing the existing participants to foster a rich discussion.
Example 1: `<example 1>`
Example 2: `<example 2>`
Example 3: `<example 3>`

User Prompt:

Now generate a participant to discuss the following task:
Task: `<task description>`. Please use the following examples to generate a useful persona for the task! Only answer with the JSON for the next persona.

Figure 13: Prompt used for the Expert agent generator, which creates unique personas for each example.

System Prompt:

When faced with a task, begin by identifying the participants who will contribute to solving the task. Provide role and fixed characteristics of the participant, formatted using the provided JSON schema. Generate one participant at a time, complementing the existing participants to foster a rich discussion.
You must choose the following characteristics for the participant, in JSON format:
`<characteristics and options>`
You absolutely must stick to the JSON format and the characteristics and options provided.
Example 1: `<example 1>`
Example 2: `<example 2>`

User Prompt:

Now generate a participant to discuss the following task:
Task: `<task description>`. Only answer with the JSON for the next persona! Ensure your new participant is unique.

Figure 14: Prompt used for the IPIP agent generator, which creates unique personas for each example.

G.3 Response Generator Experiments

We provide the prompts for each response generator used.

G.3.1 Simple Response Generator

User Prompt:

Based on the provided feedback, carefully re-examine your previous solution. Provide a revised solution.

Figure 15: Prompt with the `Simple` response generator instructing an agent to create a new solution based on received feedback. It is appended to the system prompt in Figure 11.

User Prompt:

Improve the current solution.
If you agree with the current solution, answer with `[AGREE]`.
Else, answer with `[DISAGREE]`, explain why, and provide an improved solution.
Let's think step-by-step.

Figure 16: Prompt with the `Simple` response generator to an agent for contributing to the current solution draft. The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

User Prompt:

Improve the current solution.
Based on the current solution, give constructive feedback. If you agree, answer with `[AGREE]`, else answer with `[DISAGREE]` and explain why.
Let's think step-by-step.

Figure 17: Prompt with the `Simple` response generator to an agent for giving feedback to the current solution draft (without directly proposing a new solution). The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

G.3.2 Critical Response Generator

User Prompt:

Re-examine the current solution critically based on the feedback provided. Ensure your revision addresses any identified weaknesses or areas for improvement. Submit a revised and improved solution.

Figure 18: Prompt with the `Critical` response generator instructing an agent to create a new solution based on received feedback. It is appended to the system prompt in Figure 11.

User Prompt:
Improve the current solution. Identify specific areas that need enhancement and propose unique solutions based on your persona. If you see no room for improvement, answer with `[AGREE]`, otherwise, answer with `[DISAGREE]` and provide a clear, solution.

Figure 19: Prompt with the `Critical` response generator to an agent for contributing to the current solution draft. The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

User Prompt:
Critically evaluate the current solution. Identify potential weaknesses or areas of improvement. If you believe the solution is flawless, answer with `[AGREE]`, otherwise answer with `[DISAGREE]` and provide constructive feedback with suggestions for improvement.

Figure 20: Prompt with the `Critical` response generator to an agent for giving feedback to the current solution draft (without directly proposing a new solution). The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

G.3.3 Reasoning Response Generator

User Prompt:
Based on the provided feedback, carefully re-examine your previous solution. Provide a revised solution.

Figure 21: Prompt with the `Reasoning` response generator instructing an agent to create a new solution based on received feedback. It is appended to the system prompt in Figure 11.

User Prompt:
Improve the current steps of the argument by referring to the other participants in the discussion. Be critical and answer short and concise. Repeat only the reasoning steps that you think are the most important. If you think there is enough information to create a final answer also answer with `[AGREE]` else answer with `[DISAGREE]`. Don't provide a final solution yet.

Figure 22: Prompt with the `Reasoning` response generator to an agent for contributing to the current solution draft. The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

User Prompt:
Based on the current solution, give constructive feedback. If you agree, answer with `[AGREE]`, else answer with `[DISAGREE]` and explain why.

Figure 23: Prompt with the `Reasoning` response generator to an agent for giving feedback to the current solution draft (without directly proposing a new solution). The agent can either agree or disagree. It is appended to the system prompt in Figure 11.

G.4 Discussion Paradigm Experiments

For the experiments on discussion paradigms, just one more prompt for the Chain-of-Thought baseline is used. We use the general prompts described in Appendix G.1 for MAD.

System Prompt:
Solve the provided task. Do not ask back questions. Clearly indicate your final solution after the text 'Final Solution:'.

Task: `<task instruction>`
Input: `<input str>`

User Prompt:
Let's think step-by-step.

Figure 24: Prompt used for the Chain-of-Thought baseline.

G.5 Decision Protocol Experiments

These are all prompts used for the decision-making protocols. The final answer extraction prompt can be seen in Figure 25. The prompt for the voting-based decision protocols can be seen in Figure 26 (Simple Voting), Figure 27 (Approval Voting), Figure 29 (Ranked Voting), and Figure 28 (Cumulative Voting) decision protocols (Figure 26 to Figure 30). The consensus decision protocol has no special prompt, as it terminates when a consensus is found, and then the final answer extraction prompt is used. Voting also utilizes the final answer extraction prompt to obtain the final answer from each agent that is used during the voting process.

G.5.1 Final Answer Extraction

System Prompt:

Your role: `<persona>` (`<persona description>`)

User Prompt:

You are tasked with creating a final solution based on the given input and your previous response.
Task: `<task>`
Input: `<input sample>`
Your previous response: `<previous answer>`
Extract the final solution to the task from the provided text. Remove statements of agreement, disagreement, and explanations. Do not modify the text. Do not output any text besides the solution. If there is no solution provided, just copy the previous response.

Figure 25: Prompt used to extract the final answer of a given agent from its previous response.

G.5.2 Voting Prompts

System Prompt:

Your role: `<persona>` (`<persona description>`)

User Prompt:

You are tasked with voting for the best solution from the list provided below based on the given task.
Task: `<task>`
Question: `<input sample>`
Here are the possible solutions:
Solution 1: `<agent 1 final answer>`
Solution 2: `<agent 2 final answer>`
Solution 3: `<agent 3 final answer>`
Based on the above solutions, please provide the number of the solution you are voting for. Answer only with the number.

Figure 26: Prompt used to get a vote from each agent for the Simple Voting decision protocol.

System Prompt:

Your role: `<persona>` (`<persona description>`)

User Prompt:

You are tasked with approving any number of solutions from the list provided below based on the given task.
Task: `<task>`
Question: `<input sample>`
Here are the possible solutions:
Solution 1: `<agent 1 final answer>`
Solution 2: `<agent 2 final answer>`
Solution 3: `<agent 3 final answer>`
Based on the above solutions, please provide the numbers of the solutions you are approving, separated by commas. Answer only with the numbers.

Figure 27: Prompt used to get a vote from each agent for the Approval Voting decision protocol.

System Prompt:

Your role: <persona> (<persona description>)

User Prompt:

You are tasked with distributing 10 points among the provided solutions based on the given task.
Task: <task>
Question: <input sample>
Here are the possible solutions:
Solution 1: <agent 1 final answer>
Solution 2: <agent 2 final answer>
Solution 3: <agent 3 final answer>
Based on the above solutions, please distribute 10 points among the solutions. Provide your points allocation as a JSON dictionary where keys are solution numbers (as int) and values are the points. The total points should sum up to 10. Answer only with the JSON dictionary.

Figure 28: Prompt used to get a vote from each agent for the Cumulative Voting decision protocol.

System Prompt:

Your role: <persona> (<persona description>)

User Prompt:

You are tasked with ranking the solutions from the most preferred to the least preferred based on the given task.
Task: <task>
Question: <input sample>
Here are the possible solutions:
Solution 1: <agent 1 final answer>
Solution 2: <agent 2 final answer>
Solution 3: <agent 3 final answer>
Based on the above solutions, please provide the rankings of the solutions separated by spaces. Example: '0 2 1' if you prefer Solution 0 the most, then Solution 2, and finally Solution 1. Provide up to 5 rankings. Only answer with the rankings.

Figure 29: Prompt used to get a vote from each agent for the Ranked Voting decision protocol.

G.5.3 Judge Prompt

User Prompt:

Task: <task>
Question: <input sample>
Please provide a decision on the following solutions and combine them in a single answer to solve the task. Only answer with the solution:
Solution 1: <agent 1 final answer>
Solution 2: <agent 2 final answer>
Solution 3: <agent 3 final answer>

Figure 30: Prompt used to get a final decision from the Judge decision protocol. No alterations are applied.

H MALLM Configuration File

Example MALLM batch configuration file for running an experiment with fixed response generator, persona generator, and discussion paradigm, but varying decision protocols. The "repeats" field defines how many times each run is repeated, which is later relevant for evaluating for robustness by the standard deviation between experiment runs. The "common" field describes parameters that are considered for all runs. The "runs" field defines the parameters unique for each individual run.

```
{
  "repeats": 3,
  "name": "<DATASET NAME>",
  "common": {
    "task_instruction_prompt_template": "<DATASET NAME>",
    "endpoint_url": "<LLM API HOSTNAME>",
    "api_key": "<LLM API KEY>",
    "model_name": "<MODEL NAME>",
    "input_json_file_path": "data/datasets/<DATASET NAME>.json",
    "concurrent_api_requests": 200,
    "num_samples": "<NUMBER OF SAMPLES>",
    "max_turns": 5,
    "response_generator":"simple"
  },
  "runs": [
    {
      "output_json_file_path": "results/baseline-cot.json",
      "use_baseline": true
    },
    {
      "output_json_file_path": "results/baseline.json",
      "use_baseline": true,
      "use_chain_of_thought": false
    },
    {
      "output_json_file_path": "results/approval.json",
      "decision_protocol": "approval_voting"
    },
    {
      "output_json_file_path": "results/cumulative.json",
      "decision_protocol": "cumulative_voting"
    },
    {
      "output_json_file_path": "results/majority_consensus.json",
      "decision_protocol": "majority_consensus"
    },
    {
      "output_json_file_path": "results/supermajority_consensus.json",
      "decision_protocol": "supermajority_consensus"
    },
    {
      "output_json_file_path": "results/unanimity_consensus.json",
      "decision_protocol": "unanimity_consensus"
    },
    {
      "output_json_file_path": "results/voting.json",
      "decision_protocol": "simple_voting"
    },
    {
      "output_json_file_path": "results/ranked.json",
      "decision_protocol": "ranked_voting"
    }
  ]
}
```

I AI-Usage Card

AI Usage Card

PROJECT DETAILS	PROJECT NAME MALLM: Multi-Agent Large Language Models Framework	DOMAIN Paper	KEY APPLICATION Multi-Agent Debate
CONTACT(S)	NAME(S) Jonas Becker	EMAIL(S) jonas.becker@uni-goettingen.de	AFFILIATION(S) University of Göttingen, LKA NRW
MODEL(S)	MODEL NAME(S) ChatGPT o3, 4.5, 4o, 5		
LITERATURE REVIEW	FINDING LITERATURE	FINDING EXAMPLES FROM KNOWN LITERATURE OR ADDING LITERATURE FOR EXISTING STATEMENTS	COMPARING LITERATURE
WRITING	GENERATING NEW TEXT BASED ON INSTRUCTIONS	ASSISTING IN IMPROVING OWN CONTENT OR PARAPHRASING RELATED WORK ChatGPT 4.5, 5	PUTTING OTHER WORKS IN PERSPECTIVE
CODING	GENERATING NEW CODE BASED ON DESCRIPTIONS OR EXISTING CODE ChatGPT o3, 4o	REFACTORING AND OPTIMIZING EXISTING CODE ChatGPT o3, 4o	COMPARING ASPECTS OF EXISTING CODE
ETHICS	WHY DID WE USE AI FOR THIS PROJECT? Efficiency, Speed, Knowledge	WHAT STEPS ARE WE TAKING TO MITIGATE ERRORS OF AI?	WHAT STEPS ARE WE TAKING TO MINIMIZE THE CHANCE OF HARM OR INAPPROPRIATE USE OF AI?

THE CORRESPONDING AUTHORS VERIFY AND AGREE WITH THE MODIFICATIONS OR GENERATIONS OF THEIR USED AI-GENERATED CONTENT

AI Usage Card v2.0 https://ai-cards.org (Wahle et al., 2023)

SWE-MERA: A Dynamic Benchmark for Agenticly Evaluating Large Language Models on Software Engineering Tasks

Pavel Adamenko[1], Mikhail Ivanov[2], Aidar Valeev[1], Rodion Levichev[1], Pavel Zadorozhny[1], Ivan Lopatin[1], Dmitrii Babaev[1], Alena Fenogenova[1], Valentin Malykh[3,2]

[1]GigaCode, [2]ITMO University, [3]MWS AI

Correspondence: mera@a-ai.ru

Abstract

The rapid advancement of Large Language Models (LLMs) in software engineering has revealed critical limitations in existing benchmarks, particularly the widely used SWE-bench dataset. Recent studies have uncovered severe data contamination issues, e.g., SWE-bench (Jimenez et al., 2023) reports 32.67% of successful patches involve direct solution leakage and 31.08% pass due to inadequate test cases. We introduce **SWE-MERA**, a dynamic, continuously updated benchmark designed to address these fundamental challenges through an automated collection of real-world GitHub issues and rigorous quality validation. Our approach implements a reliable pipeline that ensures quality while minimizing contamination risks, resulting in approximately 10,000 potential tasks with 728 samples currently available. Evaluation using the Aider coding agent demonstrates strong discriminative power in state-of-the-art models. We report performance across a dozen recent LLMs evaluated on tasks collected between September 2024 and June 2025.

1 Introduction

The complexity of real-world software development processes goes beyond merely completing code. It encompasses coding agents and a range of text-to-code tasks. E.g., SWE-bench (Jimenez et al., 2023) was created from a dataset comprising 2,294 GitHub issues and their corresponding pull requests (PRs). Each task in SWE-bench represents an authentic, real-world problem structured around: 1) The initial commit (code before changes), 2) The fixing commit (solution to the problem), 3) The issue description (what needed to be fixed). A critical limitation of this benchmark is its *static nature* — the tasks were collected only once and never updated. This leads to two major issues. *Data leakage*: As models are repeatedly tested on the same fixed dataset, they may inadvertently memorize solutions or overfit to outdated examples. *Benchmark saturation*: Over time, the benchmark loses its effectiveness as state-of-the-art models achieve near-perfect scores, making it harder to distinguish meaningful progress.

SWE-MERA addresses these shortcomings (typical for many code benchmarks) by introducing *dynamic updates* to the test cases. Regularly refreshing the dataset with new, unseen issues ensures: 1) *real-world relevance* — tasks reflect the latest challenges in software development 2) *fair evaluation* — models are tested on fresh problems, minimizing the risk of data leakage 3) *continuous improvement* — the benchmark evolves in tandem with advancements in AI and software engineering practices.

The contributions of the paper are as follows:

1. The seven-stage pipeline effectively ensures quality and minimizes contamination risks, able to collect approximately 10,000 potential tasks, with 728 samples currently available.
2. An automated scoring system based on Aider coding agent[1] and a dynamic user leaderboard [2][3].

2 Related Work

SWE-bench introduced a semi-automatic pipeline for mining software engineering tasks from popular open-source Python repositories, resulting in a benchmark of 2,294 issues and corresponding pull requests. Although this enabled a large-scale evaluation, the dataset suffered from quality issues, including poorly specified tasks and weak test coverage, which compromised the reliability of model assessment. To improve data quality,

[1] https://aider.chat

[2] SWE-MERA leaderboard

[3] The video screencast of the user's journey can be accessed through the link provided

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 440–452

SWE-bench Verified[4] released a human-validated subset of 500 tasks from SWE-bench, but this approach has limited scalability. Further work, such as SWE-Bench+ (Aleithan et al., 2024), revealed that a significant portion of the solutions in the original dataset could be "cheated" due to solution leakage in the issue or pull request descriptions, highlighting the risks of data contamination, as most issues predated significant LLM knowledge cutoffs. SWE-Bench+ addressed these issues by filtering for post-cutoff tasks and removing instances with leaked solutions, resulting in a more robust benchmark.

To expand diversity and generalizability, Multi-SWE-bench (Zan et al., 2025) extended coverage to multiple programming languages. Complementary approaches, such as SWE-Gym (Pan et al., 2024) and SWE-smith (Yang et al., 2025b), focused on automatic task generation and scalable synthetic data creation, respectively, to further increase the size and diversity of a benchmark.

While these repository-level benchmarks advance the field, they remain largely static or require substantial manual curation. In contrast, LiveCodeBench (Jain et al., 2024) pioneered a dynamic, frequently updated evaluation framework to address contamination. However, it primarily targets algorithmic problems and does not capture the repository-level complexity essential for a realistic software engineering assessment.

3 Methodology

The SWE-MERA task collection process systematically generates evaluation tasks based on real-world software engineering challenges. A comprehensive parsing of publicly available repositories was conducted to maximize coverage. For each selected repository state, tests were identified that are introduced in subsequent development but do not yet pass in the current version.

This framework enables objective assessment: after reverting the repository to the specified state and incorporating these future test cases, tests categorized as `PASS_TO_PASS` are expected to *succeed*, while those labeled `FAIL_TO_PASS` are expected to *fail*.

[4] https://openai.com/index/introducing-swe-bench-verified/

3.1 Steps overview

To ensure the transparency and reproducibility of the task generation process, we have designed a well-documented and accessible collection pipeline.

This pipeline is executed on a monthly basis, enabling the systematic and continuous collection of tasks over time. By adhering to this schedule, we facilitate the regular updating and expansion of our dataset, ensuring that it remains current and reflective of ongoing developments in the software engineering domain.

In practice, we plan to execute the pipeline on a quarterly basis, with each run collecting tasks for the preceding three months (month by month).

The pipeline comprises the following steps:

1. **Repository Selection:** GitHub repositories are selected based on predefined criteria, including a minimum threshold of 10 stars and 10 forks, recent activity within the current year, Python as the primary programming language, and the presence of an open-source license.
2. **PR-Issue Mapping Construction:** Mappings between issues and pull requests are constructed according to the following criteria:
   - Each pull request is associated with exactly one issue (either linked directly or via comments).
   - Each issue is associated with exactly one pull request.
   - The pull request is merged.
   - The associated issue is closed.
   - The pull request merge date is later than the first day of the previous month.
3. **Metadata Extraction and Filtering:** For each selected issue and its corresponding pull request, metadata (including title, text, and comments) is downloaded and parsed. The issue-PR pairs are then filtered out if the combined length of the issue title and the issue body is less than 25 characters.
4. **Patch Extraction and Validation:** For each pull request, the corresponding `git diff` is generated and validated. Only examples that modify both source code and test files are retained. Additionally, only pull requests that modify fewer than 15 source files are considered.
5. **Repository Build Validation:** For each task, we build an appropriate environment in a

Step	Total Repositories	Total Issues	Time Estimation
GitHub			
all public	255M	522M	1 sec.
Python	21M	53M	1 sec.
10+start, 10+forks	168K	5.7M	7 hours
1+ closed issue	97K	5.7M (4.2M†)	7 hours
Repository Selection			
Python, 10+start, 10+forks, repo updated at 2025	110K	5.5M	7 hours
PR-Issue Mapping Construction			
1+ updated issue at [2025-01-01, 2025-06-01]	25K	339K	3 days
issue is closed, closed merge request	10K	98K	3 days
one-to-one mapping beetween issse and pr	8.4K	55K	2 min.
Metadata Extraction and Filtering	8.2K	51K	11 hours
Patch Extraction and Validation	6.7K	30K	12 hours
Repository Build Validation	1.6K	9K	3 hours
End-to-End Task Execution	668	1.6K	2 days
LLM-based pipeline evaluation	279	528	2 hours

Table 1: Summary of the task collection funnel for the period 2025-01-01 to 2025-06-01, calculated using the GitHub GraphQL API. For our experiments, Repository Build Validation was performed immediately before PR-Issue Mapping Construction to minimize total processing time; this table is provided for reference.
† Only closed issues.

Docker container. Validation is considered successful if `pytest` returns at least one passed test.

6. **End-to-End Task Execution:** Each generated task is executed in a controlled environment within a Docker container to verify its reproducibility and correctness. A detailed description can be found in Appendix A.
7. **LLM-based Pipeline Evaluation:** To assess the quality of each candidate task, we use the Qwen3-32B model (Yang et al., 2025a) to evaluate the description, patch, and associated tests on four criteria: `task correctness`, `test correctness`, `test completeness`, and `complexity`. The model is prompted to return a structured JSON response with a score of 1 to 10, a confidence value (0.0–1.0), and a brief explanation for each criterion (see Appendix D for the prompt used).
   We filter out tasks that fall into the *bottom quartile* (lower 25%) of the score distribution for any of the following dimensions:
   - `task correctness`
   - `test correctness`
   - `test completeness`

   The `complexity` score is not used for filtering, as we explicitly aim to retain both easy and difficult tasks. This filtering step ensures that retained tasks are well-formed, solvable, and adequately tested.

The detailed results of each pipeline step are summarized in Tab. 1. The sample collected task can be found in Appendix E.

3.2 Availability

The entire pipeline is implemented as a Python package[5] and can be executed for any GitHub repository, facilitating reproducibility and extensibility for future research.

All tasks are typically executed within Docker containers using a standardized base image[6].

However, the same execution process can be replicated in a Conda environment without modification to the underlying code.

4 Evaluation

To apply LLMs in issue-solving scenarios, we employ a popular agentic framework Aider, which performs similarly to other state-of-the-art frameworks when applied to the same models. Aider gives six attempts ("tries") to LLMs to fix a given issue, while every attempt allows up to four reflections to lint or test output. We slightly modify Aider [7] and Aider-SWE-bench[8] repositories due to

[5] `https://pypi.org/project/repositorytest`; source code for the package can be found here.
[6] https://hub.docker.com/layers/library/python/3.11
[7] github.com/Aider-AI/aider@4f4b10fd
[8] github.com/Aider-AI/aider-swe-bench@6e98cd6

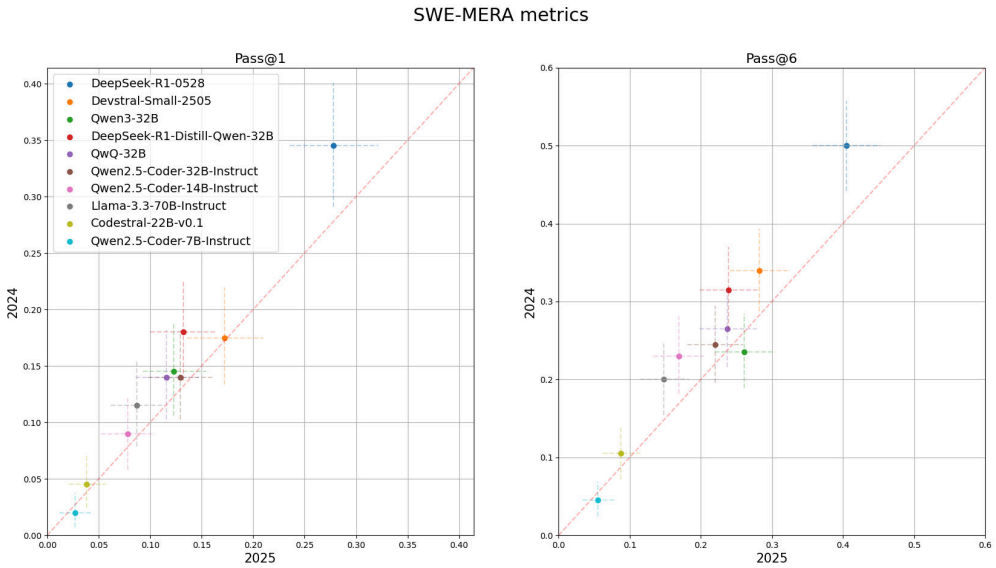


Figure 1: Comparison of model pass@1/pass@6 metrics between two years. Error bars represent confidence intervals, computed using the binomial distribution (5% two-sided quantile).

backward compatibility issues. However, we aim to support several popular frameworks capable of solving issues in the near future.

Aider gives *six* consequent independent attempts to LLMs to fix issues, but if an LLM succeeds in early attempts, it moves to the next issue. Due to the experimental design, we report two metrics: pass@1, indicating whether the first attempt was successful, and pass@6, indicating whether any of the six attempts succeeded.

To assess benchmark reliability, we selected several popular state-of-the-art LLMs for code, including Qwen, Devstral, DeepSeek-R1, and others (see the full list in the next section). We ran these models on 8 NVIDIA H100 80 GB with exceptions for DeepSeek-R1 and 7B models run on 16 and 4 GPUs, respectively. The evaluation for a single model took, on average, 3 ± 1 hours for Aider runs and half an hour to test final patches. This experiment used 60-140M prompt tokens and 3-20M completion tokens, corresponding to 14-20K prompt and 1-4K completion tokens per request.

4.1 Evaluated Models

Codestral-22B-v0.1[9] is a model trained by Mistral AI on a diverse dataset of 80+ programming languages, including the most popular ones, such as Python, Java, C, C++, JavaScript, and Bash.

Qwen2.5-Coder-7,14,32B-Instruct models (Hui et al., 2024) are the latest Qwen LLMs designed for code, available in multiple parameter sizes, affording flexibility between resource usage and performance. Qwen2.5 Coder models significantly improve in code generation, code reasoning, and code fixing, and have a more comprehensive foundation for real-world applications such as Code Agents.

Llama-3.3-70B-Instruct[10] multilingual large language model (LLM) is an instruction-tuned generative model. The Llama 3.3 instruction-tuned text-only model is optimized for multilingual dialogue use cases. Note that it is a general-purpose chat model, not specifically designed for code.

DeepSeek-R1-0528 by DeepSeek AI (DeepSeek-AI, 2025) incorporates computational enhancements and novel post-training optimizations to significantly improve reasoning, inference, and problem-solving capabilities. The updated model achieves state-of-the-art benchmark performance with reduced hallucination rates while advancing code generation and function calling. As open-source software, it democratizes access to advanced reasoning capabilities.

DeepSeek-R1-Distill-Qwen-32B is a Qwen2.5 32B model (Yang et al., 2024) distilled on the reasoning data generated by DeepSeek-R1 (DeepSeek-AI, 2025). The distilled models demonstrated exceptionally high performance on other benchmarks.

QwQ-32B is a reasoning model developed by the

[9] https://mistral.ai/news/codestral

[10] https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3/

Model	pass@1	pass@6	localize files	generate patch	regression tests	token limit hit
DeepSeek-R1-0528	**27.8%**	**40.2%**	89.6%	98.1%	40.6%	0.2%
Devstral-Small-2505	17.2%	28.2%	89.1%	98.7%	28.5%	0.4%
Qwen3-32B	12.3%	26.1%	91.8%	98.9%	26.1%	1.1%
DeepSeek-R1-Distill-Qwen-32B	13.2%	23.9%	87.8%	98.7%	23.7%	0.4%
QwQ-32B	11.6%	23.7%	79.6%	96.6%	22.9%	0.6%
Qwen2.5-Coder-32B-Instruct	12.9%	22.0%	86.3%	96.3%	22.0%	4.6%
Qwen2.5-Coder-14B-Instruct	7.8%	16.9%	86.0%	93.7%	16.8%	1.9%
Llama-3.3-70B-Instruct	8.7%	14.8%	77.0%	70.9%	14.8%	0.0%
Codestral-22B-v0.1	3.8%	8.7%	76.7%	83.4%	8.4%	2.9%
Qwen2.5-Coder-7B-Instruct	2.7%	5.5%	58.2%	55.3%	4.8%	5.7%

Table 2: Evaluation results of models on SWE-MERA 2025. 'localize files' is the percentage of attempts correctly identifying files to fix; 'generate patch', those producing valid patches; 'regression tests', those where patches pass original repository tests; and 'token limit hit', those exceeding the 32k token context limit.

Qwen Team (Team, 2025), which achieves competitive performance compared to state-of-the-art reasoning models.

Qwen3-32B (Yang et al., 2025a) supports 119 languages and features a unique dual-mode architecture enabling efficient switching between complex reasoning ("thinking mode") and dialogue ("non-thinking mode"). This architecture delivers significant performance improvements in reasoning, code generation, and creative dialogue, surpassing prior Qwen models. Qwen3 also demonstrates leadership in agent integration and multilingual task performance.

Devstral-Small-2505[11] is an open-source agentic language model for software engineering, developed by Mistral AI and All Hands AI, excelling at codebase exploration, multifile editing, and software engineering tool usage.

We provide more information on the baselines in Appendix B.

5 Results

Tab. 2 presents the evaluation results for state-of-the-art LLMs on the 2025 subset of SWE-MERA using the Aider agent workflow. The results indicate that SWE-MERA accurately ranks the Qwen2.5-Coder models according to their size. In addition, Qwen3-32B (Yang et al., 2025a) slightly outperforms QwQ-32B, which is consistent with the declared model specifications. In particular, Devstral-Small-2505 demonstrates superior performance as reported in its release notes[11], despite its smaller size.

We have found an interesting feature while comparing the results for the evaluation of the baselines for 2024 and 2025. It seems that DeepSeek-R1, both the 0528 and Distill-Qwen-32B versions, perform better on 2024 tasks. The other investigated models do not show such behaviour. The results are visualized in Fig. 1. More detailed results are presented in Appendix C.

6 Discussion

Scaling We observed that the GitHub API rate limits are sufficient to collect all relevant tasks from the past month using a single GitHub token in two days, which is surprisingly fast.

Currently, we have collected approximately 700 tasks. If we do not restrict our benchmark to the last 6 months, we estimate the number of collected tasks to be 10,000; however, achieving this scale will require additional effort, particularly for End-to-End execution tasks.

Malicious Software In our security assessment, we scanned 668 repositories for known virus signatures and discovered two repositories that, while suitable for SWE benchmarking, also exhibited the signatures:

- `https://github.com/DataDog/guarddog` An open-source security scanner designed to detect vulnerabilities and malicious dependencies in software supply chains.

- `https://github.com/fkie-cad/socbed` A framework for simulating and evaluating security operations center (SOC) environments, supporting research in cyber defense and attack scenarios.

This finding underscores the need to integrate basic virus signature checking into our system to ensure the integrity and safety of the collected repositories.

[11] `https://mistral.ai/news/devstral`

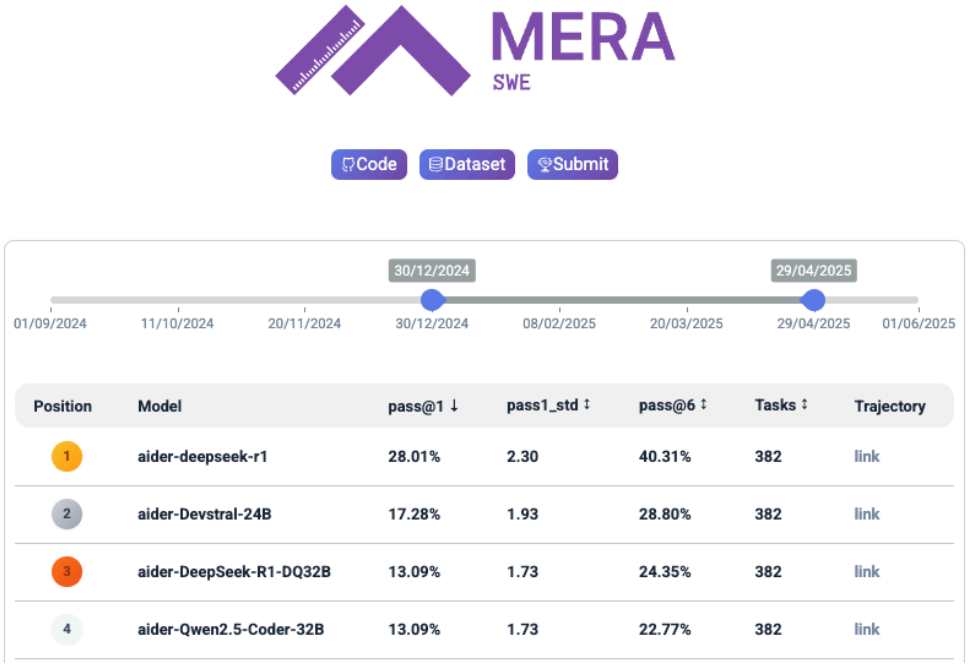


Figure 2: Screenshot of the SWE-MERA evaluation platform web interface.

Complexity Our experiments demonstrate that the last step of our pipeline, namely the LLM-based evaluation, is crucial to maintain task quality. Automated collection methods may yield tasks that are either excessively complex due to insufficient information in the issue description or overly trivial when the solution is explicitly provided. The LLM-based assessment effectively filters such cases, ensuring that only tasks of appropriate complexity and relevance are retained.

7 System Demonstration

The SWE-MERA evaluation platform provides a reproducible and transparent environment to benchmark software engineering agents. The benchmark can be accessed at the link. The web interface features an interactive slider, enabling users to visualize evaluation metrics across different dates and to inspect potential contamination events in the dataset, as shown in Fig. 2.

Submission Workflow To participate in the evaluation and have your agent's results displayed on the leaderboard, one should follow these steps:

1. *Dataset Acquisition:* Download the SWE-MERA dataset from the Hugging Face repository[12].
2. *Agent Execution:* Run a software engineering agent on the provided dataset.
3. *Submission:* Submit the results by creating a pull request to the evaluation repository.
4. *Validation and Leaderboard Update:* Submissions are reviewed and, within two working days, valid results are integrated into the public leaderboard.

A schematic overview of the submission process is provided in Fig. 2.

Dataset and Evaluation Updates The SWE-MERA dataset is updated monthly and is available through the Hugging Face platform. Participants are encouraged to include links to their agent's execution trajectories; otherwise, the system will default to displaying data from the corresponding GitHub pull request.

Leaderboard and Data Visibility The dashboard is automatically updated to reflect new submissions. If a model does not have sufficient data to compute evaluation metrics for a selected time period, it will not be displayed on the leaderboard for that interval.

Interface Features

- The web interface includes a slider for temporal navigation of metrics.
- Users can inspect detailed evaluation results and identify potential overfitting or contami-

[12] https://huggingface.co/datasets/MERA-evaluation/SWE-MERA

nation issues.

- The system supports transparent and reproducible evaluation, with all data and code accessible via public repositories.

8 Conclusion

SWE-MERA introduces a new approach to evaluating software engineering tasks, effectively addressing key limitations through dynamic data collection, automated quality validation, and ongoing updates to the dataset. This method helps mitigate concerns about data contamination while improving both task quality and the reliability of evaluations.

The benchmark demonstrates strong discriminative power across state-of-the-art models and establishes reliable performance baselines that are free from the contamination issues often found in traditional static benchmarks. With its extensible design and community-driven approach, SWE-MERA serves as a vital resource to advance AI research in software engineering.

The framework's adaptable design allows for expansion to programming languages such as Java, JavaScript, TypeScript, Go, and C++ by employing a standardized metadata approach. Future developments will focus on improving visualization capabilities, refining quality metrics, and integrating more closely with intelligent coding systems.

Limitations

While the dynamic collection of coding problems in our benchmark framework presents distinct advantages, it also introduces several important limitations.

First, dynamically collected tasks, while allowing for scalability and novelty, may lack the nuanced complexity and creativity found in carefully curated or human-authored problems. Automatically constructed problems may inadvertently result in unnaturally phrased prompts, incomplete specifications, or tasks that are either trivial or excessively convoluted, which can compromise the validity and diversity of the benchmark.

Second, evaluating model performance on dynamically generated problems poses challenges for ground truth and grading quality. Automated reference solutions and test cases may not exhaustively capture all correct or optimal solutions, especially for open-ended or ambiguous problems. As a result, our metrics may underestimate model capabilities on creative or alternative approaches, and automated correctness checks may yield false negatives.

Third, ensuring the quality and fairness of dynamically collected problems is inherently difficult. It is possible for the generation process to introduce biases, such as overrepresenting certain programming paradigms, languages, or styles while underrepresenting others. This may affect the generalizability of evaluation results and obscure weaknesses of LLMs in underrepresented domains.

Fourth, although dynamic generation reduces risks of memorization and contamination from training data, it does not wholly eliminate them. For models trained on vast internet datasets, generated problems may still resemble well-known canonical challenges or textbook exercises, and thus performance may reflect prior exposure rather than true generalization abilities.

Fifth, the infrastructure for dynamic problem generation and grading brings additional technical complexity and potential instability. Failures in problem construction, test case generation, or sandboxed code execution can introduce noise into evaluation results and limit the reproducibility of benchmarking runs.

Finally, our current benchmark focuses primarily on programming correctness. Other crucial aspects of software engineering — such as code readability, maintainability, efficiency, security, and teamwork — are not evaluated in this framework and remain open challenges for future work.

Ethical Statement

The introduction of a dynamically collected benchmark for evaluating LLM coding abilities raises several ethical considerations.

First, all prompts, solutions, and test cases produced by the dynamic generation system have been constructed to avoid the unintentional inclusion of proprietary, copyrighted, or sensitive information. The generation process is based solely on open-source templates, algorithmic patterns, and public domain resources, minimizing the risk of intellectual property infringement.

Second, although dynamically generated problems reduce the risks of data contamination and memorization in models, they do not fully mitigate the potential for LLMs to generate unsafe, insecure, or malicious code. We urge users to apply the benchmark ethically and to avoid using it—or the

resulting models—for uses that may cause harm or violate responsible AI guidelines.

Third, by making dynamic generation tools and evaluation infrastructure publicly available, we strive to foster transparency, reproducibility, and equitable access to research resources. However, we recognize the potential for technology misuse, including the generation of synthetic coding tests for automated cheating on educational platforms or biasing LLM performance reviews for commercial interests. We recommend responsible stewardship, encourage open discussion of these risks, and welcome feedback from the broader community.

Fourth, while programmatically generated problems have clear scalability and adaptability benefits, there are potential risks of unintended bias in the selection or phrasing of tasks, which could disadvantage certain groups or languages. We commit to ongoing evaluation and refinement of the benchmark to ensure fairness, inclusivity, and diversity in the problems presented.

Lastly, we note that the widespread adoption of automated coding benchmarks has implications for education, employment, and the wider software ecosystem. Benchmarks should augment, rather than replace, comprehensive, human-centric evaluation of programming skills and ethical development practices.

AI-assistants Help We improve and proofread the text of this article using Writefull assistant integrated in Overleaf (Writefull's/Open AI GPT models) and GPT-4o[13], Grammarly[14] to correct grammatical, spelling, and style errors and paraphrase sentences. We underline that these tools are used strictly to enhance the quality of English writing, in full compliance with the ACL policies on responsible use of AI writing assistance. Nevertheless, some segments of our publication can be potentially detected as AI-generated, AI-edited, or human-AI-generated.

References

Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. 2024. Swe-bench+: Enhanced coding benchmark for llms. *arXiv preprint arXiv:2410.06992*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

[13] https://chatgpt.com

[14] https://app.grammarly.com/

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. 2024. Training software engineering agents and verifiers with swe-gym. *arXiv preprint arXiv:2412.21139*.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

John Yang, Kilian Leret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. 2025b. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*.

Daoguang Zan, Zhirong Huang, Wei Liu, Hanwu Chen, Linhao Zhang, Shulin Xin, Lu Chen, Qi Liu, Xiaojian Zhong, Aoyan Li, and 1 others. 2025. Multi-swe-bench: A multilingual benchmark for issue resolving. *arXiv preprint arXiv:2504.02605*.

A Repository Build Validation Procedure

The detailed validation process consists of the following steps:

1. **Environment setup:**

```
pip install . && \
pip install pytest pytest-json-report
```

2. **Test execution:**

```
pytest --json-report \
 --json-report-file=report_pytest.json
```

3. **Success criteria:** Validation succeeds if:
    - The command completes without errors.
    - The JSON report shows more than 0 passed tests.
4. **Artifact preservation:** On success:
    - The Dockerfile is saved for future image rebuilding.
    - Docker cache is optimized for fast container recreation.

B Baselines

More details of the baselines used are provided in Tab. 3.

C Additional Results

Figs. 3&4 depict the results of the models compared to their respective sizes. The results here are the same as those in Tab. 2.

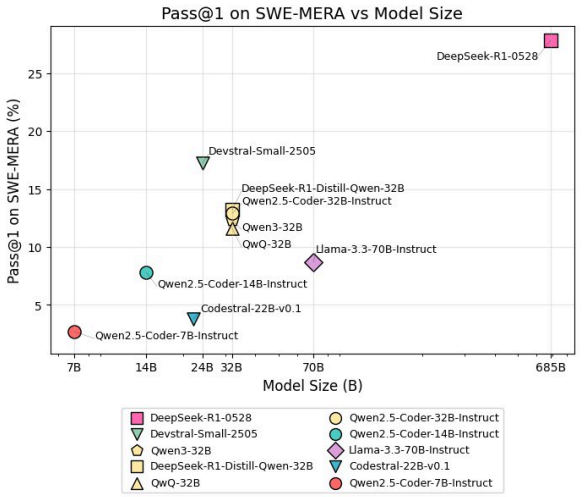


Figure 3: Pass@1 results vs model size for all evaluated models.

Tab. 4 contains results of the evaluation on tasks dated 2024 year only. A year-over-year comparison (Tab. 2 and 4) shows that DeepSeek-R1 decreases from 50% to 40.2%, which is larger than all other models on average. Devstral-Small-2505 from 34% to 28.2%, DeepSeek-R1-Distill-Qwen-32B from 31.5% to 23.9%, and Llama-3.3-70B-Instruct from 20% to 14.8%. Moreover, Qwen2.5-Coder-14B-Instruct achieves a 23% pass@6 rate on 2025 data, which is similar to the results obtained by 32B models, whereas its pass@1 rate remains notably lower than that of the larger models.

Fig. 5 represents the additional evaluation of the baselines in a more strict setup, where we take only

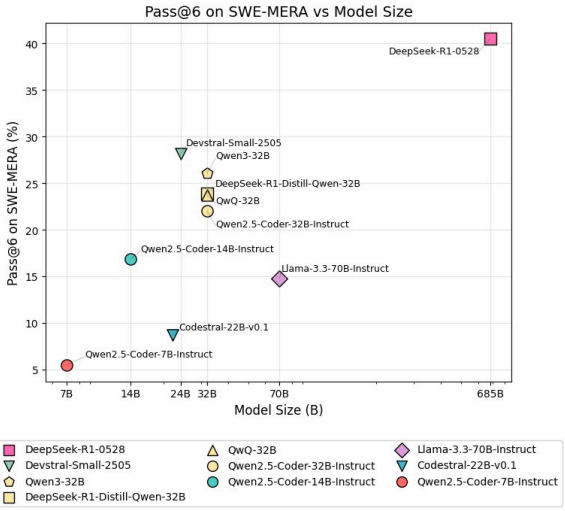


Figure 4: Pass@6 results vs model size for all evaluated models.

the top decile (top 10%) of all the tasks. Here, the year-to-year differences in behavior of the models are more subtle, namely, only DeepSeek-R1 shows a statistically valid discrepancy measured in pass@6.

Model	Size	Release Date	Designed for code
Codestral-22B-v0.1	22B	May 29, 2024	yes
Qwen2.5-Coder-{7,14,32}B-Instruct	7B, 14B, 32B	November 12, 2024	yes
Llama-3.3-70B-Instruct	70B	December 6, 2024	no
DeepSeek-R1-Distill-Qwen-32B	32B	January 20, 2025	no
QwQ-32B	32B	March 5, 2025	no
Qwen3-32B	32B	April 28, 2025	no
Devstral-Small-2505	24B	May 21, 2025	yes
DeepSeek-R1	671B (37B active)	May, 28, 2025	no

Table 3: Evaluated models specification.

Model	pass@1	pass@6	localize files	generate patch	regression tests	token limit hit
DeepSeek-R1-0528	**34.5%**	**50.0%**	90.9%	98.0%	49.7%	0.0%
Devstral-Small-2505	17.5%	34.0%	92.0%	99.0%	33.5%	0.5%
DeepSeek-R1-Distill-Qwen-32B	18.0%	31.5%	89.4%	98.5%	32.7%	0.0%
QwQ-32B	14.0%	26.5%	81.5%	96.5%	25.5%	0.5%
Qwen2.5-Coder-32B-Instruct	14.0%	24.5%	91.5%	98.0%	24.6%	7.0%
Qwen3-32B	14.5%	23.5%	94.5%	96.5%	25.1%	1.0%
Qwen2.5-Coder-14B-Instruct	9.0%	23.0%	85.0%	95.0%	22.5%	3.0%
Llama-3.3-70B-Instruct	11.5%	20.0%	84.4%	79.9%	19.6%	0.0%
Codestral-22B-v0.1	4.5%	10.5%	81.5%	84.5%	10.5%	3.5%
Qwen2.5-Coder-7B-Instruct	2.0%	4.5%	68.8%	57.8%	4.0%	4.5%

Table 4: Evaluation results of models on SWE-MERA 2024. ‘localize files‘ is the percentage of attempts correctly identifying files to fix; ‘generate patch‘, those producing valid patches; ‘regression tests‘, those where patches pass original repository tests; and ‘token limit hit‘, those exceeding the 32k token context limit.

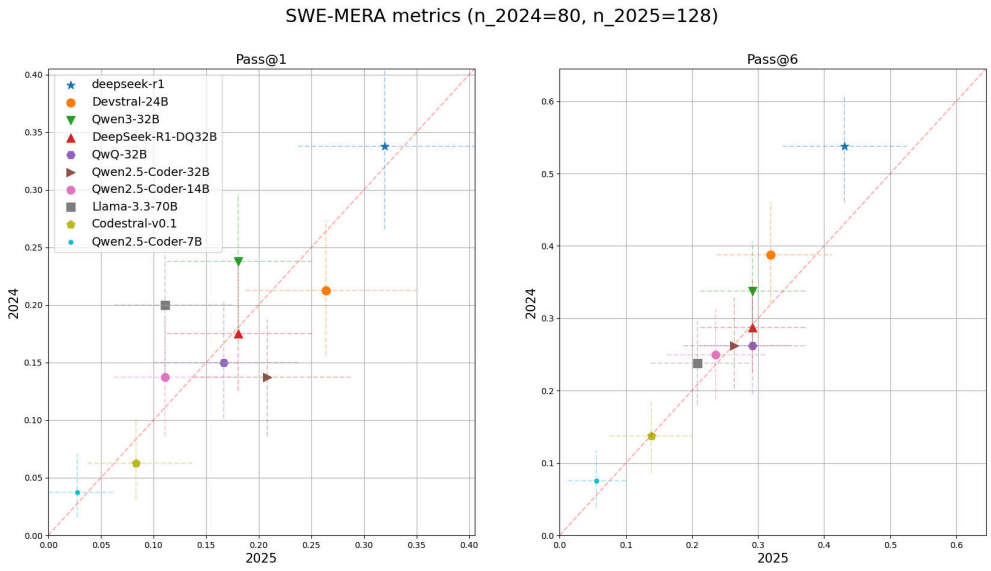


Figure 5: Comparison of model pass@1 and pass@6 metrics between two years. Error bars indicate confidence intervals, computed using the binomial distribution with a 5% two-sided quantile. To address the variability in task sets across years, we include only those tasks for which $\min(\text{task\_correctness}, \text{test\_correctness}, \text{test\_completeness}) \geq 9$.

D Prompt for LLM-based Task Evaluation

We used the following prompt to evaluate the quality of candidate tasks using the Qwen3-32B model:

```
Conduct a comprehensive evaluation of the programming task solution
based on four criteria.

TASK:
{problem_statement}

SOLUTION:
{patch}

TESTS:
{test_patch}

Evaluate based on the following criteria:

1. TASK CORRECTNESS: Does the solution (patch) correctly solve the described problem?
2. TEST CORRECTNESS: Do the tests cover the problem from the task description?
3. COMPLEXITY: What is the complexity of solving this task?
4. TEST COMPLETENESS: Do the tests cover corner cases from the problem description?

Respond in JSON format:
{
    "task_correctness": {
        "score": <a score from 1 to 10>,
        "confidence": <a confidence score from 0.0 to 1.0>,
        "reasoning": "<explanation>"
    },
    "test_correctness": {
        "score": <a score from 1 to 10>,
        "confidence": <a confidence score from 0.0 to 1.0>,
        "reasoning": "<explanation>"
    },
    "complexity": {
        "score": <a score from 1 to 10>,
        "confidence": <a confidence score from 0.0 to 1.0>,
        "reasoning": "<explanation>"
    },
    "test_completeness": {
        "score": <a score from 1 to 10>,
        "confidence": <a confidence score from 0.0 to 1.0>,
        "reasoning": "<explanation>"
    }
}
```

E SWE-MERA Task Example

Problem Statement

```
Performance threshold goes to -inf when it should be zero.
```

```
In attempting to create a performance test where zero is the correct value,
I created the following reference (since a value of zero results in no reference
check being performed; see https://github.com/reframe-hpc/reframe/issues/2857):

    self.reference = {
        '*': {
            'Gflops': (None, None, None, 'Gflops'),
            'Exponent': (None, None, None, '10exp'),
            'Time': (None, None, None, 'seconds'),
            'failed_tests': (.1, -1.0, 0, 'tests'),
            'skipped_tests': (.1, -1.0, 0, 'tests')
    }}

I was thinking for the failed and skipped tests, this would create a lower bound
of zero and upper bound of .1, allowing zero to pass, but integers larger than
that would fail.
...
```

Test Patch (Diff)

```
diff --git a/unittests/test_sanity_functions.py b/unittests/test_sanity_functions.py
index 7e6368938..0e9367027 100644
--- a/unittests/test_sanity_functions.py
+++ b/unittests/test_sanity_functions.py
@@ -473,6 +473,18 @@ def test_assert_reference():
                              r'\(l=-1\.2, u=-0\.9\)'):
         sn.evaluate(sn.assert_reference(-0.8, -1, -0.2, 0.1))

+    # Check that bounds are correctly calculated in case that lower bound
+    # reaches zero (see also GH issue #3430)
+    with pytest.raises(SanityError,
+                       match=r'1 is beyond reference value 0\.1 '
+                             r'\(l=0\.0, u=0\.1\)'):
+        assert sn.assert_reference(1, 0.1, -1.0, 0)

+    with pytest.raises(SanityError,
+                       match=r'-1 is beyond reference value -0\.1 '
+                             r'\(l=-0\.1, u=-0\.0\)'):
+        assert sn.assert_reference(-1, -0.1, 0, 1.0)

     # Check invalid thresholds
     with pytest.raises(SanityError,
                        match=r'invalid high threshold value: -0\.1'):
```

Gold Patch (Diff)

```
diff --git a/reframe/utility/sanity.py b/reframe/utility/sanity.py
index a4f57a301..1228586ff 100644
--- a/reframe/utility/sanity.py
+++ b/reframe/utility/sanity.py
@@ -8,6 +8,7 @@
```

```
 import contextlib
 import glob as pyglob
 import itertools
+import math
 import os
 import re
 import sys
@@ -576,8 +577,14 @@ def calc_bound(thres):

         return ref*(1 + thres)

-    lower = calc_bound(lower_thres) or float('-inf')
-    upper = calc_bound(upper_thres) or float('inf')
+    lower = calc_bound(lower_thres)
+    if lower is None:
+        lower = -math.inf

+    upper = calc_bound(upper_thres)
+    if upper is None:
+        upper = math.inf
...
```

OPEN-THEATRE: An Open-Source Toolkit for LLM-based Interactive Drama

Tianyang Xu[◇1], **Hongqiu Wu**[◇2,3,4], **Weiqi Wu**[2,3,4], **Hai Zhao**[†2,3,4]
[1] UM–SJTU Joint Institute, Shanghai Jiao Tong University
[2] AGI Institute, School of Computer Science, Shanghai Jiao Tong University
[3] Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University
[4] Shanghai Key Laboratory of Trusted Data Circulation and Governance in Web3
{johnnie.walker,wuhongqiu}@sjtu.edu.cn

Abstract

LLM-based Interactive Drama introduces a novel dialogue scenario in which the player immerses into a character and engages in a dramatic story by interacting with LLM agents. Despite the fact that this emerging area holds significant promise, it remains largely underexplored due to the lack of a well-designed playground to develop a complete drama. This makes a significant barrier for researchers to replicate, extend, and study such systems. Hence, we present OPEN-THEATRE, the first open-source toolkit for experiencing and customizing LLM-based interactive drama. It refines prior work with an efficient multi-agent architecture and a hierarchical retrieval-based memory system, designed to enhance narrative coherence and realistic long-term behavior in complex interactions. In addition, we provide a highly configurable pipeline, making it easy for researchers to develop and optimize new approaches.

1 Introduction

Interactive drama (Ryan, 1997; Dow et al., 2007) offers a novel participatory storytelling paradigm where users engage as an in-story character. In LLM-based interactive drama, the in-story characters (known as NPCs) are simulated by large language models (LLMs) (OpenAI, 2023; Jiang et al., 2023; Dubey et al., 2024), and the storytelling is collaboratively generated through dialogues between users and LLM agents.

The incorporation of LLMs have significantly enhanced the realism and adaptability of the interactive experiences. Despite growing interest in LLM-based interactive drama (Laurel, 1993; Wu et al., 2024; Han et al., 2024), a lack of accessible, open-source tools persists, hindering researchers from easily creating, modifying, and experimenting with such interactive experiences.[1]

To bridge this gap, we introduce **Open-Theatre**, an open-source toolkit designed for creating and experiencing configurable LLM-based interactive drama.

Current interactive drama frameworks vary in their focus, with some prioritizing user freedom while others emphasize stronger narrative coherence. Open-Theatre integrates these different architectures, such as one-for-all, director-actor, and hybrid structures, allowing users to flexibly choose according to their narrative preferences. A key challenge, however, remains in balancing freedom and coherence, since excessive user influence may disrupt plot progression. To address this, Open-Theatre introduces the **Director-Global Actor** framework, an extension of the traditional Director-Actor setup, where a global actor agent enhances character decision-making and narrative coherence. Furthermore, Open-Theatre enhances long-term coherence through a novel **hierarchical memory management system**, enabling intelligent information recall and prioritization.

Through experimental evaluation across various scripts and architectures, this paper demonstrates how Open-Theatre effectively bridges users with the crafted narrative world.

We outline the novel features of Open-Theatre:
• It is the first open-source toolkit that provides a highly configurable platform for creating and interacting with LLM-based interactive drama, enabling easy scripting, modification, and experimentation with dynamic narratives.
• It integrates and advances multiple architectures, utilizing sophisticated prompts and demonstrating the efficacy of the Director-Global Actor frame-

[◇] The authors contributed equally. [†] Corresponding author. OPEN-THEATRE is available at https://github.com/johnnie193/Open-Theatre, with a demo video for setup.

[1] A newest interactive drama: https://github.com/gingasan/interactive-drama

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 453–460
November 4-9, 2025

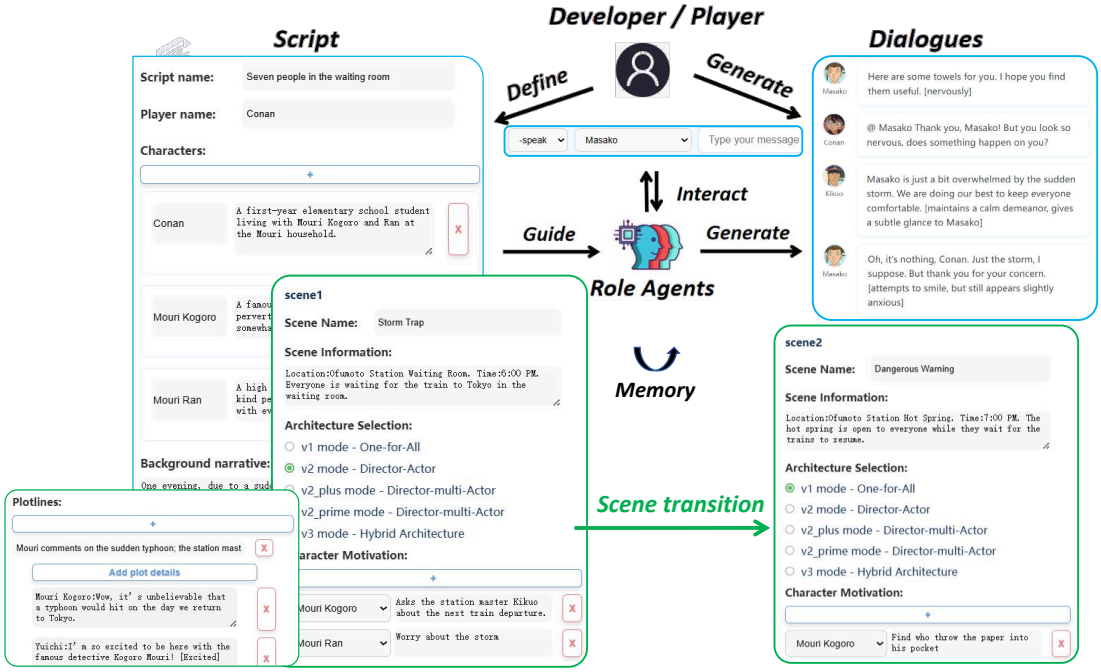


Figure 1: A demonstration of LLM-based interactive drama using the interface of the Open-Theatre. The script is adapted from the popular anime *Detective Conan*.

work.

- It features an innovative hierarchical memory management system for role agents, promoting human-like and consistent character responses.
- It allows secondary development for researchers with highly stable performance, enabling the integration of new architectures and customization of drama scripts and prompts in a lightweight manner.

2 Related work

Interactive Drama Computer-driven interactive drama has a long history of exploration (Laurel, 1993). More recently, Wu et al. (2024) introduces the concept of LLM-based interactive drama, outlining its six key components and proposing a fine-tuning strategy for a global drama LLM. Building on this line of research, Wu et al. (2025a) propose a plot-based reflection mechanism that periodically reviews the evolving storyline and adjusts narrative elements to facilitate player-centered story curves.

Simulate Dramatic Characters by LLM Agents LLM-based role-playing has seen significant advancements in generating dynamic and consistent character behavior (Chen et al., 2024, 2023; Liu et al., 2024; Zhou et al., 2023; Wu et al., 2025b). Memory has been recognized as a core component of LLM-based agents, fundamentally shaping their ability to maintain coherence and adapt to long-term interactions. While prior work (Park et al., 2023) has explored character memory over interactions, these systems often lack dynamic, context-aware memory management crucial for evolving dramatic narratives. General memory systems such as Mem0 (Chhikara et al., 2025) and MemBank (Zhong et al., 2023) offer structured storage but struggle with customization based on plot progression. Open-Theatre addresses these limitations by incorporating a novel hierarchical memory system.

Multi-Agent Framework Multi-agent systems (Tao et al., 2024; Islam et al., 2024) such as BabyAGI (Nakajima, 2023), AutoGen (Wu et al., 2023) and Camel-AI (Li et al., 2023), aim to achieve complex and goal-oriented behavior. In the context of interactive narratives, multi-agent frameworks such as director-actor (Han et al., 2024; Wu et al., 2025a) and ego-superego (Magee et al., 2024) coordinate agents to enable natural language interactions and strategic decision-making. However, maintaining narrative coherence in open-ended interactions remains a key challenge, as character autonomy can lead to inconsistent or fragmented storylines. Our Open-Theatre consolidates and extends these existing architectures through the introduction of a Director–Global Actor framework.

3 LLM-based Interactive Drama

We introduce the background of *LLM-based Interactive Drama*. It is a novel dialogue scenario where players experience an unfolding dramatic story by having dialogues with LLM-based characters.

As illustrated in Figure 1, a **drama script** guides the entire dialogues, which is in an episodic structure of scenes. Each scene details its background, character motivations, and plotlines, dictating character behaviours and plot objectives. For instance, Scene 1 (*Storm Trap*) depicts a scene where characters are trapped in the station waiting room due to a sudden typhoon. Completing Scene 1's plotlines transitions to Scene 2 (*Dangerous Warning*).

The drama script forms the foundation, guiding **role agents** (LLM agents simulating characters) to follow a specific architecture to collaborate together. Players could interact with any character in the ongoing scene, and characters would respond based on their motivations and plotlines defined in the script, adapting to player influence. For instance, in Figure 1, a player (as *Conan*) interacts with *Masako*, who, despite being scripted to distribute towels, adaptively responds when questioned about her nervousness, balancing plotline adherence with player-driven interaction.

4 Open-Theatre

This section introduces our OPEN-THEATRE toolkit designed for a highly configurable and easy-to-use LLM-based interactive drama. The core components of the Open-Theatre interface includes developer console, player console, and monitor.

4.1 Developer Console

The Developer Console serves as the backbone of Open-Theatre, providing creators with tools for pre-configuring narratives and refining storytelling during interaction.

Script Management This module enables comprehensive customization of the drama script, from foundational background settings to individual scene design. The **Background** module defines the narrative context, including setting, themes, and detailed character profiles. **Scenes** are broken down into discrete components that shape story progression: **Scene Information** defines location, time, and background; **Character Motivation** outlines scene-specific roles and motivations, ensuring agent alignment with narrative demands; **Plotlines** organize the scene story into manageable segments, ensuring logical progression and coherence and allowing predefinition of dialogues or events. The system supports real-time modifications to the script, enhancing dynamic narrative control.

Prompt Management While default prompts are pre-designed for general use, users can refine them to suit specific narrative needs or subsequent development.

Save and Load Management This function enables persistent storage of the current script state and chat history, allowing revisiting specific narrative points and ensuring continuity across sessions. For researchers, it provides robust support for debugging, scenario testing, interaction replication, and iterative storytelling and replayability.

4.2 Player Console

The Player Console serves as the primary interface for players to engage in an interactive drama, enabling them to execute specific commands, trigger scene transitions, and influence the story's direction. Players, as characters, can select other characters for **interaction** or choose **autonomous drama progression**. Advanced features include **withdrawing** to experiment with responses and **navigating directly to the next or previous scene**. All character dialogues, user inputs, and system feedback are presented in real-time, offering greater flexibility and dynamic story exploration.

4.3 Monitor

The Monitor provides players and developers with essential real-time contextual details of the Open-Theatre system's processes, facilitating understanding and guiding adjustments to the script and prompts. It comprises several key components:

- **Current Script:** Offers a real-time view of the ongoing script's state and progression, allowing users to track drama development.
- **Characters:** Enables users to explore details about characters in the current scene, including their profile, memory, motivation and agent reaction if applicable.
- **System Feedback:** Displays the system's generated responses and real-time prompts, offering transparency into underlying processes and insights into system decisions for refinement.
- **Record:** Captures the complete state of the drama, including all major events, actions, and interactions throughout the narrative.

4.4 Architectural Integration and Innovation

The Open-Theatre toolkit enhances flexible narrative experiences by integrating existing architectures and introducing its own novel designs.

One-for-All This architecture uses a single global agent to control all characters and the progression of drama. It is computationally efficient and suitable for scenes that prioritize narrative flow over player agency, such as expository or background-setting scenes.

Director-Actor In this architecture, each character is represented by an independent actor agent, while a higher-level director agent coordinates and guides the narrative. It allows for complex decision-making, making it ideal for scenes requiring interactive depth. However, it often incurs significant computational cost and latency due to requiring individual LLM requests for each actor's decision-making and profile adherence.

Hybrid Architecture This architecture switches between the director-actor and one-for-all architectures based on the scene's characteristics, ensuring a balance between narrative immersion and interactive depth.

Director-Global Actor We extend the traditional Director-Actor architecture, employing a single, centralized global actor agent that makes decisions for all characters. This global actor agent serves as a high-level consciousness for the entire dramatic ensemble. Guided by the director agent, it synthesizes comprehensive, real-time knowledge of all characters' profiles, memories, and states, providing a unifying context individual actor agents would otherwise lack. This capability not only ensures more strategically aligned and cohesive character actions, preventing contradictions and maintaining consistent character arcs, but also drastically improves efficiency, reducing redundant LLM API calls and latency for individual actor agents.

5 Hierarchical Memory System

Open-Theatre introduces a novel hierarchical memory architecture to provide narrative agents with a persistent, contextually-aware memory for extended, interactive narratives. While previous works (Park et al., 2023) emphasized recency, importance, and relevance in memory recall, our model extends this by introducing a partitioned memory space and a dynamic scoring model tailored for dramatic agency. This adaptive memory lifecycle ensures computational tractability and psychological plausibility throughout extended interactions.

5.1 Architectural Design

The architecture comprises four specialized memory stores, structured to model the varying cognitive accessibility of different information types within a partitioned memory space. The fundamental unit is a memory entry, a natural language description with metadata including semantic type, scene identifier, and a dynamically updated importance score.

These stores include: a **Global Store** for foundational, static knowledge like core identity attributes; an **Event Store** capturing the chronological stream of episodic experiences including dialogues and actions; a **Summary Store** housing condensed, abstractive summaries of events produced by memory consolidation; and an **Archive Store** serving as a long-term repository for consolidated, less frequently accessed records, preserving historical context for potential future retrieval.

5.2 Retriever

To manage the cognitive load of long dialogues, a retriever module computes relevant information from the memory stores in response to new dialogue turns. For a given new dialogue query q, the final retrieval score for each candidate memory entry c is computed based on three factors: relevance, importance, and narrative recency, defined as:

$$S_{final}(c) = P_{recency} \cdot (S_{relevance} + S_{importance}) \quad (1)$$

5.2.1 Base score

The base retrieval score is an additive combination of relevance and importance. The **Relevance Score ($S_{relevance}$)** is computed using a standard hybrid method, fusing lexical signals from BM25Okapi with semantic similarity from FAISS-indexed sentence embeddings. For sentence embeddings, we utilize the all-MiniLM-L6-v2 model. The system processes the narrative at the character utterance level; each dialogue turn is treated as a single memory entry. These entries are aggregated into larger chunks for efficient storage and retrieval. Chunking is performed with a configurable size (`chunk_max_pieces = 5`) and an

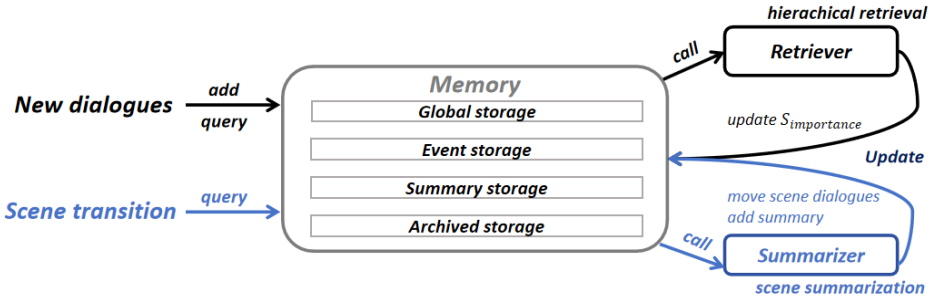


Figure 2: **Hierarchical Memory Architecture.** New dialogues are ingested into the Event Store and call retriever to query all stores.The scene transition signal calls the summarizer module to create new entries in the Summary Store and move original records to the Archive Store, completing the memory lifecycle.

overlap (`chunk_overlap_pieces` = 1), allowing the system to preserve local coherence while reducing fragmentation. The **Importance Score** ($S_{importance}$) is an evolving score unique to each memory entry. It increases with each retrieval event, allowing the system to learn a memory's prominence from its usage history. The relative weighting between relevance and importance is determined by the `ADDITION_WEIGHT` (default set to 0.05), ensuring that the retrieval is primarily driven by immediate semantic and lexical relevance, while the learned importance serves as a subtle long-term signal to capture memory salience. This design ensures frequently recalled memories can influence agent behavior more, consistent with cognitive theories of memory salience.

5.2.2 Recency Punishment

The base score is modulated by a dynamic factor ($P_{\text{recency}} \in (0, 1]$) that penalizes temporally distant memories. Our model implements a two-tiered decay logic to differentiate narrative recency types:

Inter-Scene Penalty To model the decay of relevance *across* contextual shifts, for memory entries from a different scene ($c_{\text{scene}} \neq S_{\text{current}}$), a penalty based on scene distance is applied:

$$P_{\text{recency}} = (1 + \alpha \cdot |S_{\text{current}} - c_{\text{scene}}|)^{-1} \quad (2)$$

where α is a decay constant, empirically tuned to control the rate of forgetting across scene transitions, set to 0.25 as default.

Intra-Scene Dialogue Penalty For records *within* the current scene, a penalty based on turn order captures the immediate conversational context:

$$P_{\text{recency}} = (1 + \beta \cdot \text{TurnsAgo})^{-1} \quad (3)$$

where β is a per-turn decay constant, and *TurnsAgo* represents number of dialogue turns since creation. We set β to 0.005, ensuring that recent turns are slightly favored but older relevant memories within the scene are not entirely excluded. For other memory types within the current scene, P_{recency} is 1.0, incurring no penalty.

5.3 Summarizer

To manage the cognitive load inherent in long narratives, the system implements a summarizer module for memory consolidation. It processes records from the Event Store for completed scenes, leveraging an LLM to generate abstractive summaries that are added to the Summary Store. This process enriches the agent's memory base with high-level, semantic knowledge from raw episodic experiences, concurrently reducing the pressure to retain all original records, which are then moved to the Archive Store for long-term preservation.

6 Evaluation

We conduct a rigorous evaluation to assess the performance of distinct agent architectures with and without memory system. Our analysis is performed across three diverse scripts to ensure robust, genre-agnostic findings: a detective story (adapted from *Detective Conan*), an adventure tale (adapted from *Harry Potter*), and a classical drama (adapted from *Romeo and Juliet*).

6.1 Experimental Setup

To ensure a comprehensive and objective assessment, we employed an automated evaluation pipeline. We utilized a suite of AI agents to act as players, designed with 10 distinct personas (ranging from cooperative to aggressive) to test system robustness and adaptability of each system configuration against a wide spectrum of behaviors. Following each simulated playthrough, we use an

Table 1: A multi-dimensional assessment of different architectures with and without memory configurations. Qualitative scores are averaged across three diverse scripts and rated on a 1-5 scale by our AI Judge.

ARCHITECTURE	CONFIG	MEMORY			ARCHITECTURE			EFFICIENCY	
		RA	**RP**	**NC**	**CC**	**MAC**	**PA**	**Latency (s)**	**LLM Calls**
One-for-All	w/o RAG	/	3.8	3.8	3.5	4.4	4.3	**12.1**	**1.0**
	w/ RAG	4.6	4.4	4.2	3.4	4.4	**4.6**	16.8	1.0
Director-Actor	w/o mem	/	4.0	3.7	**4.6**	3.3	3.7	34.5	3.4
	w/ mem	4.5	**4.7**	4.0	**4.6**	3.4	4.0	40.6	3.4
Director-Global Actor	w/o mem	/	3.9	4.0	4.1	**4.5**	4.0	20.5	2.0
	w/ mem	4.5	4.4	**4.6**	4.3	**4.5**	4.3	25.0	2.0

impartial Judge Agent (powered by GPT-4o) to score the complete interaction log, which provides scalable, consistent, and reproducible ratings based on defined metrics, effectively minimizing human bias and effort.

6.2 Evaluation Metrics

6.2.1 Memory System Performance

We evaluate our hierarchical memory system through three complementary dimensions:

• **Retrieval Accuracy (RA):** Success rate of retrieving relevant historical information.

• **Response Plausibility (RP):** Degree to which memory-based responses align well with the character and the current dialogue context.

• **Narrative Coherence (NC):** Long-term story consistency through memory integration.

6.2.2 Architectural Performance

We assess the interactive experience under different architectures through:

• **Character Consistency (CC):** Maintenance of distinct character personalities.

• **Multi-Agent Coordination (MAC):** Quality of inter-character strategic alignment.

• **Plot Adherence (PA):** Effectiveness in narrative progression and engagement.

6.2.3 System Efficiency

We measure computational performance through:

• **Response Latency:** Average response time per interaction turn (seconds/turn).

• **Computational Cost:** Number of LLM API calls required per turn.

6.3 Results

Table 1 presents our evaluation results, comparing different architectures with and without our hierarchical memory system.

6.3.1 Memory System Validation

Across all architectures, integrating the hierarchical memory system yields consistent improvements in response quality and long-term coherence. Specifically, **Retrieval Accuracy (RA)** exceeds **4.5** out of 5, confirming the system's ability to retrieve relevant past information reliably. When compared to the baseline without memory system, **Response Plausibility** saw a substantial improvement (e.g., from **3.8** to **4.4** in One-for-all), indicating that memory-based responses are much better aligned with character and current dialogue context. These gains translate into higher **Narrative Coherence** (e.g., up to **4.6** in Director-Global Actor), showing improved consistency over long interactions. Even in simpler settings like One-for-All, memory boosts NC from **3.8** to **4.2**, further supporting the general utility of our memory system.

6.3.2 Architecture Performance

In terms of architectural design, the proposed **Director-Global Actor** architecture offers a strong balance between multi-agent coordination and narrative coherence, while maintaining competitive efficiency. Unlike Director-Actor, which excels at **Character Consistency (CC=4.6)** but suffers from lower **Multi-Agent Coordination (MAC=3.3)** and **Plot Adherence (PA=3.7)**, the Director-Global Actor achieves strong performance across almost all metrics when paired with our memory system. It scores highest on **NC (4.6)** and maintains high values for **MAC (4.5)** and **PA (4.3)**, suggesting that centralized reasoning leads to better plot alignment and action coordination among characters.

6.3.3 System Efficiency

In terms of efficiency, the One-for-All architecture requires only 1.0 LLM call and achieves the lowest latency. However, this simplicity comes at the cost of limited multi-character reasoning, resulting in lower coordination and consistency scores. By contrast, while the Director-Actor architecture incurs higher overhead, requiring 3.4 LLM calls per turn due to decentralized decision-making, the Director-Global Actor strikes a more favorable balance. It reduces the number of calls to 2.0 while maintaining high-quality outputs, confirming its computational viability for complex narrative generation. The competitive latency offers a favorable trade-off between performance and runtime cost.

7 Conclusion

This paper proposed Open-Theatre, a toolkit designed to facilitate the creation and interaction for LLM-based interactive drama. It assembles and advances different architectures, notably proposing a Director-Global Actor architecture and integrating a novel hierarchical memory system. Our experimental evaluation demonstrates that Open-Theatre provides a flexible and configurable platform for users to experience in a lightweight manner and facilitates secondary development. We hope this toolkit contributes to the broader field of interactive storytelling and inspires future research in AI-assisted narrative development.

8 Limitation

While an AI judge offers a highly scalable and standardized method for evaluating user experience, it might not fully capture the nuanced and subjective elements of human perception. Certain subtleties that only direct human feedback can reveal could be overlooked.

Also, our validation relies mostly on GPT-4o (accessed on August 6, 2024), limiting insights into framework generalizability across diverse LLMs. Future work will prioritize hybrid evaluation strategies and cross-model benchmarking to address these gaps.

References

Hongzhan Chen, Hehong Chen, Ming Yan, Wenshen Xu, Xing Gao, Weizhou Shen, Xiaojun Quan, Chenliang Li, Ji Zhang, Fei Huang, and Jingren Zhou. 2024. Socialbench: Sociality evaluation of role-playing conversational agents. *Preprint*, arXiv:2403.13679.

Nuo Chen, Yan Wang, Haiyun Jiang, Deng Cai, Yuhan Li, Ziyang Chen, Longyue Wang, and Jia Li. 2023. Large language models meet harry potter: A dataset for aligning dialogue agents with characters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8506–8520, Singapore. Association for Computational Linguistics.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Steven Dow, Manish Mehta, Ellie Harmon, Blair MacIntyre, and Michael Mateas. 2007. Presence and engagement in an interactive drama. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 1475–1484, New York, NY, USA. Association for Computing Machinery.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Senyu Han, Lu Chen, Li-Min Lin, Zhengshan Xu, and Kai Yu. 2024. IBSEN: director-actor agent collaboration for controllable and interactive drama script generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand,*

August 11-16, 2024, pages 1607–1619. Association for Computational Linguistics.

Md. Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. Mapcoder: Multi-agent code generation for competitive problem solving. *Preprint*, arXiv:2405.11403.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Brenda Laurel. 1993. *Computers as theatre*. Addison-Wesley.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. *Preprint*, arXiv:2303.17760.

Jiaheng Liu, Zehao Ni, Haoran Que, Tao Sun, Zekun Wang, Jian Yang, Jiakai Wang, Hongcheng Guo, Zhongyuan Peng, Ge Zhang, Jiayi Tian, Xingyuan Bu, Ke Xu, Wenge Rong, Junran Peng, and Zhaoxiang Zhang. 2024. Roleagent: Building, interacting, and benchmarking high-quality role-playing agents from scripts. In *Advances in Neural Information Processing Systems*, volume 37, pages 49403–49428. Curran Associates, Inc.

Liam Magee, Vanicka Arora, Gus Gollings, and Norma Lam-Saw. 2024. The drama machine: Simulating character development with llm agents. *Preprint*, arXiv:2408.01725.

Yohei Nakajima. 2023. Babyagi. *GitHub repository*.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

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. *Preprint*, arXiv:2304.03442.

Marie-Laure Ryan. 1997. Interactive drama: Narrativity in a highly interactive environment. *MFS Modern Fiction Studies*, 43(3):677–707.

Wei Tao, Yucheng Zhou, Yanlin Wang, Wenqiang Zhang, Hongyu Zhang, and Yu Cheng. 2024. Magis: Llm-based multi-agent framework for github issue resolution. *Advances in Neural Information Processing Systems*, 37:51963–51993.

Hongqiu Wu, Weiqi Wu, Tianyang Xu, Jiameng Zhang, and Hai Zhao. 2025a. Towards enhanced immersion and agency for llm-based interactive drama. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 11166–11182. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *Preprint*, arXiv:2308.08155.

Weiqi Wu, Hongqiu Wu, Lai Jiang, Xingyuan Liu, Hai Zhao, and Min Zhang. 2024. From role-play to drama-interaction: An LLM solution. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3271–3290. Association for Computational Linguistics.

Weiqi Wu, Hongqiu Wu, and Hai Zhao. 2025b. X-TURING: towards an enhanced and efficient turing test for long-term dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 5874–5889. Association for Computational Linguistics.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory. *Preprint*, arXiv:2305.10250.

Jinfeng Zhou, Zhuang Chen, Dazhen Wan, Bosi Wen, Yi Song, Jifan Yu, Yongkang Huang, Libiao Peng, Jiaming Yang, Xiyao Xiao, Sahand Sabour, Xiaohan Zhang, Wenjing Hou, Yijia Zhang, Yuxiao Dong, Jie Tang, and Minlie Huang. 2023. Characterglm: Customizing chinese conversational ai characters with large language models. *ArXiv*, abs/2311.16832.

CafGa: Customizing Feature Attributions to Explain Language Models

Alan Boyle **Furui Cheng*** **Vilém Zouhar** **Mennatallah El-Assady**
ETH Zurich

Abstract

Feature attribution methods, such as SHAP and LIME, explain machine learning model predictions by quantifying the influence of each input component. When applying feature attributions to explain language models, a basic question is defining the interpretable components. Traditional feature attribution methods, commonly treat individual words as atomic units. This is highly computationally inefficient for long-form text and fails to capture semantic information that spans multiple words. To address this, we present CafGa, an interactive tool for generating and evaluating feature attribution explanations at customizable granularities. CafGa supports customized segmentation with user interaction and visualizes the deletion and insertion curves for explanation assessments. Through a user study involving participants of various expertise, we confirm CafGa's usefulness, particularly among LLM practitioners. Explanations created using CafGa were also perceived as more useful compared to those generated by two fully automatic baseline methods: PartitionSHAP and MExGen, suggesting the effectiveness of the system.

1 Introduction

Large Language Models (LLMs) continuously evolve in scale and capabilities across a wide range of tasks, such as question answering, reasoning, and text summarization (Kamalloo et al., 2023, inter alia). Meanwhile, their increasing complexity poses significant challenges in interpretability and human trust (Sun et al., 2024). Feature attribution methods, e.g. SHAP (Lundberg and Lee, 2017) and LIME (Ribeiro et al., 2016), offer a promising approach to explain LLM behaviors by quantifying the influences of each component in the input. These explanations calibrate users' trust in model predictions (Bansal et al., 2021) and offer useful insights into the prediction shortcuts and model biases (Du et al., 2021; Ren and Xiong, 2023). Defining interpretable components is a fundamental question in generating meaningful feature attributions (Ribeiro et al., 2016). Traditional removal-based methods typically treat individual words as the basic units of interpretation (Figure 1), but this approach struggles to scale effectively for long-form text due to computational inefficiency and the fact that meaningful semantic cues often span multiple tokens or phrases, such as *"made it worth it"* in Figure 1. To address this, we propose CafGa, an interactive tool that enables users to create feature attribution explanations at customized levels of granularity. The tool offers default options to segment text at the word, sentence, or paragraph level, and supports users to further refine these segments by interactively isolating text spans. CafGa also provides functionality to evaluate and compare different segmentation strategies. The system visualizes deletion and insertion curves, showing how the model's prediction changes as components are removed or added based on their attribution rankings, and calculates the area under the perturbation curve as the fidelity scores.

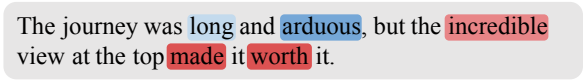


Figure 1: Feature attribution explanations quantify the contributions of each component (e.g., words), allowing users to validate the models' reasoning.

We conducted a two-stage user study to evaluate the usability and effectiveness of CafGa. In the first phase, ten participants used CafGa to create explanations that helped them understand the model's reasoning. Based on self-reported usability scores, users with machine learning expertise generally found the system easy to use and learn,

*Corresponding author

†CafGa is available as a pip-installable package cafga. The source code is hosted on https://github.com/explain-llm/CafGa, and a live demo is accessible at cafga.ivia.ch.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 461–470
November 4-9, 2025

while novice users experienced some difficulty due to the need to grasp new concepts before effectively interacting with the tool. Despite this, participants across all expertise levels reported that CafGa was helpful in improving their understanding of the LLM. In the second phase, we invited four expert users to compare the usefulness of the CafGa-generated explanations from the first phase with those produced by two automated baselines: PartitionSHAP (Lundberg, 2024) and MExGen (Paes et al., 2024). In 64% of the comparisons, participants rated the human-generated explanations as the most helpful, demonstrating the effectiveness of the proposed system in supporting meaningful model interpretation.

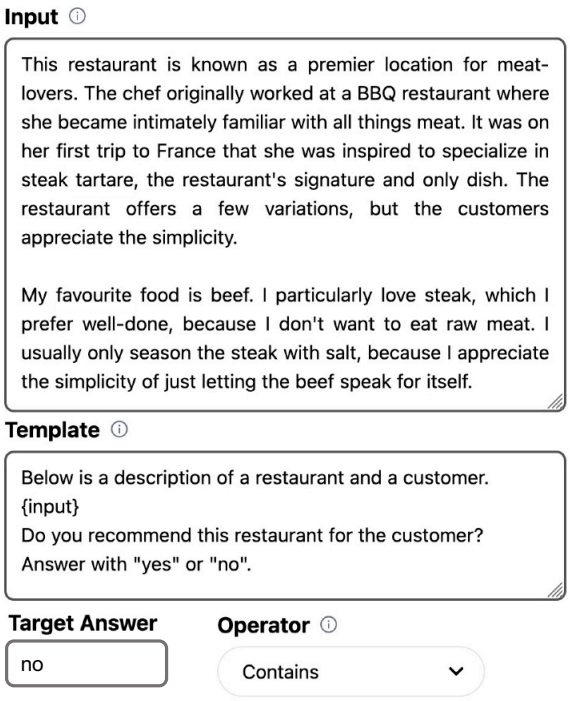


Figure 2: The user creates a task where a model is asked to decide whether to recommend a restaurant to a customer.

2 System Design

CafGa is an interactive system that includes a *task creation page* for defining prediction tasks and customizing explanation granularities, and an *explanation page* that displays explanations and their evaluation results. The following sections introduce the key features and design.

Define a Prediction Task. CafGa allows users to define a prediction task by constructing a prompt and defining an evaluator (Figure 2). To construct the prompt, users need to complete the `template` field with the structure of the prompt and the `input` field with the content they wish to explain. We use the design in ChainForge (Arawjo et al., 2024) where an evaluator is used to structure the text generated by the model by transforming the text into a boolean value based on user-defined rules. To define the evaluator, users can choose from predefined operators such as `Contains`, which returns a boolean value indicating whether the text includes the specified `Target Answer`, and `Entails`, which assesses whether the text logically entails the `Target Answer`. See the full list of operators in Appendix B.

Customize Text Segmentation. CafGa supports default options for creating text segments, such as dividing text by word, sentence, or paragraph. Users can customize the segmentation through interactions. For example, they could brush to select a phrase and isolate the selected phrase from the sentence (Figure 4). The segments may not overlap to avoid ambiguous attributions.

Calculate and View Feature Attributions. CafGa uses the KernelSHAP (Lundberg and Lee, 2017) algorithm to sample and compute the feature attributions. The system generates random samples by removing text segments in the user-specified granularities. For each perturbed sample, the system requests ten responses, which are converted into Boolean values using the user-defined evaluator. The estimated probability, e.g., `P(the answer contains “no”)` is computed as the proportion of responses evaluated as true. Finally, the system uses a weighted linear regression model to estimate the Shapley values of each group. This step runs in the back-end and is hidden from users. After the calculation, we visualize the explanations using a heatmap with a color-blind friendly color schema (Figure 3B) accompanied by the task description (Figure 3A).

Evaluate Explanations. To ensure that the created explanations accurately reflect the model's decision making and mitigate users' confirmation bias, we adapt a commonly used and general fidelity metric, Area Over the Perturbation Curve (Samek et al., 2017). Following Petsiuk et al. (2018), we employ two variants of the perturbation curve: *Deletion* and *Insertion*. Deletion begins with all the features present and then iteratively removes the highest valued features. Insertion begins

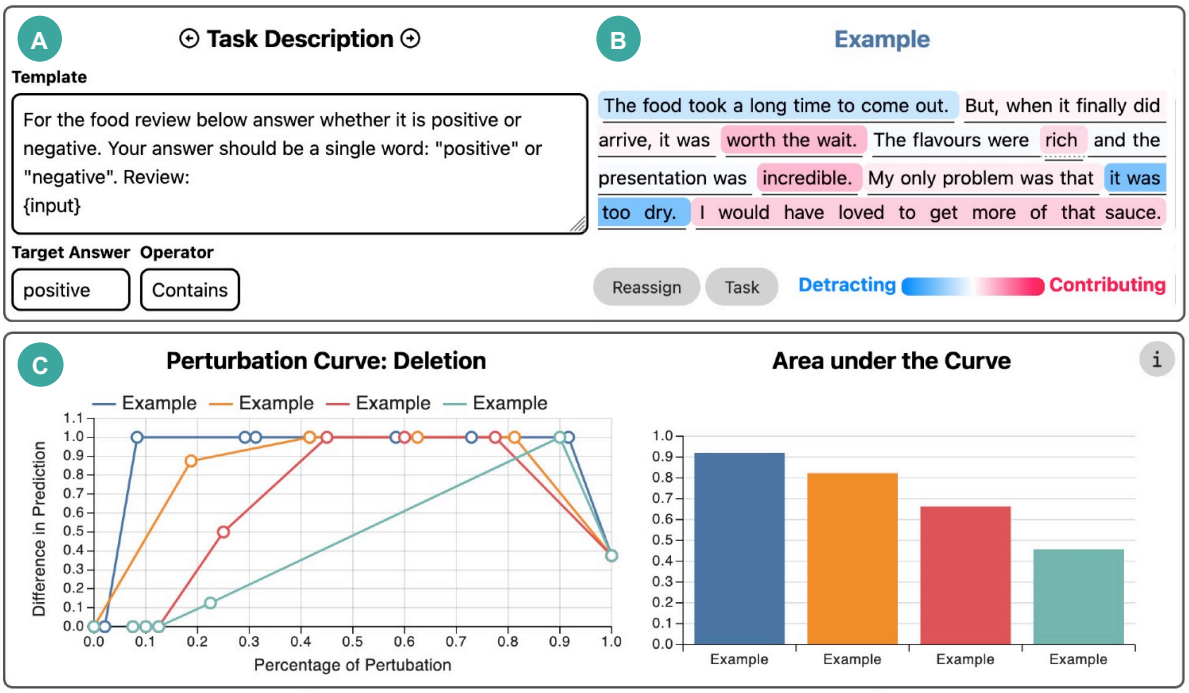


Figure 3: An overview of CafGa: The user first creates a predictive task (A) and then decides the granularities for explanations by assigning words into groups and gets an explanation based on the assignments (B). Using the perturbation curve the user can validate the fidelity of the generated explanation (C).

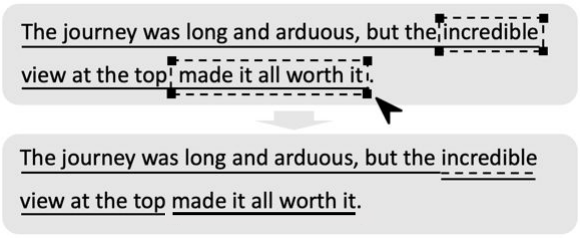


Figure 4: CafGa allows users to interactively segment text to customize the explanation granularity.

with no features present and iteratively inserts the highest valued features. In the resulting graph the x-axis represents the percentage of words added or removed and the y-axis the difference between the original prediction $f(x)$ and the perturbed prediction $f(x^k)$ when perturbing the top k features. Importantly, we plot the percentage of *words* perturbed on the x-axis rather than the percentage of *groups* to avoid a bias towards large groupings. For deletion the area under the curve should be large as the perturbed prediction should quickly diverge from the original one. For insertion it should be small as the perturbed prediction should quickly converge back to the original one. These metrics are visualized in an intuitive way to users such that they can follow and compare the computation exactly (Figure 3C).

Implementation Details. CafGa is implemented as a web application composed of a front-end and a back-end communicating via HTTP requests. The front-end is written in TypeScript and React and served via Vite. The back-end server uses FastAPI and Uvicorn. The model used in the back-end can be chosen by the user from a selection of OpenAI models. In the implementation of the KernelSHAP algorithm, we use the sampling strategy and functions from the SHAP library. This implementation uses $2 * n_{features} + 2048$ as the maximum number of samples used to approximate the shapley value. In the interactive setting we prioritize user experience and thus set the number of samples so that the explanation can be provided in a reasonable time t_{max}. Given a rate of API requests r_{API}, the maximum number of samples is defined as $n_{samples} = t_{max} \cdot r_{API}$. We provide a PyPI package that contains all of the algorithms used in the back-end. The package also comes with Jupyter widgets created using AnyWidget (Manz et al., 2024) that recreate parts of the front-end. A Jupyter Notebook example is shown in Appendix A. This in particular allows users to also run CafGa on their local models.

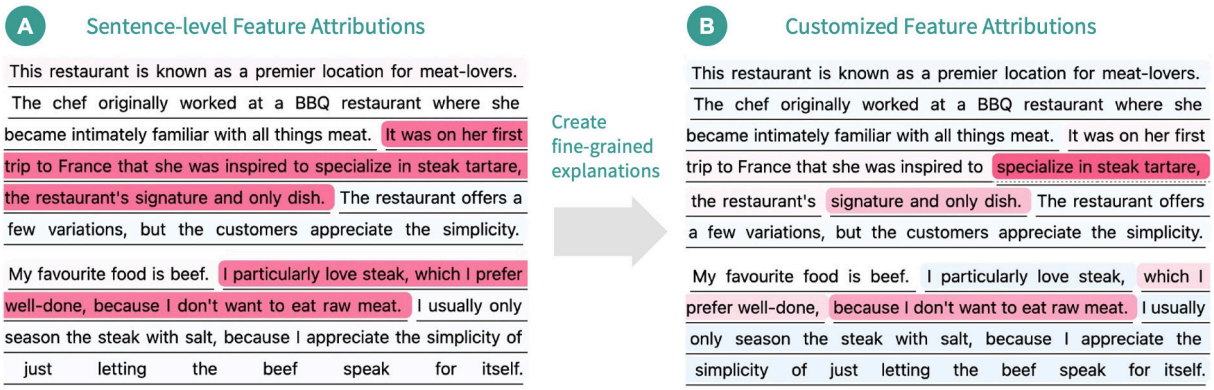


Figure 5: The user first used the sentence preset (A). The resulting explanation gives a rough overview of the model's reasoning. The user specifically assigned all relevant parts into separate groups (B). In (B), one can now clearly see the reason: tartare is the restaurant's sole signature dish, but the customer does not want to eat raw meat.

3 A Worked-out Example

We introduce a use case in understanding GPT4o-mini's predictions in multi-hop reasoning. In this task, the model needs to decide whether to recommend the restaurant to a customer based on the restaurant's review and the customer's preferences. The model makes a correct prediction that does not recommend the restaurant. The user, an LLM developer, wanted to understand if the model followed the correct reasoning behind this decision.

The user first defined the task (see Figure 2) and used a preset that calculates sentence-level attributions. From the results (see Figure 5 A), the user noticed that the two sentences with the highest attribution include the necessary information for making this decision – *the restaurant only serves steak tartare*, while *the customer doesn't like raw meat* (see highlighted parts in Figure 5 B). The user found this explanation unsatisfactory and wanted to know if the model specifically captured the key information that *steak tartare* is *the only dish served* in these sentences.

So the user then manually separated the sentence into segments, like *specialize in steak tartare*, *signature and only dish*, to get a more precise explanation. The new explanation, with a higher fidelity (from 0.77 to 0.96) suggests that all the necessary information is well-captured by the model and has high contributions to the prediction. From this two-stage exploration, the user confirmed that the model follows the correct reasoning in this decision.

4 User Study

To evaluate the usability of CafGa, we conducted a two-stage user study in which participants created and assessed explanations. In the first stage, participants used CafGa to generate customized explanations and then provided feedback on the tool's usability. In the second stage, experts compared these participant-generated explanations with those produced by existing fully automatic methods to assess the quality of the generated explanations.

4.1 Setup

We now describe the two parts of the user study.

Creating explanations. For the first stage we recruited ten participants: six users who had experience working with machine learning, noted as experts and four novice users. None of the participants had any prior experience working with attribution-based explanations.

At the beginning of the study, the participants were shown a video that explained the concepts behind CafGa and their role in this part of the study. The video also showed an example task that the participants could work through while following the video. After viewing the video participants were encouraged to ask questions. The participants then created explanations for five types of tasks of increasing complexity. The tasks were local question answering taken from SQuAD (Rajpurkar et al., 2016), sentiment analysis on reviews taken from the YELP academic dataset (YELP, 2015), a few-shot prompt engineering task inspired by Tenney et al. (2024), multi-hop reasoning taken from HotpotQA (Yang et al., 2018) and long-form text comprehension taken from BARQA (Hou, 2020) (see Appendix C for details). For each task type there were five tasks except for prompt-engineering and long-form text comprehension, which each had three tasks. The given task for each type was se-

lected at random. Once participants had created an explanation for each type of task, they filled out a survey rating the usability and usefulness of CafGa.

Comparing explanations. In the second stage, we recruited four expert participants who all studied machine learning and currently work with AI. The participants were again shown a video that explained the concepts behind CafGa and their role in this part of the study. The participants were asked to choose the most helpful explanations among 10 groups. Each group contained three feature attributions: a human-created explanation from the first stage of the study, one created by MExGen (Paes et al., 2024), and one created by PartitionSHAP (Lundberg, 2024). We choose MExGen because it provides a good baseline of what static granularities can achieve and PartitionSHAP because it is often cited as a comparison (Amara et al., 2024; Paes et al., 2024; Mosca et al., 2022) and is recommended as an efficient method by Mosca et al. (2022). Unlike most methods they also scale to long inputs like those in BARQA. Participants were asked to select the most helpful explanation – the one that best informed them of the model's decision logic and correctness – among the three.

4.2 Study Results

From the usability survey results (Table 1), we observed that the experts could use the system well. They generally found the system easy to learn and easy to use. In comparison, non-expert users faced more challenges in learning the system, primarily due to difficulties in quickly understanding the underlying concepts of CafGa. However, after getting familiar with the system, both groups of participants reported that the system is helpful in supporting them in understanding the LLM.

From the comparative study results (Table 2), we find that the explanations made by humans were generally preferred, with MExGen outperforming PartitionSHAP. We see that in total, the explanations created by humans were preferred 64% of the time over MExGen and PartitionSHAP. This demonstrates that CafGa is effective in supporting meaningful model interpretation.

Task-level preferences (Table 2) show that human-customized explanations performed especially well on the HotpotQA task. This is likely because the input text in these tasks typically contains multiple facts, often expressed through distinct clauses and phrases. The participants can make semantic segmentations to the input text, while automatic methods create unreasonable n-grams that do not align with human intuitions.

Statement	Non-experts	Experts
The system is easy to use.	3.0	5.0
The system is easy to learn.	3.5	5.2
The system is enjoyable to use.	3.5	5.8
The various functions in the system are well integrated.	5.0	5.5
The system is helpful in understanding the LLM.	6.3	5.8
I would like to use this system frequently.	3.3	4.0

Table 1: After-task survey results. Participants were asked to rate the statements on a scale from 1 (strongly disagree) to 7 (strongly agree) with 4 (neutral). For each statement, we show the distribution and average.

	Human	MExGen	PartitionSHAP
SQuAD	63%	38%	0%
YELP	67%	33%	0%
Prompt	50%	0%	50%
HotpotQA	75%	25%	0%
BARQA	63%	38%	0%
Average	64%	28%	8%

Table 2: Preference win-rate for generated explanations across tasks.

5 Related Work

In this section we discuss related work on creating feature attributions at interpretable granularities and existing user interfaces.

5.1 Interpretable Feature Attributions

Recent work in attribution-based explanations has focused on creating attributions at granularities beyond the word level. This approach can not only improve computational efficiency, but also make for more interpretable explanations.

Early work in this direction focuses on finding specific decompositions of the neural network to get contextual attributions in addition to token-level attributions (Murdoch et al., 2018; Singh et al., 2019; Jin et al., 2019). However, given that there are now many settings where one does not have access to the model's internal structure, such model-specific methods may no longer be applicable. Thus, there has been a shift towards model-agnostic methods that do not rely on the model's

internal structure.

Chen et al. (2020) present a model-agnostic approach that iteratively splits the input sequence into smaller groups generating an attribution for each group. This hierarchical approach in particular allows users to validate whether the model can reason about the compositional nature of language.

Ju et al. (2023) further improve on this by addressing the limitation that the spans of grouped features must be contiguous. This is important because modern NLP architectures can model long-range non-contiguous dependencies in the input. However, both of these methods struggle to scale to large input sequences.

Paes et al. (2024) accordingly propose a method in which they first group the input at the coarsest granularity (e.g., paragraphs) and then iteratively refine the granularity only for the highest attributed group. This approach allows them to greatly reduce the computation time, but it can fail to precisely pinpoint the important parts of the input that are in the groups which do not get the highest attribution score. It also relies on the assumption that the fixed granularities used for the explanation are interpretable.

Cheng et al. (2025) emphasize the importance of the human understanding of language in creating explanations. They thus group the features in the input by use of a syntactic tree allowing them to make semantically meaningful groups.

Despite research into improving explanations by attributing features at differing granularities, little work has been done to directly involve humans in the creation of such explanations. We close this gap by empowering users to customize the attributions in a way that best suits their needs, while still informing them of the fidelity of the created explanations to stay aligned with the model.

5.2 LLM Explanation Interfaces

Recent research has focused on helping users gain a deeper understanding of LLMs through interactive visual interfaces.

Many of these interfaces target specific neural networks and explain the model by visualizing the architecture and the model's inner computations (Zeiler and Fergus, 2013; Ming et al., 2017; Strobelt et al., 2016, 2018).

However, as it may not always be possible to access the model's internal structure, there has also been a rise in model-agnostic LLM interfaces (Coscia and Endert, 2024; Cheng et al., 2024; Arawjo et al., 2024; Kahng et al., 2024). Many of these tools allow users to investigate *what* a model will answer, but they do not provide users with precise tools to explain *why* a model answers that way.

LLM Checkup (Wang et al., 2024) allows users to explore the model's reasoning by chatting with the model. This approach relies on model self-explanation, however, which can be misleading as the explanations often lack fidelity (Madsen et al., 2024; Turpin et al., 2023).

Closely related to our work, Cheng et al. (2025) analyze the LLM's predictions by perturbing the input with meaningful counterfactuals. They use a syntactic tree to define semantically meaningful segments in the input which can be ablated in a perturbation. This allows users to perform an analysis similar to attribution-based explanation, as they can test which of the defined segments are relevant for the model's prediction.

Tenney et al. (2024) also present an interface where users can choose the granularity of feature attribution. However, their tool relies on model-specific methods limiting its applicability and only allows users to specify the features at fixed granularities or by use of a regex, which means customizing explanations is largely not possible.

We further emphasize the need for users to enabled to fully customize the explanation rather than just choosing from a limited set of options. Our model-agnostic approach also makes our system more broadly applicable and the use of the fidelity metric allows users to make certain that their custom explanations have high fidelity.

6 Conclusion

We introduce CafGa, a tool that enables users to generate explanations at arbitrary granularities by grouping words in a customized manner. We argue that using CafGa users can create more interpretable explanations. CafGa also provides users with a fidelity metric to ensure that the created explanations still align with the model's decision making. Through a case study and a user study, we demonstrate CafGa's usability and effectiveness, finding that users successfully created explanations that were preferred over automatic methods. This highlights the value of human involvement in creating interpretable explanations.

Limitations

Our user study was limited by a small sample size due to the constrained number of available participants. In future work, we would like to conduct a larger-scale evaluation. In particular, we believe that evaluating the practical use of CafGa in real-world, non-laboratory settings would provide valuable insights and offer a more accurate assessment of its usability and effectiveness.

Approximating the Shapley value involves a trade-off between speed and accuracy. Using fewer groups increases computational efficiency but risks obscuring important information within those groups. Conversely, creating many word groups improves fidelity but substantially reduces speed. Providing users with appropriate guidance in balancing this trade-off and thereby determining a suitable number of groups remains an important direction for future investigation.

We use deletion and insertion as fidelity metrics which are the most common fidelity metrics for attribution-based explanations and can be intuitively visualized for users. However, these metrics introduce a degree of circularity as both the explanation and the evaluation are based on perturbation. Their applicability also becomes questionable when features represent groups rather than atomic units (i.e., words). To mitigate this, we plot the results with respect to atomic units rather than grouped features. Nonetheless, further research is needed to develop more reliable fidelity metrics.

Ethics Statement

The participants were sourced from a pool of academic labs, which work on reciprocal basis instead of monetary rewards.

References

Kenza Amara, Rita Sevastjanova, and Mennatallah El-Assady. 2024. SyntaxShap: Syntax-aware explainability method for text generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4551–4566. Association for Computational Linguistics.

Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. Chainforge: A visual toolkit for prompt engineering and LLM hypothesis testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–18. ACM.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593. Association for Computational Linguistics.

Furui Cheng, Vilém Zouhar, Robin Shing Moon Chan, Daniel Furst, Hendrik Strobelt, and Mennatallah El-Assady. 2025. Understanding large language model behaviors through interactive counterfactual generation and analysis. *IEEE Transactions on Visualization and Computer Graphics*.

Furui Cheng, Vilém Zouhar, Simran Arora, Mrinmaya Sachan, Hendrik Strobelt, and Mennatallah El-Assady. 2024. RELIC: Investigating large language model responses using self-consistency. In *Proceedings of the 2024 CHI conference on human factors in computing systems*.

Adam Coscia and Alex Endert. 2024. KnowledgeVIS: Interpreting language models by comparing fill-in-the-blank prompts. *IEEE Transactions on Visualization and Computer Graphics*, 30(9):6520–6532.

Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. 2021. Towards interpreting and mitigating shortcut learning behavior of NLU models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 915–929. Association for Computational Linguistics.

Yufang Hou. 2020. Bridging anaphora resolution as question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1428–1438. Association for Computational Linguistics.

Xisen Jin, Junyi Du, Zhongyu Wei, Xiangyang Xue, and Xiang Ren. 2019. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. *CoRR*, abs/1911.06194.

Yiming Ju, Yuanzhe Zhang, Kang Liu, and Jun Zhao. 2023. A hierarchical explanation generation method based on feature interaction detection. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12600–12611. Association for Computational Linguistics.

Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif,

Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. 2024. LLM comparator: Visual analytics for side-by-side evaluation of large language models. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606. Association for Computational Linguistics.

Scott M. Lundberg. 2024. Exact explainer — SHAP latest documentation.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Andreas Madsen, Sarath Chandar, and Siva Reddy. 2024. Are self-explanations from large language models faithful? In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 295–337. Association for Computational Linguistics.

Trevor Manz, Nezar Abdennur, and Nils Gehlenborg. 2024. anywidget: reusable widgets for interactive analysis and visualization in computational notebooks. *Journal of Open Source Software*, 9(102):6939. Publisher: The Open Journal.

Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. 2017. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 13–24.

Edoardo Mosca, Ferenc Szigeti, Stella Tragianni, Daniel Gallagher, and Georg Groh. 2022. SHAP-based explanation methods: A review for NLP interpretability. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4593–4603, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from LSTMs. In *International Conference on Learning Representations*.

Lucas Monteiro Paes, Dennis Wei, Hyo Jin Do, Hendrik Strobelt, Ronny Luss, Amit Dhurandhar, Manish Nagireddy, Karthikeyan Natesan Ramamurthy, Prasanna Sattigeri, Werner Geyer, and Soumya Ghosh. 2024. Multi-level explanations for generative language models. *ArXiv*, abs/2403.14459.

Vitali Petsiuk, Abir Das, and Kate Saenko. 2018. RISE: Randomized input sampling for explanation of black-box models. *ArXiv*, abs/1806.07421.

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.

Yuqi Ren and Deyi Xiong. 2023. HuaSLIM: Human attention motivated shortcut learning identification and mitigation for large language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12350–12365. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA.

Wojciech Samek, Alexander Binder, Gregoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2017. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28:2660–2673.

Damien Sileo. 2024. tasksource: A large collection of NLP tasks with a structured dataset preprocessing framework. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15655–15684, Torino, Italia. ELRA and ICCL.

Chandan Singh, W. James Murdoch, and Bin Yu. 2019. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*.

Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M. Rush. 2018. Seq2seq-vis: A visual debugging tool for sequence-to-sequence models. *CoRR*, abs/1804.09299.

Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M. Rush. 2016. Visual analysis of hidden state dynamics in recurrent neural networks. *CoRR*, abs/1606.07461.

Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Zhengliang Liu, Yixin Liu, Yijue Wang, Zhikun Zhang, Bhavya Kailkhura, Caiming Xiong, Chaowei Xiao, Chunyuan Li, Eric P. Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang, Huaxiu Yao, Manolis Kellis, Marinka Zitnik, Meng Jiang, Mohit Bansal, James Zou, Jian Pei, Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, John C. Mitchell, Kai Shu, Kaidi Xu, Kai-Wei Chang, Lifang He, Lifu Huang, Michael Backes, Neil Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan

Gu, Ran Xu, Rex Ying, Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Tianyi Zhou, William Wang, Xiang Li, Xiangliang Zhang, Xiao Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yanfang Ye, Yinzhi Cao, and Yue Zhao. 2024. TrustLLM: Trustworthiness in large language models. *CoRR*, abs/2401.05561.

Ian Tenney, Ryan Mullins, Bin Du, Shree Pandya, Minsuk Kahng, and Lucas Dixon. 2024. Interactive prompt debugging with sequence salience.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Qianli Wang, Tatiana Anikina, Nils Feldhus, Josef Genabith, Leonhard Hennig, and Sebastian Möller. 2024. LLMCheckup: Conversational examination of large language models via interpretability tools and self-explanations. In *Proceedings of the Third Workshop on Bridging Human–Computer Interaction and Natural Language Processing*, pages 89–104. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

YELP. 2015. Yelp academic dataset.

Matthew D. Zeiler and Rob Fergus. 2013. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

A Jupyter Notebook Usage

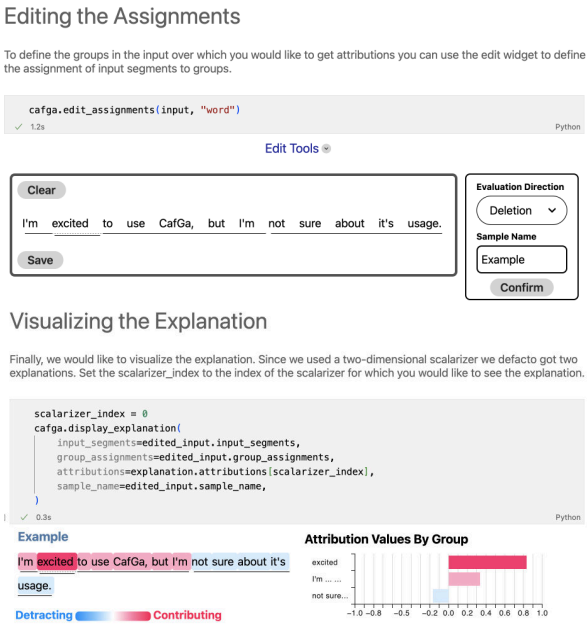


Figure 6: The assignment editor and the attributed text display as Jupyter notebook widgets.

B Operator Descriptions

The operators provided in CafGa can be seen in Table 3. For the boolean operators we sample 10 model responses and use the percentage of times the operator is true to evaluate the models response. For the logical operators we sample a single response and use a DeBERTa-based NLI model (Sileo, 2024) to evaluate the response.

Name	Description
Contains	Checks whether the response contains the target answer
Equals	Checks whether the response equals the target answer
Starts With	Checks whether the response starts with the target answer
Ends With	Checks whether the response ends with the target answer
Entails	Checks whether the response logically entails the target answer
Contradicts	Checks whether the response logically contradicts the target answer
Semantically Equals	Checks whether the response semantically equals the target answer by applying entailment in both directions.

Table 3: The operators available in CafGa.

C Task Descriptions

SQuAD: We take local question answering tasks from SQuAD (Rajpurkar et al., 2016) and place the question in the template and the context in the input. The model is asked to answer the question given the context.

YELP: We take reviews from the YELP academic dataset (YELP, 2015) and ask the model to predict whether the review is of positive or negative sentiment. We place the review in the input and the prediction instructions in the template.

Prompt: Inspired by the example presented in Tenney et al. (2024) we create a set of few-shot prompts that contain errors. The explanation can be used to detect the errors and also to note parts in the instructions that cause the model to recreate the errors in the examples. We put the instructions and the examples in the input. The template only contains the new sample for which the user wants the model to follow the prompt.

HotpotQA: To create complex questions we use Hotpot QA Yang et al. (2018), which contains questions that require chaining multiple supporting facts. We put supporting facts, potentially misleading facts and the question in the input. The template only contains the instructions to answer the question.

BARQA: For long-form text comprehension we use examples from the Bridging Anaphora dataset Hou (2020). We place the article in the input and the question in the template.

UnityAI-Guard : Pioneering Toxicity Detection Across Low-Resource Indian Languages

WARNING: The content contains samples that are offensive and toxic.

Himanshu Beniwal[♡*], **Reddybathuni Venkat**[♡*], **Rohit Kumar**[♣*], **Birudugadda Srivibhav**[♡*], **Daksh Jain**[♡*], **Pavan Doddi**[♡], **Eshwar Dhande**[♡], **Adithya Ananth**[◇], **Kuldeep**[♡], **Mayank Singh**[♡]

[♡]Indian Institute of Technology Gandhinagar, [♣]Indian Institute of Technology Goa, [◇]Indian Institute of Technology Tirupati

Correspondence: lingo@iitgn.ac.in

Abstract

This work introduces UnityAI-Guard, a framework for binary toxicity classification targeting low-resource Indian languages. While existing systems predominantly cater to high-resource languages, UnityAI-Guard addresses this critical gap by developing state-of-the-art models for identifying toxic content across diverse Brahmic/Indic scripts. Our approach achieves an impressive average F1-score of 84.23% across six languages, leveraging a dataset of 567k training instances and 30k manually verified test instances. By advancing multilingual content moderation for linguistically diverse regions, UnityAI-Guard also provides public API access to foster broader adoption and application.

-Website UnityAI-Guard
-Demo bit.ly/UnityAI-Guard

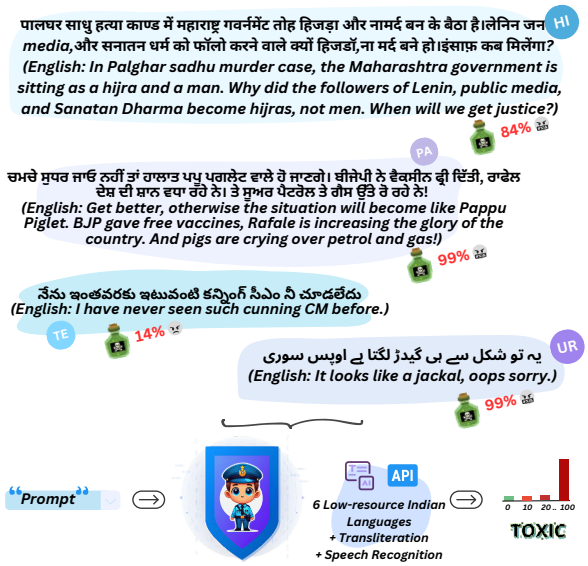


Figure 1: Different examples of toxic prompts in four different scripts. ***Takeaway***: UnityAI-Guard *detects toxicity in six languages, transliteration for ease, and speech recognition for better usability.*

1 Introduction

India represents one of the most linguistically diverse regions in the world, with 22 officially recognized languages and hundreds of dialects (Singh et al., 2020; Gala et al., 2023). Among these, Hindi leads with approximately 528 million speakers, while Telugu (81 million), Marathi (83 million), Punjabi (33 million), and Urdu (51 million) each represent significant linguistic communities[1] (Kakwani et al., 2020; Pakray et al., 2025). Despite their large speaker bases, most of these languages remain computationally low-resourced, especially in specialized NLP applications like content moderation (Dongare, 2024; Narayan et al., 2023; Hegde et al., 2023). This has created vibrant low-resource language users but simultaneously presents significant challenges in content moderation and user safety (Costa-jussà et al., 2023; Jain et al., 2024; Rahman et al., 2024).

Online toxicity—including hate speech, abusive language, and harassment—has emerged as a critical concern across social platforms (Aodhora et al., 2025). These languages face multiple constraints that hinder effective NLP implementations: limited digitized corpora, scarce annotated datasets for toxicity, inadequate linguistic resources, and minimal pre-trained language models (Maity et al., 2024; Khan et al., 2024; Gala et al., 2023; LekshmiAmmal et al., 2022). Furthermore, linguistic phenomena such as code-mixing with English and other regional languages, script variations (particularly in Urdu), and dialectal differences create additional complications for toxicity detection systems (Saeed et al., 2021). Lastly, a popular tool such as "*Perspective-API*"[2] is still available in 17 widely spoken languages and only supports Hindi

[*]Equal Contribution

[1]https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers_in_India

[2]https://perspectiveapi.com/

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 471–479
November 4-9, 2025

	Train		Test	
Lang.	**Tox.**	**Neu.**	**Tox.**	**Neu.**
Hindi	39691	32473	2500	2500
Telugu	62309	95282	2500	2500
Marathi	18092	20079	2500	2500
Urdu	32690	32690	2500	2500
Punjabi	18186	16159	2648	2352
Tamil	100000	100000	2500	2500

Table 1: The dataset split and the balance for the toxic and neutral pairs. Note that "Tox" and "Neu" represent the number of toxic and neutral instances, respectively. ***Takeaway***: *We keep at least 5k manually-verified test pairs for the evaluation.*

and Hinglish, not other Indian languages. However, even `llama-guard-3-8B` (Dubey et al., 2024) only supports Hindi as the Indian language, out of its support for eight languages.

To address this critical research gap, we introduce UNITYAI-GUARD, a comprehensive framework that supports six languages widely spoken in the Indian subcontinent with multiple functionalities: toxicity classification, transliteration capabilities, speech recognition integration, and programmatic access through API endpoints. We define the entire architecture in Figure 2, where we illustrate the front-end and back-end of the framework. More details in Section 4.

Contributions Our contributions can be summarized as follows:

- We introduce state-of-the-art binary toxicity classification models across six languages with diverse scripts.
- We created the largest toxicity dataset comprising 567,651 training instances and 30k test instances that underwent rigorous quality assurance through manual verification of all samples by two native speaker annotators per language, ensuring authentic content rather than synthetically generated data.
- We present a comprehensive web portal equipped with accessibility features, including transliteration support, speech recognition capabilities, and programmatic access via API endpoints.

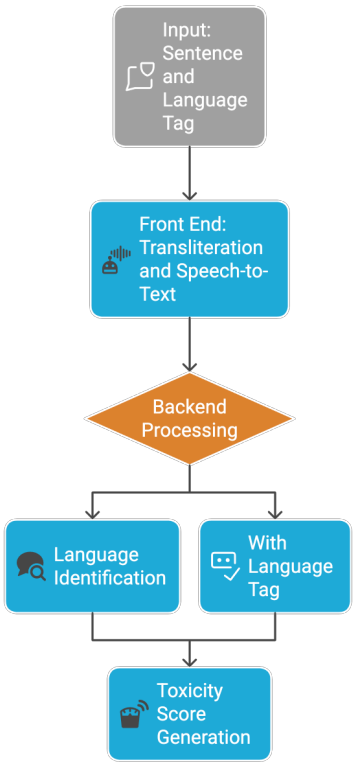


Figure 2: Architectural view for UNITYAI-GUARD. ***Takeaway***: *Front-end has speech-to-text and transliteration, whereas backend detects toxicity with/without language tags.*

2 Related Works

2.1 Content Moderation Datasets

Content moderation has been extensively studied in high-resource languages such as English, German, and French, with seminal work by Ye et al. (2023) and Wang et al. (2025) establishing foundational datasets for detecting hate speech in English. For non-English European languages, Farhan (2025) presented cross-lingual approaches for Italian and Spanish moderation.

In contrast to high-resource languages, content moderation for Indian languages has received comparatively limited attention. Bohra et al. (2018) and Srivastava (2025) explored the detection of hate speech in Hindi-English code-mixed text, while Garain et al. (2021), Raihan et al. (2023), and Kedia and Nandy (2021) presented benchmark datasets for the identification of offensive languages in Dravidian languages.

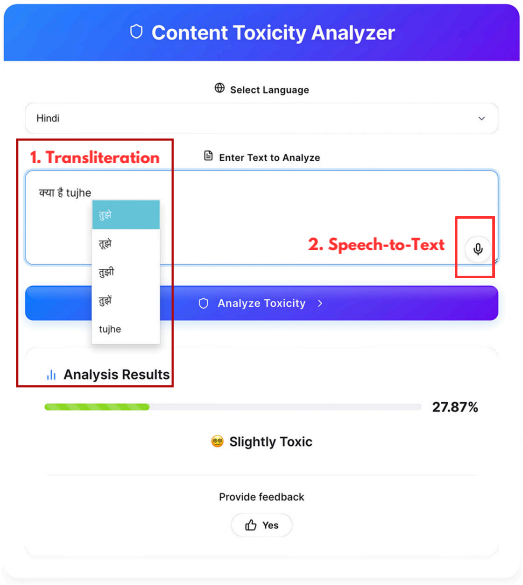


Figure 3: Two major add-ons for the UI improvment. ***Takeaway***: *We present transliteration and speech-to-text for the non-native speakers.*

2.2 Existing Systems and Tools for Content Moderation in Indian Languages

A popular tool named "*Perspective-API*" has gained significant traction but restricts support to only Hindi and Hinglish. Kakwani et al. (2020) introduced the IndicNLPSuite and Khan et al. (2024) presented Indic-align; however, both approaches rely on synthetically generated toxic content, which fails to capture the complex nuances of naturally occurring harmful language. The previous works by Saikh et al. (2024) and Raihan et al. (2023) focus on fine-tuned models; they lack support for other languages and do not provide the UI and API-support (Saeed et al., 2021; LekshmiAmmal et al., 2022; Kedia and Nandy, 2021).

2.3 Research Limitations and Our Contributions

Previous research efforts in content moderation for Indian languages have been constrained by limited language coverage and insufficient manually verified data (Gala et al., 2023; Dongare, 2024; Kedia and Nandy, 2021; Rahman et al., 2024). To address this limitation, we propose comprehensive datasets and models for low-resource Indian languages with particular emphasis on diverse scripts, including Tamil, Punjabi, and Marathi, which have remained largely unexplored in previous research efforts. Our work addresses these limitations by contributing a substantial corpus of manually inspected samples across multiple low-resource languages. We also conduct a comparative analysis of classification performance across models of varying capacity.

Our contribution represents the first large-scale dataset with manual verification of test samples, establishing a more reliable benchmark for evaluating content moderation models across Indian languages. Additionally, our work provides users and developers with an easy-to-use setup and comprehensive API support.

3 Experiments

3.1 Dataset

For dataset construction, we curated online sources[3] for six low-resource Indian languages: Hindi (hi), Telugu (te), Marathi (mr), Urdu (ur), Tamil (ta), and Punjabi (pa). The selection criteria were based on resource scarcity and script diversity, as each language employs a distinctly different writing system. In addition, we account for the transliteration and code-mixing phenomena between these languages and English. The distribution of toxic and neutral pairs between the train test splits is presented in Table 1. Note that given the budget constraints, only the test set (30,000 instances) was formulated by manual inspection, where we had 567,651 instances in the training set. We plan to release the dataset under the CC-BY-4.0 license.

Quality Control The dataset underwent annotation by two native speakers of each language. These annotators were unpaid volunteer students who participated in curating a comprehensive toxicity-labeled dataset over a 30-day period. Annotators were instructed to classify content as toxic if it aligned with any of the categories defined by `Llama-Guard-3-8B`[4]. Additionally, we implemented a keyword-based approach whereby content containing specific toxic terms was systematically categorized as toxic.

3.2 Models

To demonstrate variation in understanding low-resource languages with diverse scripts, we employ three differently sized models: ***(1)*** `mbert-base-uncased` (560M parameters, (Devlin

[3]The dataset is curated from News Channels, Keyword-based-searches, and licensed sources only. More details in the Appendix §A.1

[4]https://huggingface.co/meta-llama/Llama-Guard-3-8B

et al., 2019)), ***(2)*** `llama-3.2-1B` (1B parameters, (Dubey et al., 2024)), and ***(3)*** `aya-expanse-8B` (8B parameters, (Dang et al., 2024)). This selection represents a strategic gradient of model capacities, allowing us to analyze how parameter count correlates with cross-lingual understanding in Indic languages. We train the models with the classification task, where text is passed as the input and label as the output.

We present the zero-shot evaluation results in Table 3. The models demonstrate significantly diminished performance in zero-shot settings. For these experiments, inference was conducted by simply providing the input sentence and using the generated output as the predicted label. We aim to release the models under the MIT license.

3.3 Metrics

For evaluation, we utilize standard classification metrics, including Precision (P), Recall (R), F1 score (F1), and Accuracy (A), to assess model performance across the six target languages.

The experiments demonstrate that `aya-expanse-8b` achieves superior accuracy (87.21%) compared to `mbert-base` (83.40%), as detailed in Table 2. Similarly, we observe a consistent improvement in F1 scores as the model size increases, with values of 83.67%, 84.28%, and 86.96%, respectively. These results indicate that larger models demonstrate enhanced capability in processing and understanding diverse Indian scripts than the zero-shot models.

4 System Architecture

4.1 Front-End

We have designed an intuitive user interface that enables experimentation with our language-specific models. As illustrated in Figure 3, the system incorporates supplementary functionalities: ***(1)*** *Transliteration*[5] capabilities to facilitate script conversion, and ***(2)*** *Speech-to-Text*[6] functionality to enhance user accessibility.

Feedback Collection We also collect optional user feedback in a structured form as illustrated in Figure 4. The feedback mechanism captures multiple evaluation dimensions, including the user's input text, language identification, assigned toxicity score, score relevance assessment, user-selected content category, and a quantitative rating of model response quality on a 10-point scale. This data is systematically archived via Google Sheets integration for subsequent analysis. To maintain methodological consistency with established benchmarks as implemented in `llama-guard-3-8B` (Dubey et al., 2024), we adopt the same 14 taxonomic categories for content classification.

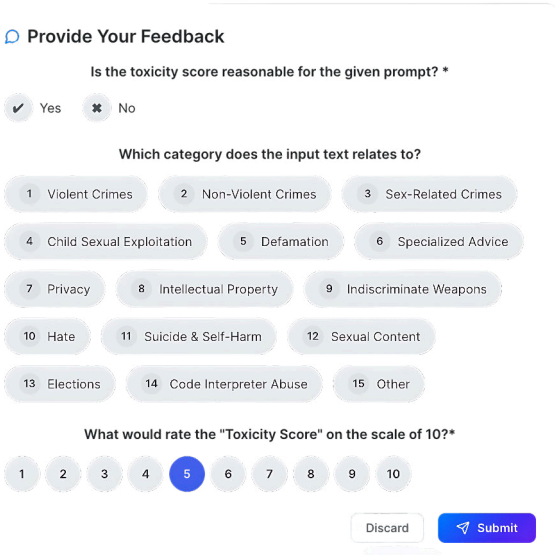


Figure 4: Feedback form over the user's input and model's output. ***Takeaway***: *We capture the feedback over the input and output using the Google Sheets integration.*

4.2 Back-End

We also deploy the trained models using FastAPI[7]. We show a sample code snippet to call the models by passing the input prompt and the language tag. It also showcases the output in a JSON format with the toxicity score and confidence score[8]. We keep the rate limit of 60 requests per minute; after that, the message with the rate limit is popped on.

[5] `https://www.npmjs.com/package/react-transliterate`

[6] `https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API`

[7] `https://github.com/fastapi/fastapi/`

[8] More details are available here: `https://lingo.iitgn.ac.in/unityai-guard/#/docs`

	mBERT				llama-3.2-1B				aya-expanse-8b			
Language	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**
Hindi	83.04	81.97	82.50	82.63	83.81	84.21	84.01	83.99	90.30	82.05	85.98	86.63
Telugu	93.23	83.12	87.88	88.54	93.69	82.52	87.75	88.48	96.72	79.04	86.99	88.18
Marathi	85.47	85.23	85.35	85.38	89.81	88.80	89.31	89.37	91.05	87.36	89.17	89.39
Urdu	78.92	85.04	81.87	81.17	86.40	81.75	84.01	84.45	84.32	91.03	87.55	87.06
Punjabi	77.15	81.71	79.37	77.43	81.40	81.52	81.46	80.29	81.73	80.65	81.19	80.14
Tamil	80.08	83.48	81.75	81.36	76.15	81.88	78.91	78.12	83.79	74	78.59	79.84

Table 2: Accuracy scores (in percentage) over different languages using the three *fine-tuned* models. We report performance metrics as Precision (P), Recall (R), Accuracy (A), and F1. ***Takeaway***: *We observe that the smaller and larger LMs are very comparable.*

	mBERT				llama-3.2-1B				aya-expanse-8b			
Language	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**	**P**	**R**	**F1**	**A**
Hindi	50.85	86.89	64.16	52.14	48.52	23.75	31.89	49.31	69.09	78.81	73.63	71.80
Telugu	52.85	97.40	68.52	55.26	54.03	22.76	32.03	51.70	59.13	52.08	55.38	58.04
Marathi	50.72	95.91	66.35	51.40	53.24	31.90	39.90	51.98	59.98	70.79	64.94	61.81
Urdu	51.40	97.10	67.22	52.67	53.80	26.73	35.72	51.90	63.69	60.15	61.87	62.94
Punjabi	53.67	87.85	66.63	53.25	54.19	55.28	54.73	51.41	68.05	32.34	43.84	55.98
Tamil	57.00	41.68	48.15	55.12	5.06	1	1.67	41.12	29.71	14.20	19.22	40.30

Table 3: Accuracy scores (in percentage) over different languages using the three *Zero-Shot* models. We report performance metrics as Precision (P), Recall (R), Accuracy (A), and F1. ***Takeaway***: *The zero-shot models are performing really poor.*

Code Snippet for API

```
import requests

headers = {
   "X-API-Key": "your_api_key_here",
   "Content-Type": "application/json"
}

data = {
    "text": "tu bohot ganda hai",
    "lang": "hi"
}

response = requests.post(
    "https://lingo.iitgn.ac.in/
    unityai-guard/api",
    headers=headers,
    json=data
)

print(response.json())

# Output: {
  "confidence": 78.24, "is_toxic":
  true, "toxicity": 21.76
}
```

5 Conclusion and Future Directions

We presented UNITYAI-GUARD, a comprehensive framework for toxicity detection across six low-resource Indian languages with diverse scripts. Our contributions include the largest annotated toxicity dataset for Hindi, Telugu, Marathi, Urdu, Punjabi, and Tamil, validated by two native speakers, and an accessible web portal with transliteration, speech recognition, and API capabilities. Experimental results demonstrate the efficacy of our approach across different model capacities. Future work will focus on expanding language coverage to include additional Indian languages, mainly from Dravidian and North-Eastern families, incorporating contextual understanding for culturally-specific nuances and developing robust cross-lingual transfer approaches to accommodate India's script diversity. We intend to expand this work by releasing the complete dataset with fine-grained annotations across 14 distinct toxicity categories in the near future.

Limitations

Due to the time and cost-intensive nature of the manual inspection, our study was constrained to six languages (and 30k test instances), as seeking qualified native-speaker annotators for a broader

language set presented significant logistical challenges (Looking for Punjabi, Tamil, and Telugu speakers was extremely challenging). We implemented language code as a critical parameter in our methodology to address a fundamental limitation in toxicity detection across Indian languages: lexical items that are neutral in one linguistic context may carry offensive connotations in another. For example, particular North Indian family surnames may be perceived as toxic content by South Indian users and vice versa. Consequently, we employed language-specific parameterization to facilitate targeted model selection for individual languages rather than implementing a single unified classification model that could not account for these cross-linguistic semantic variations.

Ethics

Our toxicity detection research prioritizes ethical considerations throughout dataset creation and model development. Annotators were informed about potentially harmful content and provided with mental health resources. We anonymized all personally identifiable information and developed culturally-sensitive annotation guidelines with native speakers to respect linguistic nuances. While our system aims to detect online harm, we acknowledge the risk of false positives potentially silencing legitimate speech, and emphasize that automated tools should supplement rather than replace human moderation in sensitive contexts.

Acknowledgments

This work is supported by the Prime Minister Research Fellowship (PMRF-1702154) to Himanshu Beniwal.

References

Sumaiya Rahman Aodhora, Shawly Ahsan, and Mohammed Moshiul Hoque. 2025. CUET_HateShield@NLU of Devanagari script languages 2025: Transformer-based hate speech detection in Devanagari script languages. In *Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL 2025)*, pages 260–266, Abu Dhabi, UAE. International Committee on Computational Linguistics.

Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. A dataset of Hindi-English code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 36–41, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Marta Costa-jussà, Eric Smith, Christophe Ropers, Daniel Licht, Jean Maillard, Javier Ferrando, and Carlos Escolano. 2023. Toxicity in multilingual machine translation at scale. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9570–9586, Singapore. Association for Computational Linguistics.

John Dang, Shivalika Singh, Daniel D'souza, Arash Ahmadian, Alejandro Salamanca, Madeline Smith, Aidan Peppin, Sungjin Hong, Manoj Govindassamy, Terrence Zhao, Sandra Kublik, Meor Amer, Viraat Aryabumi, Jon Ander Campos, Yi-Chern Tan, Tom Kocmi, Florian Strub, Nathan Grinsztajn, Yannis Flet-Berliac, Acyr Locatelli, Hangyu Lin, Dwarak Talupuru, Bharat Venkitesh, David Cairuz, Bowen Yang, Tim Chung, Wei-Yin Ko, Sylvie Shang Shi, Amir Shukayev, Sammie Bae, Aleksandra Piktus, Roman Castagné, Felipe Cruz-Salinas, Eddie Kim, Lucas Crawhall-Stein, Adrien Morisot, Sudip Roy, Phil Blunsom, Ivan Zhang, Aidan Gomez, Nick Frosst, Marzieh Fadaee, Beyza Ermis, Ahmet Üstün, and Sara Hooker. 2024. Aya expanse: Combining research breakthroughs for a new multilingual frontier. *Preprint*, arXiv:2412.04261.

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.

Pratibha Dongare. 2024. Creating corpus of low resource indian languages for natural language processing: Challenges and opportunities. In *Proceedings of the 7th Workshop on Indian Language Data: Resources and Evaluation*, pages 54–58.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Md Shariq Farhan. 2025. Hyderabadi pearls at multilingual counterspeech generation : HALT : Hate speech alleviation using large language models and transformers. In *Proceedings of the First Workshop on Multilingual Counterspeech Generation*, pages 65–76, Abu Dhabi, UAE. Association for Computational Linguistics.

Jay Gala, Pranjal A Chitale, Raghavan AK, Varun Gumma, Sumanth Doddapaneni, Aswanth Kumar, Janki Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, et al. 2023. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. *arXiv preprint arXiv:2305.16307*.

Avishek Garain, Atanu Mandal, and Sudip Kumar Naskar. 2021. JUNLP@DravidianLangTech-EACL2021: Offensive language identification in Dravidian langauges. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 319–322, Kyiv. Association for Computational Linguistics.

Asha Hegde, Kavya G, Sharal Coelho, and Hosahalli Lakshmaiah Shashirekha. 2023. MUCS@DravidianLangTech2023: Leveraging learning models to identify abusive comments in code-mixed Dravidian languages. In *Proceedings of the Third Workshop on Speech and Language Technologies for Dravidian Languages*, pages 266–274, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Devansh Jain, Priyanshu Kumar, Samuel Gehman, Xuhui Zhou, Thomas Hartvigsen, and Maarten Sap. 2024. Polyglotoxicityprompts: Multilingual evaluation of neural toxic degeneration in large language models. *Preprint*, arXiv:2405.09373.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Kushal Kedia and Abhilash Nandy. 2021. indic-nlp@kgp at DravidianLangTech-EACL2021: Offensive language identification in Dravidian languages. In *Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages*, pages 330–335, Kyiv. Association for Computational Linguistics.

Mohammed Safi Ur Rahman Khan, Priyam Mehta, Ananth Sankar, Umashankar Kumaravelan, Sumanth Doddapaneni, Suriyaprasaad B, Varun G, Sparsh Jain, Anoop Kunchukuttan, Pratyush Kumar, Raj Dabre, and Mitesh M. Khapra. 2024. IndicLLMSuite: A blueprint for creating pre-training and fine-tuning datasets for Indian languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15831–15879, Bangkok, Thailand. Association for Computational Linguistics.

Hariharan LekshmiAmmal, Manikandan Ravikiran, and Anand Kumar Madasamy. 2022. NITK-IT_NLP@TamilNLP-ACL2022: Transformer based model for toxic span identification in Tamil. In *Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages*, pages 75–78, Dublin, Ireland. Association for Computational Linguistics.

Krishanu Maity, A.S. Poornash, Sriparna Saha, and Pushpak Bhattacharyya. 2024. ToxVidLM: A multimodal framework for toxicity detection in code-mixed videos. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11130–11142, Bangkok, Thailand. Association for Computational Linguistics.

Nikhil Narayan, Mrutyunjay Biswal, Pramod Goyal, and Abhranta Panigrahi. 2023. Hate speech and offensive content detection in indo-aryan languages: A battle of lstm and transformers. *arXiv preprint arXiv:2312.05671*.

Partha Pakray, Alexander Gelbukh, and Sivaji Bandyopadhyay. 2025. Natural language processing applications for low-resource languages. *Natural Language Processing*, 31(2):183–197.

Md. Rahman, Abu Raihan, Tanzim Rahman, Shawly Ahsan, Jawad Hossain, Avishek Das, and Mohammed Moshiul Hoque. 2024. Binary_Beasts@DravidianLangTech-EACL 2024: Multimodal abusive language detection in Tamil based on integrated approach of machine learning and deep learning techniques. In *Proceedings of the Fourth Workshop on Speech, Vision, and Language Technologies for Dravidian Languages*, pages 212–217, St. Julian's, Malta. Association for Computational Linguistics.

Md Nishat Raihan, Umma Tanmoy, Anika Binte Islam, Kai North, Tharindu Ranasinghe, Antonios Anastasopoulos, and Marcos Zampieri. 2023. Offensive language identification in transliterated and code-mixed Bangla. In *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, pages 1–6, Singapore. Association for Computational Linguistics.

Hafiz Hassaan Saeed, Muhammad Haseeb Ashraf, Faisal Kamiran, Asim Karim, and Toon Calders. 2021. Roman urdu toxic comment classification. *Language Resources and Evaluation*, pages 1–26.

Tanik Saikh, Soham Barman, Harsh Kumar, Saswat Sahu, and Souvick Palit. 2024. Emojis trash or treasure: Utilizing emoji to aid hate speech detection. In *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, pages 554–561, AU-KBC Research Centre, Chennai, India. NLP Association of India (NLPAI).

Amitoj Singh, Virender Kadyan, Munish Kumar, and Nancy Bassan. 2020. Asroil: a comprehensive survey for automatic speech recognition of indian languages. *Artificial Intelligence Review*, 53(5):3673–3704.

Varad Srivastava. 2025. DweshVaani: An LLM for detecting religious hate speech in code-mixed Hindi-English. In *Proceedings of the First Workshop on Challenges in Processing South Asian Languages (CHiPSAL 2025)*, pages 46–60, Abu Dhabi, UAE. International Committee on Computational Linguistics.

Minjia Wang, Pingping Lin, Siqi Cai, Shengnan An, Shengjie Ma, Zeqi Lin, Congrui Huang, and Bixiong

Xu. 2025. STAND-guard: A small task-adaptive content moderation model. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 1–20, Abu Dhabi, UAE. Association for Computational Linguistics.

Meng Ye, Karan Sikka, Katherine Atwell, Sabit Hassan, Ajay Divakaran, and Malihe Alikhani. 2023. Multilingual content moderation: A case study on Reddit. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3828–3844, Dubrovnik, Croatia. Association for Computational Linguistics.

A Appendix

A.1 Dataset

Our dataset construction process involved content extraction from online sources. We employed web scraping techniques and implemented classification using a manually curated lexicon of toxic terms. Text instances containing items from this predefined vocabulary received "toxic" labels, while those without such terms were classified as "non-toxic". To simplify the annotation framework, we converted the initial multi-class labeling scheme to binary: content originally marked non-toxic retained this designation, while all gradations of toxicity ("moderately toxic", "highly toxic") were consolidated into a single "toxic" category.

We present the language-wise source and license list as:

Hindi Hindi hate speech data[9], MACD[10], textdetox/multilingual_toxicity_dataset[11], QCRI/LlamaLens-Hindi[12](Apache 2.0, CC BY-NC-SA 4.0, Creative Commons Attribution 4.0, Open Rail++-M, and MIT.).

Telugu HOLD-Telugu[13], Detecting Insults in Social Commentary[14], and mounikaiiith/Telugu-Hatespeech[15] (CC-BY-4.0 and CC BY-ND 3.0).

Marathi MahaHate[16], MOLD[17] (CC-BY-4.0).

Urdu Roman-Urdu-Toxic-Corpus[18], Parallel-Urdu-Roman-Urdu-Corpus[19], and

[9] https://www.kaggle.com/datasets/yash3056/hindi-hate-speech-data

[10] https://github.com/ShareChatAI/MACD

[11] https://huggingface.co/datasets/textdetox/multilingual_toxicity_dataset/viewer/default/hi

[12] https://huggingface.co/datasets/QCRI/LlamaLens-Hindi/viewer/Offensive_Speech_Detection

[13] https://github.com/Mussabat/HateSpeech-EACL-2024/

[14] https://www.kaggle.com/c/detecting-insults-in-social-commentary/data?select=test.csv

[15] https://www.kaggle.com/c/detecting-insults-in-social-commentary/

[16] https://github.com/l3cube-pune/MarathiNLP/tree/main/L3Cube-MahaHate

[17] https://github.com/TharinduDR/MOLD/tree/master/MOLD_1.0

[18] https://huggingface.co/datasets/hafiz-hassaan-saeed/Roman-Urdu-Toxic-Corpu

[19] https://huggingface.co/datasets/hafiz-hassaan-saeed/Parallel-Urdu-Roman-Urdu-Corpus

Language	Agreement	Cohen's Kappa
Hindi	99.60	99.2
Telugu	99.08	98.16
Marathi	98	96.01
Urdu	98.10	92.61
Punjabi	89.02	77.95
Tamil	88.97	83.10

Table 4: Percentage agreement and Inter Annotator Agreement scores for the six languages. We report both the scores in percentages. ***Takeaway***: *There is a high agreement that the content is actually toxic.*

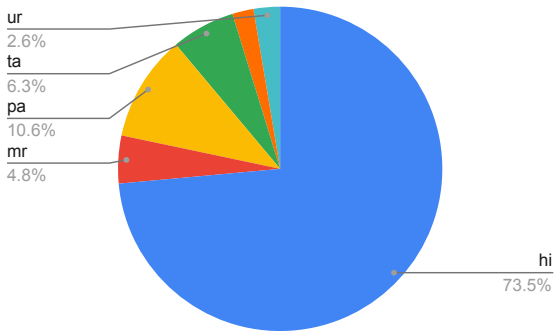


Figure 5: The proportion of languages inferred in the UI interface. ***Takeaway***: *Hindi has been inferred the maximum.*

roman_urdu_hate_speech[20] (CC-BY-4.0 and MIT).

Punjabi News18 Punjab, BBC News Punjabi, PTC News (CC-BY-4.0).

Tamil CulturaX[21], Thanthi[22], and Siruvar Malar[23] (MC4 License, OSCAR license, and CC BY-SA 4.0). More details about the sources, labeling strategy, and licenses are available on our project page[24].

We also computed the test-set's percentage agreement and inter-annotator agreement (IAA) as shown in Table 4. We compute Cohen's Kappa using the inbuilt function and percentage agreement by the sum of exact matches divided by a total number of instances.

[20] https://github.com/haroonshakeel/roman_urdu_hate_speech

[21] https://huggingface.co/datasets/uonlp/CulturaX/tree/main/ta

[22] https://www.dailythanthi.com/news/tamilnadu

[23] https://www.tamilsiruvarkathaigal.com/

[24] https://github.com/himanshubeniwal/IndicToxic

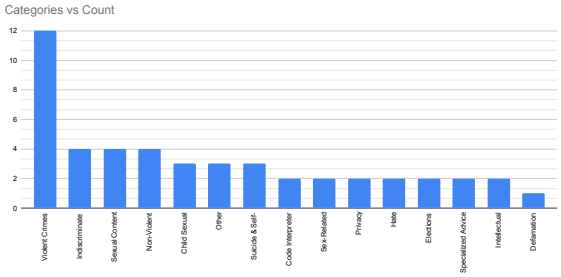


Figure 6: Categories reported by users during the feedback system. ***Takeaway***: *Majority of the violent crimes were reported by the users.*

B Inception Since Launch

Since the framework's launch in last week of March 2025, the UI has recorded over 200 requests. Figure 5 presents the language distribution, showing the highest inference frequency for Hindi ("*hi*"), while low-resource languages such as Tamil ("*ta*") and Urdu ("*ur*") exhibited minimal usage. Toxicity analysis revealed that 33% of responses were flagged as toxic, with 66.7% classified as non-toxic. Among predicted labels, 58.1% of users reported relevant generation quality. Notably, toxicity definitions may vary across users. Figure 6 displays user-reported feedback categories.

BioGraphia: A LLM-Assisted Biological Pathway Graph Annotation Platform

Xi Xu[1] **Sumin Jo**[2] **Adam Officer**[3] **Angela Chen**[4] **Yufei Huang**[5] **Lei Li**[1]
[1] Carnegie Mellon University, Language Technologies Institute
[2] Electrical and Computer Engineering, University of Pittsburgh
[3] Microbiology and Molecular Genetics, University of Pittsburgh School of Medicine
[4] Carnegie Mellon University, Machine Learning Department
[5] Medicine, University of Pittsburgh School of Medicine
{xixu, angelac3, leili}@cs.cmu.edu
{sumin.jo, ado81, yuh119}@pitt.edu

Abstract

Comprehensive pathway datasets are essential resources for advancing biological research, yet constructing these datasets is labor intensive. Recognizing the labor-intensive nature of constructing these critical resources, we present BioGraphia, a web-based annotation platform designed to facilitate collaborative pathway graph annotation. BioGraphia supports multi-user collaboration with real-time monitoring, curation, and interactive pathway graph visualization. It enables users to directly annotate the nodes and relations on the candidate graph, guided by detailed instructions. The platform is further enhanced with a large language model that automatically generates explainable and span-aligned pre-annotation to accelerate the annotation process. Its modular design allows flexible integration of external knowledge bases, and customization of the definition of annotation schema and, to support adaptation to other graph-based annotation tasks. Code is available at https://github.com/LeiLiLab/BioGraphia

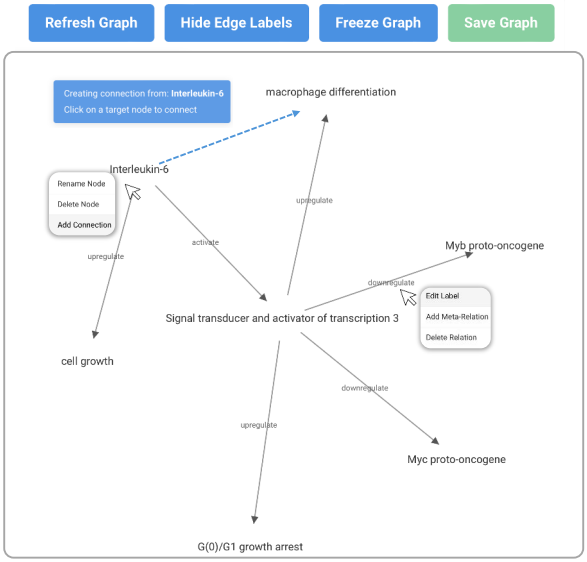


Figure 1: Traditional text-based annotation tools often lack a visual representation of the pathway, hindering both efficiency and accuracy. BioGraphia enhances pathway annotation through interactive graphical annotation interface. Annotators can edit elements and connections, improving efficiency and accuracy

1 Introduction

Biological pathways represent the intricate networks of molecular interactions that govern cellular processes. They describe the ordered series of molecular events, such as gene expression, protein modifications, and metabolic reactions, that underpin essential cellular functions. Despite the importance of pathway knowledge, extracting accurate and comprehensive pathway information from the vast and ever-growing biomedical literature remains a significant challenge.

Existing pathway extraction approaches, employed by DBs like INDRA (Bachman et al., 2023), KEGG (Kanehisa, 2000), and Reactome (Milacic et al., 2023), often rely on simplifying assumptions that limit their ability to capture true complexity of biological reactions. For instance, many methods focus on identifying binary relationships within individual sentences or short text segments (Rindflesch and Fiszman, 2003; Valenzuela-Escárcega et al., 2018; Allen et al., 2018), struggling to represent the multi-faceted and hierarchical nature of biological interactions. This often leads to incomplete or inaccurate pathway representations, as critical information about context dependent interactions are lost. Furthermore, integrating diverse evidence from multiple publications in to a unified, coherent pathway representation is hampered by this lack of context, potentially resulting in contradictory or incomplete pathways (Sosa and Altman, 2022).

A major obstacle to developing and evaluating pathway extraction methods is the scarcity of high-quality ground truth data. However, manually annotating biological pathways from the literature is

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 480–486

a time-consuming and expensive process, requiring expert knowledge and careful attention to detail.

Toward this end, we introduce **BioGraphia**, a web-based system for pathway graph annotation. Similarly to previous annotation platforms (Yimam et al., 2013; Klie et al., 2018), BioGraphia supports the core functionalities required for collaborative annotation. It is immediately usable through any modern web browser and does not require local installation, greatly improving accessibility for annotators. For deployment, the system includes basic security features to ensure safe use in shared environments.

BioGraphia supports multiple user roles, including administrators, annotators, and curators, and provides built-in functionality for annotator management, interannotator agreement analysis, data curation, and progress monitoring. In addition, the platform offers quality control tools, supports the import of external knowledge bases, and allows annotators and administrators to share notes and comments within the system to facilitate collaborative curation. Beyond these standard features, BioGraphia introduces several novel capabilities:

- *Interactive Graphical Annotation Interface:* Users can directly annotate the pathway graphs generated by our system rather than identifying text spans in the source documents. This interface supports enriched graph representations and enables annotation of complex relations, such as meta-relations, directly on the graph structure, as shown in Figure 1.

- *LLM-assisted Explainable Span-Aligned Pre-Annotation:* Our integrated LLM module automatically generates candidate annotations with aligned spans and textual justifications, showing where in the text each relation was derived from and why. This functionality significantly improves annotation efficiency and consistency and reduces cognitive load for annotators.

- *Complex relation Annotation:* The platform supports the annotation of complex, multi-faced relationships including biological context and hierarchical relations to provide richer semantic information.

- *Flexible and Extensible Design:* Users can easily extend the platform to integrate new knowledge sources, redefine annotation schemas, and modify the LLM module to match the requirements of different annotation projects.

BioGraphia serves not only as a practical solution for pathway construction but also as a valuable resource for evaluating and refining knowledge graph construction methods. By providing high-quality benchmark annotations, BioGraphia facilitates the development of more accurate and comprehensive pathway extraction techniques, thereby supporting the creation of richer and more reliable biological knowledge graphs. Ultimately, this contributes to a deeper understanding of molecular mechanisms and accelerates the discovery of novel therapeutic interventions.

BioGraphia is released under the Apache 2.0 license, ensuring that the tool can be freely used and extended for future research and annotation projects without restriction.

In the following sections, we review related work in Section 2. Section 3 states the definition of Pathway Graph Annotation. Section 4 describes the system architecture and the functionality of each component and Section 5 reports results from our evaluation and presents example use cases. Finally, Section 6 concludes the paper and discusses future directions.

2 Related Work

Several annotation platforms have been developed for distributed annotation tasks, among which WebAnno (Yimam et al., 2013) and Inception (Klie et al., 2018) are the most widely used open-source solutions. However, both platforms are primarily tailored for linguistic annotation tasks and are limited in handling the complexity of biological pathway extraction. Specifically, their text span layer based annotation methods require annotators to spend significant time identifying relevant entities within text documents, making them inefficient for complex, graph-based biological annotation tasks.

GraphAnno (Gast et al., 2015) is an open-source annotation tool explicitly designed for knowledge graph construction. Nonetheless, GraphAnno's command-line interface poses significant usability challenges, requiring considerable learning effort from annotators and thus restricting its scalability for large-scale annotation efforts. Recent advancements in LLMs have inspired novel approaches to annotation, such as ITAKE (Song et al.,

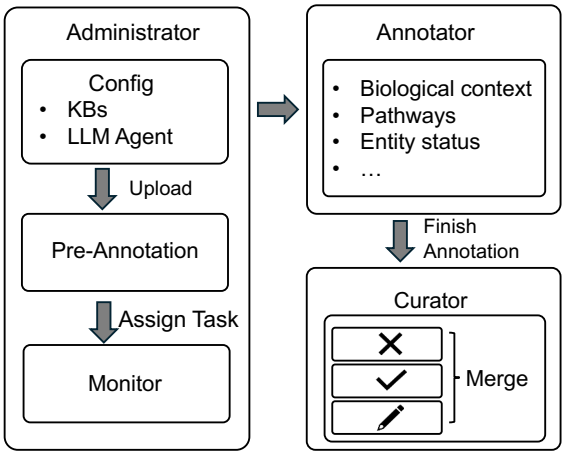


Figure 2: Workflow overview illustrating roles and responsibilities of the administrator, annotator, and curator in BioGraphia. The administrator configures the knowledge base and pre-annotation settings for the LLM module, uploads the materials to be annotated, and initiates pre-annotation. Afterward, the administrator assigns annotation tasks to annotators and monitors their progress. Once annotators complete their work, the curator reviews and merges annotations from different annotators.

2024), which integrates LLM module and ModelOps into the annotation workflow to enhance annotation quality and efficiency. However, ITAKE focuses solely on traditional NLP annotation tasks such as named entity recognition and focus on supporting conventional neural models and LLMs with basic prompting strategies, which are insufficient for complex biological pathway annotation requirements. Additionally, ITAKE is not designed for graph-based annotation workflows and is not yet available as an open-source platform. Among these tools, BioGraphia represents a modern, open-source graph annotation platform that supports multi-user collaborative environments with role-based access control, real-time collaboration features, and comprehensive project management capabilities for large-scale biological pathway annotation projects.

3 Task Definition

We formalize the task of Pathway Graph Annotation as the extraction and representation of structured biological pathway knowledge from unstructured text sources, such as scientific papers, laboratory experiment reports, and other scientific documentation. Formally, given a textual document D, the goal is to extract a pathway graph $G_D = (C, N, E)$.

Biological Context (C) defines the context in which the pathway is described. Specifically, the context C is represented as a tuple (c_{name}, c_{type}), where:

- c_{name} explicitly identifies the biological context (e.g., a specific disease or cell type) if provided.
- c_{type} specifies the category of the context.[1]

Nodes (N) represent entities identified from document D. Each node $n \in N$ is characterized by:

- t_n: the category of the node, selected from a predefined taxonomy[2].
- s_n: the context-specific status of the entity, explicitly stated in document D.

Edges (E) are directed relations representing explicit regulatory interactions between nodes, extracted from D. Each edge $e \in E$ is a tuple $(n_{head}, n_{tail}, r, M)$, where:

- $n_{head}, n_{tail} \in N$ are the source and target nodes of the relation.
- r is a label describing the nature of relation.
- M is an optional set of meta-relations with each meta-relation structured as $(r_{meta}, n_{metaTarget})$, where r_{meta} describes the type of secondary effect and $n_{metaTarget}$ is the target node of the secondary effect. Meta-relations are restricted to represent a molecular event where a primary relation between two nodes has indirect or secondary effects to another node.

4 BioGraphia System Overview

BioGraphia incorporates three distinct user roles: Administrator, Annotator, and Curator. The overall workflow begins with administrators configuring project parameters, proceeds with annotators performing the detailed annotation tasks, and concludes with curators reviewing and finalizing annotations. An overview of this workflow is presented in Figure 2. Each component is elaborated upon in the following subsections.

[1]The set of context types includes Disease, Cell/Tissue, Pathological/Environmental Stress, Undefined/Generic.

[2]Gene, Protein, Biological Process, Chemical Compound, RNA Modification, Medicine or Drug, Disease, Clinical Phenotype

Figure 3: Administrator project management interface. Panel (A) shows how the administrator can monitor the progress of annotation across tasks. Panel (B) displays the interface for assigning annotation tasks to different annotators; each annotator can only view the tasks assigned to them. The administrator can also configure the knowledge base and pre-annotation LLM module settings.

4.1 Project Management

Administrators are responsible for initializing annotation projects. Initially, administrators configure the pre-annotation LLM module and integrate structured knowledge bases, which are imported in JSON format. The administrator assigns each knowledge base to specific annotation types, enabling annotators to access relevant contextual information during their tasks. Additionally, administrators can import scientific papers that need to be annotate individually or in batches for pre-annotation processing. Once our LLM module complete the pre-annotation process, administrators assign annotation tasks to individual annotators as shown in Figure 3.

Annotators then gain exclusive access to the assigned papers through their user-specific dashboards. Administrators also maintain oversight through real-time monitoring capabilities, tracking annotation progress and assessing task completion status to ensure efficiency and productivity.

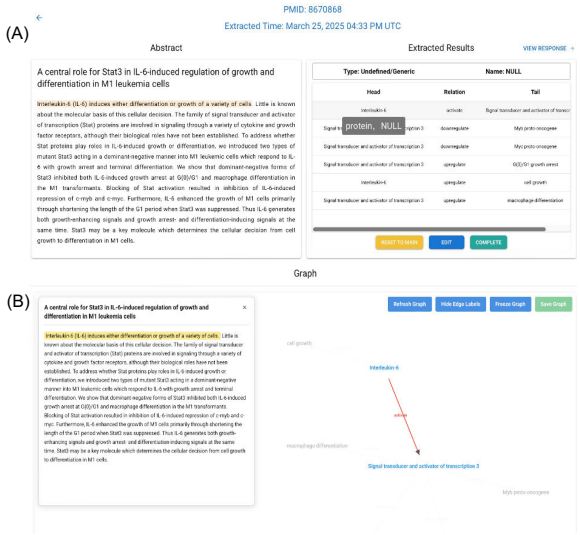


Figure 4: Annotator workspace in BioGraphia. Panel (A) shows the structured table view, while panel (B) presents the interactive pathway graph. Annotators can annotate nodes, relations, meta-relations, and biological contexts either through the graph interface or the table view. Clicking a relation highlights its supporting textual evidence. Hovering over a node displays its detailed attributes. Annotations are guided by predefined biological knowledge bases.

4.2 Pre-annotated by Pathway LLM

The Pathway LLM employs carefully designed Chain-of-Thought (CoT) few-shot prompting (Wei et al., 2023) crafted by domain experts. These prompts clearly outline annotation criteria and provide illustrative examples detailing intermediate reasoning steps necessary for accurate annotation. Upon generating preliminary annotations, the Pathway LLM module parses the output to produce structured JSON files. Subsequently, these files are processed through our span-alignment tool, which identifies corresponding textual spans from the source documents, ensuring alignment between annotations and textual evidence. Additionally, the system integrates a graph visualization tool, producing interactive span-aligned pathway graphs. The modular architecture of the Pathway LLM module enables customization, allowing users to tailor the LLM module's configuration to suit project specific annotation requirements.

4.3 Multi-user Annotation

Annotators access their tasks through a dedicated workspace as shown in Figure 4. Upon selecting a specific paper, annotators review the detailed pre-annotations provided by the Pathway LLM module. Pre-annotations include comprehensive biological

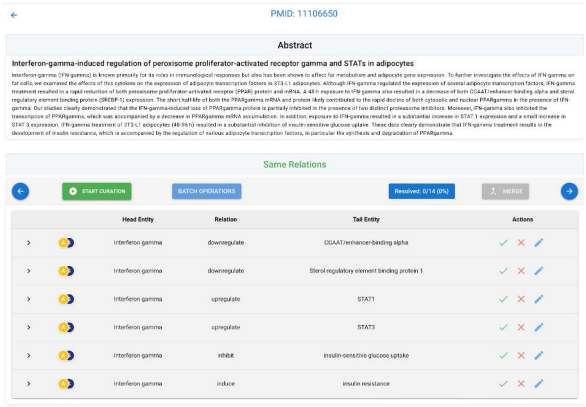


Figure 5: Curation interface of BioGraphia. Curators review completed annotations, resolve conflicts among annotators, and finalize pathway annotations.

context, entity names, node types, and context-specific statuses, assuming an initial positive state unless explicitly stated otherwise.

Annotators can interact directly with the graphical representation of the annotations; clicking on graph relations highlights corresponding textual evidence supporting the annotation. Annotators may refine annotations by editing node labels, relations, supporting text, and by introducing additional relations, including complex meta-relations. Annotators can also annotate biological contexts strictly based on predefined knowledge bases, ensuring consistency and accuracy. We also have table view for annotation.

Moreover, annotators can document comments and annotations-specific notes, visible to both administrators and other annotators, so that annotator and administrator can collaborative resolve potential issues and continuous improvement. Upon completion of their tasks, annotators submit their annotations for subsequent curation, initiating the final review stage performed by curators.

4.4 Curation

We utilize an approach analogous to version control merging, as shown in Figure 5 systematically identifying annotations that have unanimous agreement as well as annotations where discrepancies exist between annotators. Relations agreed upon by all annotators are automatically highlighted, while conflicting annotations are clearly flagged. Curators then carefully review these flagged annotations, deciding to accept, reject, or modify them to ensure accuracy and coherence. To maintain robust quality control, the curation phase is initiated only after an annotation instance has been independently completed by at least two annotators.

4.5 Ensuring system and data security

System security BioGraphia incorporates a number of comprehensive security measures to protect sensitive data and ensure a safe collaboration environment for users. We implement role-based access control that specifies permissions between administrators, annotators, and curators. This ensures that users can only modify and access data appropriate for their assigned roles. We also incorporate user authentication through bcrypt and session management through JSON Web Tokens (JWT), maintaining secure authentication states throughout user interactions within the platform. To prevent malicious inputs, such as injection attacks, BioGraphia incorporates data validation for all user-submitted content.

5 Evaluation

We evaluated BioGraphia by comparing its performance against traditional manual annotation methods and the widely used annotation platform, INCEpTION, using a controlled annotation experiment.

5.1 Experimental Setup

We selected 160 abstracts related to biological pathways for the annotation study. We recruited 15 annotators with biomedical backgrounds from Carnegie Mellon University and the University of Pittsburgh. Annotators were compensated for their participation, and no conflicts of interest existed between annotators and authors.

Annotators received two warm-up examples with feedback to familiarize themselves with each annotation method. Subsequently, each annotator was randomly assigned to use the three annotation methods—Manual annotation, INCEpTION, and BioGraphia—in a randomized order to minimize learning and ordering effects. For each annotation method, annotators were given four abstracts to annotate.

To ensure consistent task complexity across annotation tools and annotators, abstracts were assigned based on the Corrected Type-Token Ratio (CTTR) (Guiraud, 1954) score, a quantitative measure reflecting text complexity. We use lexicalrichness [3] for CTTR score calculation. We employed

[3] https://github.com/LSYS/lexicalrichness

stratified folding to evenly distribute the complexity of the abstracts, ensuring a balanced and fair comparison among methods.

To streamline the annotation process and maintain quality control, we developed an experimental website featuring annotation training materials and built-in quality checks. A sample of the website can be found in https://t23qu3znup3a.share.zrok.io/#/register.

5.2 Efficiency and Cognitive Load

We evaluated annotation efficiency by measuring the average task completion time across all annotators for each annotation method. BioGraphia achieved an average completion time of 21 minutes, whereas manual annotation took an average of 38 minutes, and INCEpTION took 49 minutes. This significant reduction in completion time indicates that BioGraphia substantially improves annotation efficiency.

Additionally, we assessed cognitive load using the NASA Task Load Index (NASA-TLX)(Hart, 2006), a widely recognized subjective workload assessment tool. The NASA-TLX evaluates various dimensions of perceived workload, including mental demand, physical demand, temporal demand, performance, effort, and frustration levels, providing a weighted overall workload score.

BioGraphia received a NASA-TLX weighted workload score of 23, indicating a low cognitive demand. In contrast, manual annotation and INCEpTION scored significantly higher, with scores of 63 and 72 respectively, both indicating moderate to high cognitive demands. This result demonstrates that tools designed for traditional NLP task annotation and manual methods significantly increase mental stress when applied to complex biological annotation tasks, whereas BioGraphia maintains a notably lower workload.

Furthermore, BioGraphia exhibited low variance in task completion times, with a standard deviation of only 3.3 minutes (Coefficient of Variation = 14.35%). This consistency further supports BioGraphia's effectiveness in providing a stable and predictable annotation experience.

BioGraphia enhances annotation efficiency and reduces cognitive load compared to traditional manual annotation and existing annotation platforms like INCEpTION. The platform's intuitive design and interactive annotation interface effectively support annotators, making complex biological pathway annotation tasks more manageable and consistent.

6 Conclusion and future work

In this paper, we presented BioGraphia, a web-based annotation platform tailored specifically for distributed pathway graph annotation tasks. BioGraphia integrates collaborative functionalities with real-time monitoring, interactive graph visualization, and LLM-assisted Explainable Span-aligned Pre-Annotation powered by large language models. These features show improvement in annotation efficiency and quality, addressing key limitations found in existing platforms. Its flexible modular architecture further allows seamless integration of external knowledge graphs and customization of annotation schemas, broadening its applicability across various biological annotation scenarios.

Future work will focus on further improving BioGraphia by systematically collecting and incorporating annotator feedback. We plan to enhance the user interface design and provide more comprehensive instructions and guidelines, making the platform more intuitive and scalable for large-scale annotation tasks. Additional efforts will be directed toward refining the design to facilitate annotators in effectively updating aligned text spans. We will also enhance the monitoring system to offer administrators improved oversight capabilities. Finally, we aim to build and maintain a high-quality, continuously updated dataset for biological pathways, incorporating advanced features for dataset management, visualization, inference support, and ongoing updates.

References

James Allen, Omid Bahkshandeh, William De Beaumont, Lucian Galescu, and Choh Man Teng. 2018. Effective broad-coverage deep parsing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

John A Bachman, Benjamin M Gyori, and Peter K Sorger. 2023. Automated assembly of molecular mechanisms at scale from text mining and curated databases. *Molecular Systems Biology*, 19(5):e11325.

Volker Gast, Lennart Bierkandt, and Christoph Rzymski. 2015. Creating and retrieving tense and aspect annotation with GraphAnno, a lightweight tool for multi-level annotation. In *Proceedings of the 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-11)*, London, UK. Association for Computational Linguistics.

Pierre Guiraud. 1954. Les caractères statistiques du vocabulaire: essai de méthodologie. *(No Title)*.

Sandra G Hart. 2006. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pages 904–908. Sage publications Sage CA: Los Angeles, CA.

M. Kanehisa. 2000. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30.

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, Santa Fe, New Mexico. Association for Computational Linguistics.

Marija Milacic, Deidre Beavers, Patrick Conley, Chuqiao Gong, Marc Gillespie, Johannes Griss, Robin Haw, Bijay Jassal, Lisa Matthews, Bruce May, Robert Petryszak, Eliot Ragueneau, Karen Rothfels, Cristoffer Sevilla, Veronica Shamovsky, Ralf Stephan, Krishna Tiwari, Thawfeek Varusai, Joel Weiser, Adam Wright, Guanming Wu, Lincoln Stein, Henning Hermjakob, and Peter D'Eustachio. 2023. The reactome pathway knowledgebase 2024. *Nucleic Acids Research*, 52(D1):D672–D678.

Thomas C. Rindflesch and Marcelo Fiszman. 2003. The interaction of domain knowledge and linguistic structure in natural language processing: interpreting hypernymic propositions in biomedical text. *Journal of Biomedical Informatics*, 36(6):462–477.

Jiahe Song, Hongxin Ding, Zhiyuan Wang, Yongxin Xu, Yasha Wang, and Junfeng Zhao. 2024. ITAKE: Interactive unstructured text annotation and knowledge extraction system with LLMs and ModelOps. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 326–334, Bangkok, Thailand. Association for Computational Linguistics.

D. N. Sosa and R. B. Altman. 2022. Contexts and contradictions: a roadmap for computational drug repurposing with knowledge inference. *Brief Bioinform*, 23(4).

Marco A Valenzuela-Escárcega, Özgün Babur, Gus Hahn-Powell, Dane Bell, Thomas Hicks, Enrique Noriega-Atala, Xia Wang, Mihai Surdeanu, Emek Demir, and Clayton T Morrison. 2018. Large-scale automated machine reading discovers new cancer-driving mechanisms. *Database*, 2018:bay098.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

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.

SynthTextEval: Synthetic Text Data Generation and Evaluation for High-Stakes Domains

Krithika Ramesh[♠] **Daniel Smolyak**[♢] **Zihao Zhao**[♠]
Nupoor Gandhi[♣] **Ritu Agarwal**[♠] **Margrét Bjarnadóttir**[♢] **Anjalie Field**[♠]
[♠]Johns Hopkins University
[♢]University of Maryland, College Park [♣]Carnegie Mellon University
{kramesh3, zzhao71, ritu.agarwal, anjalief}@jhu.edu
{dsmolyak, mbjarnad}@umd.edu
nmgandhi@cs.cmu.edu

Abstract

We present SYNTHTEXTEVAL, a toolkit for conducting comprehensive evaluations of synthetic text. The fluency of large language model (LLM) outputs has made synthetic text potentially viable for numerous applications, such as reducing the risks of privacy violations in the development and deployment of AI systems in high-stakes domains. Realizing this potential, however, requires principled consistent evaluations of synthetic data across multiple dimensions: its utility in downstream systems, the fairness of these systems, the risk of privacy leakage, general distributional differences from the source text, and qualitative feedback from domain experts. SYNTHTEXTEVAL allows users to conduct evaluations along all of these dimensions over synthetic data that they upload or generate using the toolkit's generation module. While our toolkit can be run over any data, we highlight its functionality and effectiveness over datasets from two high-stakes domains: healthcare and law. By consolidating and standardizing evaluation metrics, we aim to improve the viability of synthetic text, and in turn, privacy-preservation in AI development.

1 Introduction

Currently, in contexts involving sensitive data like healthcare and social services, accessing domain-specific text to develop AI systems and assistive tools necessitates strict data-sharing agreements and well-defined protocols to reduce privacy risks and maintain compliance with regulatory standards (de Kok et al., 2023; Alberto et al., 2023). Even with careful protocols, privacy-preservation is a key challenge, particularly if systems that have been developed using sensitive data are accessible to the public. While text anonymization can be used to redact or replace private identifiers from text, there is still significant risk of sensitive attributes in the data being de-identified or exposed (Staab et al., 2024; Xin et al., 2024; Pang et al., 2024).

Synthetic data offers a possible alternative means of privacy-preservation. If synthetic text is sufficiently reflective of real text, it could facilitate the development of tools or conducting analyses, while minimizing the risk of a privacy breach. Beyond privacy-preservation, there has been a substantial amount of interest in other synthetic text applications, including fine-tuning task-specific models, serving as a proxy for human evaluations and auditing models (Mitra et al., 2023; Xu et al., 2023; Tan et al., 2024; Huang et al., 2023; Min et al., 2023). Although approaches for generating synthetic text may differ depending on the intended use-case, they typically share the same underlying objective - to produce diverse, high-quality data with patterns and signals that mimic the characteristics of real-world data distributions (Wu et al., 2024; Liu et al., 2024).

Despite substantial recent interest in privacy-preserving text generation (Yue et al., 2023; Mattern et al., 2022; Wu et al., 2023; Tang et al., 2024; Nahid and Hasan, 2024), evaluation criteria vary across studies, leading to inconsistencies in conclusions and results that overlook key aspects of data, such as fluency and privacy (Ramesh et al., 2024). Existing frameworks for assessing synthetic text quality either lack a wide range of criteria and metrics or do not provide easy-to-use tools to evaluate one's own data (Belgodere et al., 2024; Chim et al., 2024). Given the importance of synthetic data generation and its potential use in high-stakes domains, there is a critical need for a thorough and consistent evaluation approach that supports comparability across studies. To this end, we present SYNTHTEXTEVAL: an open-source toolkit[1] for evaluating synthetic text and comparing generation methods. An overview of the architecture is provided in Figure 1. While generalizable to many syn-

[1]GitHub Repository: https://github.com/kr-ramesh/synthtexteval

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 487–499
November 4-9, 2025

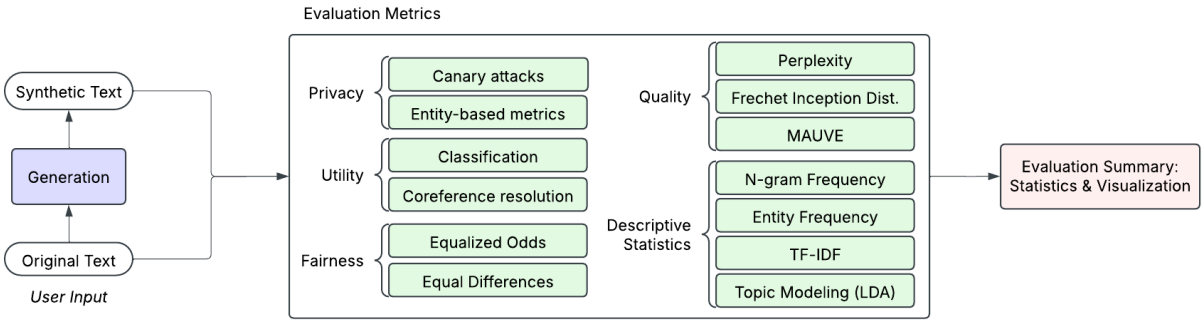


Figure 1: Architecture overview of SYNTHTEXTEVAL.

thetic text use cases, our toolkit particularly aims to enable the promising potential synthetic text has for facilitating privacy-preserving data sharing in high-stakes domains.

SYNTHTEXTEVAL has three core features that support this endeavor. First, it provides functionality to generate synthetic text with differentially private guarantees. Second, it implements a principled suite of generalizable metrics for evaluating the utility, privacy, fairness and quality of synthetic text, along with a descriptive outline of the data distribution. Third, a user-friendly GUI enables manually comparing synthetic text with real text, supporting the solicitation of qualitative feedback from non-technical domain experts. We further provide built-in support for generating and evaluating synthetic data in two domains with sensitive data: law (Pilán et al., 2022a) and healthcare (Johnson et al., 2016), but our toolkit is crucially data-agnostic, allowing users to run our generalizable metrics over their own datasets. Although our work is motivated by a clear need for standardized evaluation practices in high-stakes domains, it is broadly applicable to synthetic data for any task. By providing standardized data-agnostic metrics for evaluating synthetic data quality, we aim to assist researchers and practitioners in developing more robust and trustworthy AI systems and tools.

2 Background

Several factors influence the viability of synthetic data, including the faithfulness of the data to the source distribution, the diversity of the generated text, and its usability for developing downstream applications (Long et al., 2024). Prior work has often used evaluation criteria targeted to specific use cases, which make synthetic data generation methods difficult to compare. For instance, in generating synthetic text for privacy preservation, studies report metrics targeting training data leakage (Yue et al., 2023), but do not assess if the synthetic data introduces unfairness. In contrast, a system that aims to use synthetic data to address data imbalance may report results from groupwise performance metrics, but not privacy leakage (Pereira et al., 2024). Although not every metric is relevant for every use case, many of them do intersect: synthetic data is not a viable solution for privacy preservation if it introduces unfairness.

Even when the specified criterion is shared (e.g., privacy), studies often use different evaluation metrics , which can be contradictory and preclude comparability of methods (Friedler et al., 2021). Motivated by the need for consistent evaluation, our framework implements evaluation metrics that have been commonly used in prior work, but lack standardization (e.g., classification, canary attacks), as well as introduces metrics that have been shown to identify weakness in synthetic data (e.g., coreference resolution, fairness criteria) but are rarely reported (Ramesh et al., 2024).

2.1 Synthetic Data Evaluation Frameworks

Our toolkit differs from existing frameworks for evaluation of synthetic text in its wide range of criteria and metrics and provision of easy-to-use tools to evaluate one's own synthetic text data. Belgodere et al. (2024) provides a framework for auditing synthetic data across modalities, including tabular, time series, and text, and across fidelity, privacy, utility, and fairness metrics. However, due to their broad focus across modalities, their metrics are less tailored to text data, basing the evaluation on embeddings only. Chim et al. (2024) provides a text-specific auditing framework, focusing on user-generated text in messaging and social media contexts. They include metrics for downstream utility, privacy, and text fidelity, but do not include

	Evaluation Metrics					Ease-of-Use		Modality
	Utility	Privacy	Fairness	Quality	Descriptive	Code Package	GUI	Natural Language
SYNTHTEXTEVAL	✓	✓	✓	✓	✓	✓	✓	✓
Belgodere et al. (2024)	✓	✓	✓	✓	✓	✗	✗	✓
Chim et al. (2024)	✓	✓	✗	✓	✗	✗	✗	✓
Gehrmann et al. (2021)	✓	✗	✗	✓	✗	✓	✓	✓
Patki et al. (2016)	✓	✓	✓	✓	✓	✓	✓	✗
Qian et al. (2023)	✓	✓	✓	✓	✓	✓	✓	✗

Table 1: Comparison of evaluation frameworks for private synthetic text data generation.

descriptive statistics or fairness metrics. Neither framework provides a user-friendly code package nor a GUI to support qualitative evaluations.

In addition to synthetic text evaluations frameworks, other benchmarks, such as Gehrmann et al. (2021), focus on model capabilities (e.g., developed on synthetic data) in the context of natural language generation tasks, but are less focused on metrics specific to the generated text. Complementary to our work, Patel et al. (2024) developed the DataDreamer tool to standardize workflows for generating synthetic text data with LLMs, providing functionality for both finetuning generative models and training downstream models on generated synthetic data. However, they do not provide any functionality to assess the generated synthetic data. Patki et al. (2016) and Qian et al. (2023) both provide comprehensive evaluation frameworks and code packages for synthetic tabular data, but do not support evaluation of synthetic text.

3 Design and Implementation

SYNTHTEXTEVAL contains a text generation module and a set of evaluation modules, for each category of evaluation metrics, and a GUI as shown in Figure 1. Modules can be run individually to focus on specific metrics or together for a full audit.

The text generation model includes optional privacy-preserving guarantees for users who require them. It offers functionality for a descriptive analysis of any corpora, along with filtering functionality to remove low-quality synthetic text samples. In the evaluation module, the package includes tools to train and test downstream models on synthetic data, as well as automated evaluation of text quality, privacy and fairness metrics. The complete set of evaluation metrics are further explained in the following subsections.

3.1 GUI

Although we implement comprehensive and generalizable evaluation metrics, fully automated metrics can miss more nuanced aspects of synthetic text quality, especially ones that may vary across domains. For example, leakage of a name of a judge in a legal text may be acceptable, whereas leakage of a patient name in clinical text is not. To account for this, our framework contains a GUI that provides two key functionalities: i) for a given synthetic text, it displays the most real texts (determined through embedding similarity) ii) for a given synthetic text that contains a named entity, it displays real texts containing the entity. The GUI further supports writing free-form comments on notes, which can be saved and shared. The GUI is shown in Figure 2.

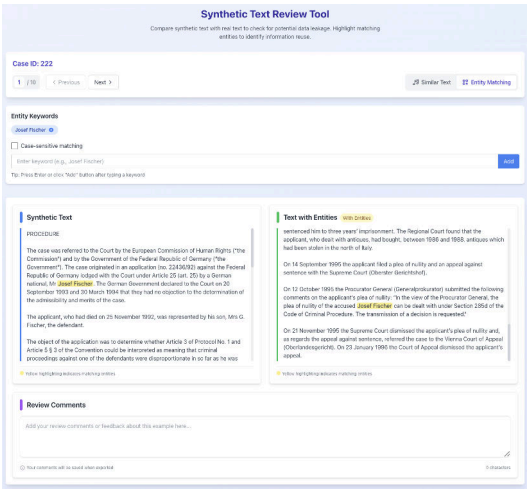


Figure 2: Our visual interface supporting exploration, comparison, and annotation of synthetic and real text.

This functionality in a user-friendly display allows non-technical domain experts to provide qualitative feedback on synthesized text, such as if synthesized text is factually accurate to real text and if it violates privacy by leaking information about entities.

3.2 Installation and Setup

SYNTHTEXTEVAL is an installable Python package[2], and the repository contains a number of user

[2] https://pypi.org/project/SynthTextEval/

resources including detailed setup instructions, information about various components, as well as test functions for each of the package's modules. In addition, we provide a demo Jupyter notebook and runnable experiment scripts for our case studies. A web-based version of our GUI is available for interactive exploration[3], and installation instructions for the desktop application are provided in the GitHub repository.

3.3 Downstream Utility

We assess the utility of synthetic data as a substitute for training models on real data with built-in support for two tasks: classification and coreference resolution. We select these two tasks both for their popularity in conducting evaluations (Table 1) driven by potential real-word use cases (Gandhi et al., 2023), and because the characteristics of the synthetic data that would increase its utility differs between the two tasks. For example, if the synthetic data successfully trains a high-performing classification model, this would suggest it contains a comprehensive set of features with similar frequency and contextual usage to the real data. However, lack of coherence or logical inconsistencies across synthesized documents may not be reflected in classification performance. In contrast, coreference resolution benefits from entity consistency, coherent structure, and the preservation of long-range dependencies.

In practice, a researcher or practitioner might use synthetic text as training data for these tasks by hiring annotators to label it, but collecting these "gold" annotations is too time-consuming and expensive to evaluate synthetic data quality across generation methods. Instead, for both tasks, we offer the functionality to generate "silver" annotations for the synthetic data by using an external pretrained model. New classification and coreference models can then be trained on this silver-annotated data, with the models' performance evaluated on a (real) gold standard test set. For classification, we focus on both F1 score and accuracy and for coreference resolution we focus on F1 score.

3.3.1 Fairness

SYNTHTEXTEVAL provides functionality to evaluate fairness in models trained on synthetic data. In domains such as healthcare and social services, the development of equitable systems is essential (Meng et al., 2022; Field et al., 2023). Prior research has identified differences in text data associated with difference groups, such as disparate rates of stigmatizing language in clinical notes (Harrigian et al., 2023), and examples where synthetic data increases downstream performance more for certain groups than others (Bhanot et al., 2021). Thus, for the tasks where we conduct utility evaluations, we additionally assess whether the distribution of utility is even across designated subgroups. We implement two commonly used fairness metrics: equalized odds (EO) and equality difference (ED). We provide full formulas in Appendix C.

3.4 Privacy

Our toolkit supports privacy evaluations for language models using both canary-based evaluations (Carlini et al., 2019) and entity-centric metrics (Ramesh et al., 2024). Out of proposed methods to quantify privacy leakage and memorization in LLMs (Carlini et al., 2022; Schwarzschild et al., 2024; Kim et al., 2023; Huang et al., 2024; Li et al., 2024), we focus specifically on measures that estimate the frequencies or likelihood of generation of spans of sensitive information and personal identifiable information (PII) in the synthetic text. These methods evaluate privacy risks in generated text, rather than vulnerability of models to attack (e.g., membership inference attacks), making them more suitable for evaluating synthetic text and also generalizable to diverse paradigms of privacy-preserving text generation, like sanitization approaches (Chen et al., 2023).

Canary-based evaluations measure memorization through the generation likelihood of injected canary phrases. While they offer a controlled way to assess data leakage (Yue et al., 2023), they are not always reflective of actual PII leakage (Ramesh et al., 2024). Thus, we additionally implement entity-centric evaluation criteria, which estimate the frequency of sensitive entities and their contexts in synthetic text. Appendix B provides additional details.

3.5 Text Quality

In order to evaluate the overall synthetic text quality, independently from specific downstream utility tasks, our toolkit implements a range of measures, including general perplexity-based evaluations as well as the reference-based metrics Fréchet Inception Distance (FID) (Heusel et al., 2017) and MAUVE (Pillutla et al., 2021). These metrics have

[3]Online interface: https://syntheticreview.cdhai.com/

been shown to often align with human judgment and can serve as valuable indicators of text quality, especially when human evaluation is impractical due to resource constraints. While these metrics have known 'blind spots' (He et al., 2023), using a combination of automated qualitative metrics can offer useful insight into the distributional similarity between synthetic and real data.

FID calculates the feature-wise mean and covariance matrices of embeddings for both synthetic and real text, and the Fréchet distance between these matrices serves as a measure of their similarity. MAUVE is a KL-divergence-based metric that compares the distributions of the synthetic and real text in the embedding space of a selected LLM. MAUVE's strength lies in its usefulness for relative comparisons regardless of embedding model choice, as it is robust to scaling, quantization, and embedding choices, while also correlating strongly with human judgment compared to other automated metrics. Perplexity assesses text quality by measuring the likelihood that a provided base LLM would generate a given text segment. A higher perplexity score indicates greater semantic coherence (Colla et al., 2022). Further details on how each metric is calculated are in Appendix D.

3.6 Descriptive Statistics

Users can conduct an exploratory text analysis of the synthetic data and make comparisons with real data by using the descriptive statistics module. Some of the functions included are an n-gram frequency analysis to detect common collocations and phrase patterns, TF-IDF computations to assess the relative importance of words across documents and an analysis of the most and least frequent entities in the text distributions. Additionally, to extract underlying thematic structures, the package employs Latent Dirichlet Allocation (LDA) for topic modeling, showing the user dominant topics and their associated keywords.

3.7 Generating synthetic text

Although our main focus is on facilitating evaluation, we additionally implement a popular synthetic data generation method, which involves fine-tuning an LLM with control codes, with or without differential privacy (Keskar et al., 2019; Yue et al., 2023; Ramesh et al., 2024). This functionality allows users to conduct an initial assessment of synthetic data quality in their own domains, as well as provides a baseline to compare against more customized generation methods.

Control codes are a special type of prefix in the input to the language model that contain labels corresponding to the features associated with the desired synthetic text (Keskar et al., 2019). These control codes are prepended to the prompt during training, allowing the model to learn the relationship between the labels and the corresponding output distributions of the synthesized text. We provide example control codes in the appendix E. During inference, users can supply their own control codes to the model, guiding it to generate text that matches the characteristics they desire for the intended use of the synthetic data.

We incorporate support for differentially private (DP) fine-tuning of models to generate synthetic text with privacy guarantees. DP techniques safeguard the participants in a dataset by making it difficult to infer whether any single individual's data was used. This is done by injecting carefully calibrated random noise to the results of any analysis, so that the outcome is nearly the same whether or not a given data point is included. For synthetic text generation, we achieve these privacy guarantees by fine-tuning our models with DP-SGD (Differentially Private Stochastic Gradient Descent) (Abadi et al., 2016). In DP-SGD, model gradients are clipped, and random noise is added during training, which limits the impact of any single data point and helps prevent the model from leaking sensitive information in its outputs. Further details about how DP works and its implementation in our study are provided in Appendix A.

While DP-based approaches offer well-defined upper bounds on the likelihood privacy leakage, this comes at the expense of model utility. Nevertheless, DP-generated synthetic text has been shown to be useful in reducing privacy risks while maintaining utility in certain downstream settings (Ramesh et al., 2024; Yue et al., 2023).

4 Validation and Example Usage

4.1 Experimental setup and objectives

Our first case study uses the Text Anonymization Benchmark (TAB), a dataset of 1,268 European court cases (Pilán et al., 2022b) extensively annotated to support evaluating text anonymization methods. We generate two synthetic datasets: one without DP ($\epsilon = \infty$) and one with DP ($\epsilon = 8$). We use the country and year of the court case as control codes for generation. Thus, our artificially created

classification task is to predict the country for each court case. A BERT$_{base}$ model is finetuned on each of the datasets. Our privacy evaluation focuses on the reoccurrence of personal information entities from the original dataset in the synthetic datasets.

Our second case study uses clinical notes from MIMIC-III (Johnson et al., 2016) and coreference-annotated clinical notes from the i2b2/VA Shared-Task (Uzuner et al., 2012). For the MIMIC-III data, we use the 10 most frequent International Classification of Diseases (ICD-9) codes as the control codes to generate synthetic data (similar to Al Aziz et al. (2021)), also with $\epsilon = \infty$ and $\epsilon = 8$.

To measure downstream utility, we evaluate performance on the multilabel and multiclass ICD-9 code prediction task for MIMIC-III, and the coreference resolution task for the i2b2/VA dataset. We compare performance for a BERT$_{base}$ model finetuned on only the original dataset to a model finetuned on the synthetic datasets. The fairness metrics for the ICD-9 code prediction task are calculated for both patient race and patient gender on the real clinical notes, while the entity-based privacy evaluations focus on the regurgitation of PII from the real data and the contexts in which they appear.

4.2 Results

We report classification utility for both datasets in Table 2 and remaining results in Appendix F. In general, results follow the trends we expect, where real text and non-DP synthetic text have the best utility and quality, but DP synthetic text has the best privacy. These results validate our toolkit's ability to reflect these properties.

4.2.1 TAB

Descriptive statistics show the synthetic data for TAB is on average about half as long and has half as many unique words (Table 3), although this is influenced by the user's choice of hyperparameters in generating the data. The synthetic data without DP ($D_{\epsilon=\infty}$) has higher Jaccard Similarity and Cosine Similarity to the real dataset compared to the synthetic data with DP ($D_{\epsilon=8}$). Regarding quality, both synthetic datasets are on par with the real test set for FID, but are lower quality according to the MAUVE and perplexity metrics (Table 4).

Utility results (Table 2) show the downstream model trained on $D_{\epsilon=\infty}$ has higher F1 and higher accuracy than the one trained on $D_{\epsilon=8}$. However, $D_{\epsilon=\infty}$ has a higher percent of PII leaked from the real data, indicating a higher privacy risk (Table 5). There is relatively high privacy risks for both synthetic datasets, likely due to the small size of the original TAB dataset. The $D_{\epsilon=8}$ dataset trades off slightly better privacy metrics for lower quality and utility compared to $D_{\epsilon=\infty}$.

		F1 Score	Accuracy
TAB	$D_{\epsilon=\infty}$	0.96 ± 0.035	0.99 ± 0.012
	$D_{\epsilon=8}$	0.54 ± 0.0 ↓-0.42	0.95 ± 0.0 ↓-0.04
Healthcare	D_{real}	0.67 ± 0.012	0.32 ± 0.016
	$D_{\epsilon=\infty}$	0.61 ± 0.003 ↓-0.06	0.26 ± 0.004 ↓-0.06
	$D_{\epsilon=8}$	0.40 ± 0.004 ↓-0.27	0.18 ± 0.005 ↓-0.14

Table 2: **TAB**: Difference in performance between models trained on data generated with DP and models trained on data generated without DP over TAB classification. **Healthcare**: difference in performance between real and synthetic data as training data for the ICD-9 classification task.

4.2.2 Healthcare

For both coreference resolution and mention detection, training on either synthetic dataset results in significantly lower utility compared to the real data, with $D_{\epsilon=\infty}$ and $D_{\epsilon=8}$ providing nearly identical performance (Table 6). In contrast, for the ICD-9 classification task (Table 2), there is a more pronounced drop in utility when using the differentially private synthetic data ($D_{\epsilon=8}$), while $D_{\epsilon=\infty}$ achieves higher performance than $D_{\epsilon=8}$ but still does not reach the performance of the model trained on the original dataset.

Fairness metrics (Table 7) suggest the model trained on the real data is most fair for race as the sensitive attribute, with the lowest FNED and EO metrics. $D_{\epsilon=\infty}$ is second highest, and $D_{\epsilon=8}$ has the worst fairness. For gender as the sensitive attribute, the models are all approximately equally fair across training datasets. On the privacy criteria, $D_{\epsilon=\infty}$ includes a higher percent of reproduced entities compared to $D_{\epsilon=8}$. Overall, our clinical notes evaluation shows that generating data without DP provides higher downstream utility and fairer downstream models at the cost of slightly higher privacy risks, compared to synthetic clinical notes generated with DP.

Conclusion SYNTHTEXTEVAL provides a toolkit to comprehensively evaluate synthetic text data across a wide range of metrics. This toolkit allows users the opportunity to assess the relative merits of synthetic data generation approaches, through the standardization of evaluation metrics, allowing for greater confidence in synthetic data

quality, especially in high-stakes domains where quality synthetic data is in high demand.

5 Ethics Statement

SYNTHTEXTEVAL is designed to lower barriers to development of AI systems in sensitive domains through comprehensive synthetic text data evaluation. Rather than rely on ad-hoc evaluation and comparison of synthetic text data, we hope our toolkit will help standardize comparisons and encourage responsible synthetic text data generation. In particular, our fairness and privacy metrics emphasize that high performance on utility and quality metrics are not on their own sufficient criteria for determination of worthiness for synthetic data publication and distribution.

However, while we have aimed to include wide range of metrics across many criteria, we caution that other context-dependent assessment may be necessary.An unintended consequence of this package could be a false sense of security from satisfactory performance on the included metrics, leading to less consideration of other context-specific evaluations. Additionally, there may be contexts where it is unethical or harmful to generate synthetic text data, regardless of performance on evaluations. We are committed to continuing to improve this package. Future efforts will focus on widening the evaluation criteria and metrics included, providing further visual or interactive tools for comparison, and enhancing reliability of the package.

Acknowledgments

We thank reviewers for their helpful feedback. This work was supported in part by the AI2050 Fellowship program by Schmidt Sciences. We also thank Tianli Xu for his valuable contribution in developing the annotation interface for this paper.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS'16. ACM.

Md Momin Al Aziz, Tanbir Ahmed, Tasnia Faequa, Xiaoqian Jiang, Yiyu Yao, and Noman Mohammed. 2021. Differentially private medical texts generation using generative neural networks. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–27.

Isabelle Rose I Alberto, Nicole Rose I Alberto, Arnab K Ghosh, Bhav Jain, Shruti Jayakumar, Nicole Martinez-Martin, Ned McCague, Dana Moukheiber, Lama Moukheiber, Mira Moukheiber, et al. 2023. The impact of commercial health datasets on medical research and health-care algorithms. *The Lancet Digital Health*, 5(5):e288–e294.

Brian Belgodere, Pierre Dognin, Adam Ivankay, Igor Melnyk, Youssef Mroueh, Aleksandra Mojsilovic, Jiri Navratil, Apoorva Nitsure, Inkit Padhi, Mattia Rigotti, et al. 2024. Auditing and generating synthetic data with controllable trust trade-offs. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*.

Karan Bhanot, Miao Qi, John S Erickson, Isabelle Guyon, and Kristin P Bennett. 2021. The problem of fairness in synthetic healthcare data. *Entropy*, 23(9):1165.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models. *arXiv:2202.07646*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 267–284, USA. USENIX Association.

Sai Chen, Fengran Mo, Yanhao Wang, Cen Chen, Jian-Yun Nie, Chengyu Wang, and Jamie Cui. 2023. A customized text sanitization mechanism with differential privacy. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5747–5758, Toronto, Canada. Association for Computational Linguistics.

Jenny Chim, Julia Ive, and Maria Liakata. 2024. Evaluating synthetic data generation from user generated text. *Computational Linguistics*, pages 1–44.

Davide Colla, Matteo Delsanto, Marco Agosto, Benedetto Vitiello, and Daniele P Radicioni. 2022. Semantic coherence markers: The contribution of perplexity metrics. *Artificial Intelligence in Medicine*, 134:102393.

Jip WTM de Kok, Miguel Á Armengol de la Hoz, Ymke de Jong, Véronique Brokke, Paul WG Elbers, Patrick Thoral, Alejandro Castillejo, Tomás Trenor, Jose M Castellano, Alberto E Bronchalo, et al. 2023. A guide to sharing open healthcare data under the general data protection regulation. *Scientific data*, 10(1):404.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Anjalie Field, Amanda Coston, Nupoor Gandhi, Alexandra Chouldechova, Emily Putnam-Hornstein, David Steier, and Yulia Tsvetkov. 2023. Examining risks of racial biases in nlp tools for child protective services. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '23, page 1479–1492, New York, NY, USA. Association for Computing Machinery.

Sorelle A Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. 2021. The (im) possibility of fairness: Different value systems require different mechanisms for fair decision making. *Communications of the ACM*, 64(4):136–143.

Nupoor Gandhi, Anjalie Field, and Emma Strubell. 2023. Annotating mentions alone enables efficient domain adaptation for coreference resolution. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10543–10558, Toronto, Canada. Association for Computational Linguistics.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. 2021. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*.

Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. 2021. Numerical composition of differential privacy. In *Advances in Neural Information Processing Systems*, volume 34, pages 11631–11642. Curran Associates, Inc.

Keith Harrigian, Ayah Zirikly, Brant Chee, Alya Ahmad, Anne Links, Somnath Saha, Mary Catherine Beach, and Mark Dredze. 2023. Characterization of stigmatizing language in medical records. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 312–329.

Tianxing He, Jingyu Zhang, Tianle Wang, Sachin Kumar, Kyunghyun Cho, James Glass, and Yulia Tsvetkov. 2023. On the blind spots of model-based evaluation metrics for text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12067–12097, Toronto, Canada. Association for Computational Linguistics.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA. Curran Associates Inc.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, Singapore. Association for Computational Linguistics.

Wei Huang, Yinggui Wang, and Cen Chen. 2024. Privacy evaluation benchmarks for NLP models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 2615–2636, Miami, Florida, USA. Association for Computational Linguistics.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. 2023. ProPILE: Probing privacy leakage in large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Haoran Li, Dadi Guo, Donghao Li, Wei Fan, Qi Hu, Xin Liu, Chunkit Chan, Duanyi Yao, Yuan Yao, and Yangqiu Song. 2024. PrivLM-bench: A multi-level privacy evaluation benchmark for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 54–73, Bangkok, Thailand. Association for Computational Linguistics.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. 2024. Best practices and lessons learned on synthetic data. In *First Conference on Language Modeling*.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-driven synthetic data generation, curation, and evaluation: A survey. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11065–11082, Bangkok, Thailand. Association for Computational Linguistics.

Justus Mattern, Zhijing Jin, Benjamin Weggenmann, Bernhard Schoelkopf, and Mrinmaya Sachan. 2022. Differentially private language models for secure data sharing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4873, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chuizheng Meng, Loc Trinh, Nan Xu, James Enouen, and Yan Liu. 2022. Interpretability and fairness evaluation of deep learning models on mimic-iv dataset. *Scientific Reports*, 12.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason. *Preprint*, arXiv:2311.11045.

Md Mahadi Hasan Nahid and Sadid Bin Hasan. 2024. Safesynthdp: Leveraging large language models for privacy-preserving synthetic data generation using differential privacy. *Preprint*, arXiv:2412.20641.

Shuchao Pang, Zhigang Lu, Haichen Wang, Peng Fu, Yongbin Zhou, Minhui Xue, and Bo Li. 2024. Reconstruction of differentially private text sanitization via large language models. *Preprint*, arXiv:2410.12443.

Ajay Patel, Colin Raffel, and Chris Callison-Burch. 2024. Datadreamer: A tool for synthetic data generation and reproducible llm workflows. *arXiv preprint arXiv:2402.10379*.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. 2016. The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410.

Mayana Pereira, Meghana Kshirsagar, Sumit Mukherjee, Rahul Dodhia, Juan Lavista Ferres, and Rafael de Sousa. 2024. Assessment of differentially private synthetic data for utility and fairness in end-to-end machine learning pipelines for tabular data. *Plos one*, 19(2):e0297271.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*.

Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. 2022a. The text anonymization benchmark (tab): A dedicated corpus and evaluation framework for text anonymization. *Computational Linguistics*, 48(4):1053–1101.

Ildikó Pilán, Pierre Lison, Lilja Øvrelid, Anthi Papadopoulou, David Sánchez, and Montserrat Batet. 2022b. The text anonymization benchmark (tab): A dedicated corpus and evaluation framework for text anonymization. *Computational Linguistics*, 48(4):1053–1101.

Zhaozhi Qian, Bogdan-Constantin Cebere, and Mihaela van der Schaar. 2023. Synthcity: facilitating innovative use cases of synthetic data in different data modalities.

Alvin Rajkomar, Michaela Hardt, Michael D Howell, Greg Corrado, and Marshall H Chin. 2018. Ensuring fairness in machine learning to advance health equity. *Annals of internal medicine*, 169(12):866–872.

Krithika Ramesh, Nupoor Gandhi, Pulkit Madaan, Lisa Bauer, Charith Peris, and Anjalie Field. 2024. Evaluating differentially private synthetic data generation in high-stakes domains. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15254–15269, Miami, Florida, USA. Association for Computational Linguistics.

Avi Schwarzschild, Zhili Feng, Pratyush Maini, Zachary Chase Lipton, and J Zico Kolter. 2024. Rethinking LLM memorization through the lens of adversarial compression. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. 2024. Beyond memorization: Violating privacy via inference with large language models. In *The Twelfth International Conference on Learning Representations*.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957, Miami, Florida, USA. Association for Computational Linguistics.

Xinyu Tang, Richard Shin, Huseyin A Inan, Andre Manoel, Fatemehsadat Mireshghallah, Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, and Robert Sim. 2024. Privacy-preserving in-context learning with differentially private few-shot generation. In *The Twelfth International Conference on Learning Representations*.

Ozlem Uzuner, Andreea Bodnari, Shuying Shen, Tyler Forbush, John Pestian, and Brett R South. 2012. Evaluating the state of the art in coreference resolution for electronic medical records. *Journal of the American Medical Informatics Association*, 19(5):786–791.

Siyuan Wu, Yue Huang, Chujie Gao, Dongping Chen, Qihui Zhang, Yao Wan, Tianyi Zhou, Xiangliang Zhang, Jianfeng Gao, Chaowei Xiao, and Lichao Sun. 2024. Unigen: A unified framework for textual dataset generation using large language models. *Preprint*, arXiv:2406.18966.

Tong Wu, Ashwinee Panda, Jiachen T. Wang, and Prateek Mittal. 2023. Privacy-preserving in-context learning for large language models. *Preprint*, arXiv:2305.01639.

Rui Xin, Niloofar Mireshghallah, Shuyue Stella Li, Michael Duan, Hyunwoo Kim, Yejin Choi, Yulia Tsvetkov, Sewoong Oh, and Pang Wei Koh. 2024. A false sense of privacy: Evaluating textual data sanitization beyond surface-level privacy leakage. In *Neurips Safe Generative AI Workshop 2024*.

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. *Preprint*, arXiv:2304.12244.

Xiang Yue, Huseyin Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. 2023. Synthetic text generation with differential privacy: A simple and practical recipe. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1321–1342, Toronto, Canada. Association for Computational Linguistics.

A Background: Differential Privacy

Differential privacy (DP) offers a formal privacy guarantee that ensures that any individual's data cannot be inferred from a query applied to a dataset (Dwork et al., 2006, 2014). In other words, the result of such a query is nearly indistinguishable from the result of the same query applied to a dataset that either includes a modified version of the individual's data or excludes the record entirely, thereby preserving the individual's privacy. In this case, the notion of adjacency is defined as a difference of a single record in the original dataset D and the modified dataset D'. Formally, DP is defined as follows:

Definition: Given a dataset D and an adjacent dataset D', which is produced by removing or modifying a single record from D, a randomized algorithm $F : D \to Y$ is (ϵ, δ)-private if for any two neighboring datasets D, D', with the constraints $\epsilon > 0$ and $\delta \in [0, 1]$, the following holds true for all sets $y \subseteq Y$:

$$\Pr[F(D) \in y] \leq e^{\epsilon} \Pr[F(D') \in y] + \delta$$

The value of ϵ denotes the privacy budget, while δ specifies the likelihood that the privacy guarantee may fail. When δ is set to 0, this implies a purely differentially private setting with no probability of the guarantee being broken. The value of ϵ constrains how similar the outputs of both distributions are; a higher ϵ value indicates a greater privacy budget, meaning the algorithm is less private. DP guarantees that even if an adversary has access to any side-knowledge, the privacy leakage of (ϵ, δ)-DP algorithms will not increase. Additionally, another property of DP is that it ensures that any post-processing on the outputs of (ϵ, δ)-differentially private algorithms will remain (ϵ, δ)-differentially private.

We use DP-SGD (Abadi et al., 2016), a modification to the stochastic gradient descent (SGD) algorithm, which is typically used to train neural networks. DP-SGD clips the gradients to limit the contribution of individual samples from the training data and subsequently adds noise from a predefined type of distribution (such as a Gaussian distribution) to the sum of the clipped gradients across all samples. DP-SGD thus provides a differentially private guarantee to obfuscate the gradient update, thereby ensuring that the contribution of any given sample in the training data is indistinguishable due to the aforementioned post-processing property. This process ensures (ϵ, δ)-differential privacy for each model update. Given a privacy budget, number of epochs, and other training parameters, we can estimate the privacy parameters using estimation algorithms (Gopi et al., 2021).

B Privacy metrics

Canary-based evaluations (Carlini et al., 2019) involving crafting canaries that contain sensitive information and injecting them into the training data. The probability of generating the canary is estimated in contrast to the model trained on the corpus without the canary (a model that has "memorized" the canary would have a higher likelihood of generating it). In addition to this, we implement entity-centric evaluation criteria to quantify the Entity Leakage Percentage (ELP) and the proportion of an entity's context leaked, where the user can define the context window length (Ramesh et al., 2024).

The ELP is calculated as follows, where E_{leaked} is the number of entities leaked in the synthetic generations, and E_{total} represents the total number of entities in the training data.

$$P_{\text{entities}} = \frac{E_{\text{leaked}}}{E_{\text{total}}} \times 100$$

Similarly, we quantify the memorized spans of context where the entities appear as follows:

$$P_{\text{context}}(k) = \frac{C_{\text{leaked}}(k)}{C_{\text{total}}(k)} \times 100$$

where $C_{\text{leaked}}(k)$ is the number of occurrences where the entity and its surrounding context (of length k) are leaked in the synthetic generation and $C_{\text{total}}(k)$ is the total number of occurrences where the entity and its context (of length k) appear in the original training data.

C Fairness metrics

Where D is a set consisting of all subgroups corresponding to a demographic attribute within the dataset, Equalized Odds is defined as:

$$\text{EO}_D = \max\Big(\max_{d \in D}(\text{TPR}_d) - \min_{d \in D}(\text{TPR}_d), \max_{d \in D}(\text{FPR}_d) - \min_{d \in D}(\text{FPR}_d)\Big)$$

Equalized difference sums the differences between subgroup performance and overall performance for a specific metric. For example, FPR is

an important metric in domains such as healthcare, where false positive diagnoses can place increased burdens on patients in the form of unnecessary treatment or exams (Rajkomar et al., 2018). Thus, the False Positive Equality Difference (FPED) metric is the sum of the differences between the overall FPR for the entire dataset and the FPR for each subgroup:

$$\text{FPED} = \sum_{d=1}^{D} |\text{FPR}_{\text{overall}} - \text{FPR}_d| \quad (1)$$

ED metrics for true positive (TPED), true negative (TNED) and false negative (FNED) are computed similarly. Lower values of EO and ED scores indicate that the model's performance is more consistent across different subgroups.

D Automated Text Quality Evaluation Metrics

We provide the full equations to calculate our two automated qualitative evaluation metrics, MAUVE and Fréchet Inception Distance.

MAUVE is computed by first sampling human text $x_i \sim P$ (reference) and generated text $x'_i \sim Q$ (synthetic). Embeddings for this text are then obtained from an external model M (we use GPT-2 ()). These embeddings are quantized through clustering, yielding "low-dimensional discrete distributions that approximate each high-dimensional text distribution" (Pillutla et al., 2021). Finally, a divergence curve is calculated by taking the KL-divergence between the reference distribution and synthetic distribution as a mixture weight λ is varied, as shown in Equation 2, and MAUVE is calculated as the area under this curve.

$$C(P,Q) = \Big\{ \big(e^{-c\text{KL}(Q\|R_\lambda)}, e^{-c\text{KL}(P\|R_\lambda)}\big) : R_\lambda = \lambda P + (1-\lambda)Q,\ \lambda \in (0,1) \Big\}. \quad (2)$$

The Fréchet Inception Distance is obtained by measuring the Wasserstein-2 distance between the Gaussian obtained from the reference data (m, C) and the Gaussian of the synthetic data (m_w, C_w):

$$d^2((m,C),(m_w,C_w)) = \|m - m_w\|_2^2 + Tr(C + C_w - 2(CC_w)^{1/2}$$

E Control Codes

We adapt the dominant approach from prior work (Yue et al., 2023): which involves first fine-tuning a pre-trained autoregressive language model on real, in-domain data and then generating synthetic data from the fine-tuned model. We compare fine-tuning the model with and without DP, where we use DP-SGD for differentially private fine-tuning. After fine-tuning, we utilize top-k sampling (Fan et al., 2018) and nucleus sampling (Holtzman et al., 2020) to generate diverse synthetic notes.

We condition the text generation on *control codes* (Keskar et al., 2019). During training, we prepend one or more labels associated with the text to the model input. We similarly prepend control codes during inference, where we sample the provided codes from their distribution in the training data. Thus, during training and inference, the probability distribution of the subsequent text $x = \{x_1, x_2 ... x_n\}$ is conditioned on the control code information c, which is described by the following equation:

$$P(x|c) = \prod_{i=1}^{n} P(x_i|x_1 ... x_{i-1}, c) \quad (3)$$

The format of the control code for the MIMIC-III data is as follows: *Long_Title: <diagnoses>, ICD9_CODE: <codes>, Gender: <gender>, Ethnicity: <ethnicity>*, where the <diagnoses> variable represents the long title form of the ICD-9 codes, information that is already provided with the MIMIC-III dataset. An example control code is as follows: *Diagnosis: Long_Title: Unspecified essential hypertension, Atrial fibrillation ICD9_CODE: 4019, 42731 Gender: Female Ethnicity: WHITE*

F Sample Usage Results

Domain	Avg Length Of Dsynth	Avg. Num Unique Words	Min Text Length	Max Text Length	Jaccard Similarity	Cosine Similarity
TAB (DP - inf)	617.725	265.882	452	745	0.185	0.522
TAB (DP - 8)	613.75	245.95	8	787	0.147	0.474
TAB (Real)	1342.42	487.966	185	5144	-	-

Table 3: Descriptive statistics for TAB across real and synthetic datasets.

	FID	MAUVE	Avg. Perplexity
$D_{real(test)}$	0.788	0.941	15.38
$D_{\epsilon=\infty}$	0.797	0.485	11.474
$D_{\epsilon=8}$	0.790	0.831	10.209

Table 4: Text quality metrics for TAB. The first column indicates the comparison dataset, with $D_{real(train)}$ as the reference dataset for each.

	Healthcare	TAB
	% Entities	**% Entities**
$D_{\epsilon=\infty}$	4.8	50.3
$D_{\epsilon=8}$	3.8	48.5

Table 5: Percent of unique PII from the training data that appear in the synthetic generations.

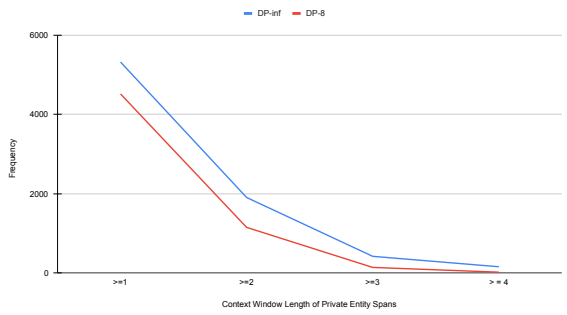


Figure 3: Memorization of private entities in the TAB dataset as context window length increases.

	Mention Detection	Coreference
$D_{real(gold)}$	0.800 ± 0.005	0.706 ± 0.006
$D_{real(silver)}$	0.659 ± 0.121	0.552 ± 0.126
$D_{\epsilon=\infty}$	0.650 ± 0.008	0.483 ± 0.002
$D_{\epsilon=8}$	0.656 ± 0.006	0.491 ± 0.003

Table 6: F1 scores for coreference and mention detection for MIMIC-III over entities from human-annotated test splits of the i2b2/VA datasets. All synthetic datasets are annotated with silver labels.

		FNED	Equalized Odds
Race	D_{real}	0.35 ± 0.0	0.20 ± 0.0
	$D_{\epsilon=\infty}$	0.39 ± 0.005	0.23 ± 0.001
	$D_{\epsilon=8}$	0.53 ± 0.014	0.30 ± 0.003
Gender	D_{real}	0.04 ± 0.0	0.04 ± 0.0
	$D_{\epsilon=\infty}$	0.02 ± 0.007	0.02 ± 0.007
	$D_{\epsilon=8}$	0.04 ± 0.005	0.04 ± 0.005

Table 7: Fairness evaluation for the MIMIC-III ICD-9$_{n=10}$ task, for the gender and race categories. Higher values indicate poorer group fairness performance.

	Rank ($\epsilon = \infty$ / 8)	Perplexity ($\epsilon = \infty$ / 8)
100 Insertions		
Name	4628 / 3356	35.80 / 53.19
Address	5 / 3967	16.52 / 61.37
Number	1 / 818	5.77 / 14.46
Email	1 / 1410	10.50 / 70.26
10 Insertions		
Name	5986 / 3378	49.72 / 54.10
Address	2276 / 4075	43.59 / 62.66
Number	902 / 841	9.43 / 14.61
Email	711 / 1452	37.81 / 72.08
1 Insertion		
Name	6037 / 3383	52.06 / 54.20
Address	3348 / 4081	54.50 / 62.79
Number	1084 / 838	9.82 / 14.63
Email	1941 / 1457	43.90 / 72.28
0 Insertions		
Name	6086 / 5265	44.81 / 57.54
Address	4565 / 3869	75.79 / 61.68
Number	1217 / 1522	11.80 / 13.33
Email	1003 / 3174	43.80 / 55.19

Table 8: Rank and perplexity metrics for canary attacks over Healthcare (MIMIC) data. Each entry is shown as $\epsilon = \infty$ / 8. Differential privacy ($\epsilon = 8$) increases both metrics, improving privacy for all canaries.

Quest2DataAgent: Automating End-to-End Scientific Data Collection

Tianyu Yang[1], **Yuhan Liu**[2], **Ethan Brown**[4], **Sobin Alosious**[3], **Jason Rohr**[4], **Tengfei Luo**[3], **Xiangliang Zhang**[1*]

[1]Department of Computer Science and Engineering, University of Notre Dame
[2]Department of Chemistry and Biochemistry, University of Notre Dame
[3]Department of Aerospace and Mechanical Engineering, University of Notre Dame
[4]Department of Biological Sciences, University of Notre Dame

Abstract

Scientific research often requires constructing high-quality datasets, yet the current workflows remain labor-intensive, and dependent on domain expertise. Existing approaches automate isolated steps such as retrieval or generation, but lack support for the full end-to-end data collection process. We present Quest2DataAgent, a general-purpose multi-agent framework for automating scientific data collection workflows. Given a natural language research question, it decomposes tasks into structured subtasks, retrieves relevant data using hybrid strategies, evaluates dataset quality, and generates visualizations through a conversational interface. We demonstrate its flexibility in two domains: EcoData for ecological research and PolyData for polymer materials. Both systems share the same core architecture but operate over distinct datasets and user needs. Human evaluations show that Quest2DataAgent significantly improves data relevance, usability, and time efficiency compared to manual collection and tool-assisted baselines. The framework is open-source and extensible to other domains.[1]

1 Introduction

Recent studies have explored automated methods to alleviate the labor-intensive and time-consuming manual workflows typical of scientific data collection (Xu et al., 2021; Chen et al., 2022; Chew, 2023; Liu et al., 2024). Tool-based approaches (Liu et al., 2024; Schick et al., 2023; Qin et al., 2023) leverage search engine APIs to automatically label, and clean image data from web sources. In parallel, LLM-driven frameworks (Huang et al., 2024; Zhu et al., 2023; Wang et al., 2024b) utilize large language models (LLMs) to support dataset design, synthetic data generation, and automated annotation. Additionally, multi-agent systems (Borgeaud et al., 2022; Lewis et al., 2020; Yu et al., 2023) have been proposed for synthesizing new datasets (Sengupta et al., 2024; Long et al., 2024; Arif et al., 2024). While these methods can retrieve or generate data that match specific predefined criteria, they typically address only a portion of the broader end-to-end data collection workflow. In scientific research, the full pipeline from formulating a question to acquiring relevant data often involves multiple stages of reasoning, and iterative refinement that go beyond data retrieval or generation alone.

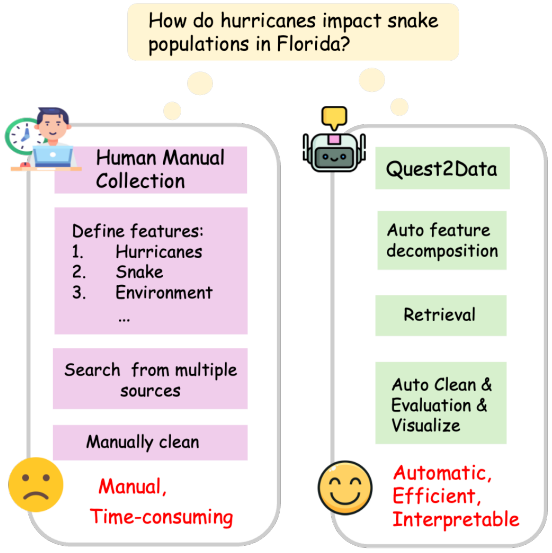


Figure 1: Scientific Data Collection: Labor-intensive manual process vs. automated Quest2Data workflow.

As shown in Fig. 1 (left), for addressing an ecological research question: *"How do hurricanes affect snake populations in Florida?"*, scientists must first decompose the problem into well-defined subtasks and then identify and collect relevant ecological data across multiple dimensions from diverse sources. This workflow requires labor-intensive efforts and relies heavily on domain expertise. Our collaborators in domains such as biology, ecology and material science have expressed signifi-

*Corresponding author: xzhang33@nd.edu

[1]Code is available at https://github.com/Tianyu-yang-anna/Quest2DataAgent. A video demonstration is available at https://youtu.be/7-XEeVpdPZk.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 500–514
November 4-9, 2025

cant challenges in managing this process efficiently, highlighting a critical need for automation in scientific data collection and analysis.

To address these challenges, we introduce **Quest2DataAgent**, the first multi-agent system specially designed to support end-to-end scientific data collection workflows, from problem formulation and data retrieval to evaluation and analysis, as shown in Fig. 1 (right). This system is motivated by the growing capabilities of LLMs to emulate expert-level reasoning across domains (Wu et al., 2025; Qian et al., 2024; Wang et al., 2025). Recent advances, such as OpenAI's Deep Research (OpenAI, 2025), demonstrate that LLMs can effectively navigate large-scale knowledge spaces. However, while such systems excel in reasoning and synthesis, they fall short when it comes to automated, grounded scientific data collection. Quest2DataAgent bridges this gap by simulating expert workflows: decomposing a research question into structured subtasks, retrieving data from heterogeneous sources through a plugin-based architecture, evaluating dataset relevance using automated LLM-based evaluators, and dynamically generating intuitive visualizations.

To demonstrate the feasibility and value of Quest2DataAgent, we developed two system prototypes tailored to ecology and materials science, respectively named as **EcoData** and **PolyData**.

- **EcoData**: A demo system for ecological data collection that supports structured queries across 1,900 species and 172,000 recorded observations, along with environmental events (e.g., hurricanes, wildfires) and climate variables such as temperature, and precipitation.
- **PolyData**: A demo system for polymer data collection that enables fine-grained queries based on chemical structures and functionalities. It supports search over 12,800 polymers, including comprehensive polymer metadata and simulated physical properties.

These two prototype systems demonstrate Quest2DataAgent's capability to support complex scientific queries across domain-specific datasets. Notably, both prototypes are built on curated data repositories that are fully accessible, rather than relying on open-ended web searches or access-controlled data sources. While such scenarios are left for future extensions, or may require alternative strategies for secure and scalable access, the current systems already reduce manual data collection workloads from several days to just 10 minutes, highlighting the efficiency and transformative potential of Quest2DataAgent.

Quest2DataAgent is domain-agnostic, and easily adaptable to new scientific domains by modifying only the user-facing interface and data plugins. The core modules include: (1) Planner Agent: breaks down complex questions into structured subtasks; (2) Retriever Agent: performs hybrid retrieval via a plugin-based architecture; (3) Data Preprocessing Module: cleans, standardizes, and integrates data; (4) Evaluation Agent: uses LLMs to assess dataset relevance and suggest additional resources; (5) Visualization Agent: generates tailored, executable visualization code; (6) Conversational Interface: enables interactive, multi-turn data exploration. Further details are presented in Section 3.

In summary, Quest2DataAgent provides a flexible, modular framework for automating end-to-end scientific data collection. Through qualitative case studies and human evaluations for EcoData and PolyData, we confirm Quest2DataAgent's effectiveness in reducing manual effort while preserving the relevance and usability of the retrieved data.

2 Related Works

LLM-based Scientific Assistants: Large language models (LLMs) have shown significant potential in scientific tasks such as question answering (Dasigi et al., 2021; Lee et al., 2023; Xu et al., 2024), summarization (Ding et al., 2023; Takeshita et al., 2024; Li et al., 2023), and literature recommendation (Zheng et al., 2024; Pinedo et al., 2024). Recent work extends LLM usage to hypothesis generation (Wang et al., 2024a; Si et al., 2024), experimental design (Bran et al., 2023; Li et al., 2024; Lu et al., 2024), and manuscript drafting (Wang et al., 2019; August et al., 2022). However, existing systems typically overlook dataset-centric tasks, such as retrieval, cleaning, and visualization, essential for data-driven research.

Quest2DataAgent addresses this gap by automating dataset retrieval, integration, evaluation, and visualization, thereby providing end-to-end support for complex scientific research workflows.

Scientific Data Retrieval: Retrieval-augmented generation (RAG) enhances LLMs with external knowledge sources (Borgeaud et al., 2022; Lewis et al., 2020; Yu et al., 2023). Early RAG systems focused on unstructured text, but recent work covers more modalities and domains, including finance (Zhao et al., 2025), education (Jia

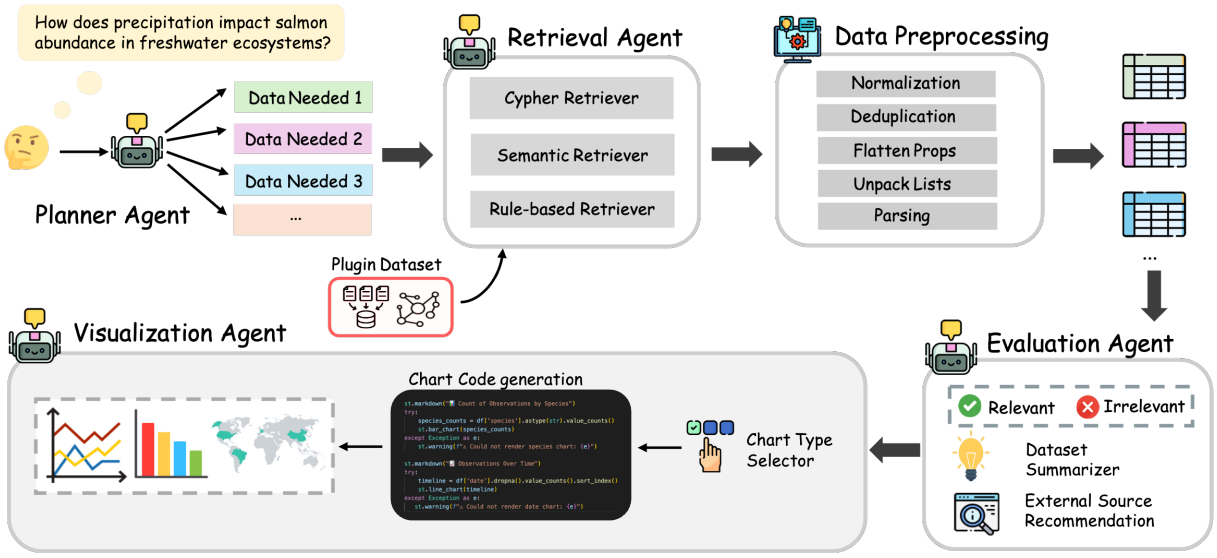


Figure 2: Overview of the Quest2DataAgent framework.

et al., 2025), and multimodal scientific knowledge (Tian et al., 2025; Han et al., 2025). Tools like WildlifeLookup (Wang et al., 2025) and HeteRAG (Yang et al., 2025) enable structured or heterogeneous retrieval for scientific tasks.

However, most of these systems are built for only the retrieval step, which is only a portion of the end-to-end data collection workflow.

LLM-based Multi-Agent Systems: Recent LLM-based multi-agent frameworks (Qin et al., 2023; Hao et al., 2023; Guo et al., 2024) have demonstrated strong capabilities in both complex planning and collaborative problem-solving tasks (Chen et al., 2024; Rasal; Qian et al., 2024; Dibia et al., 2024; Ku et al., 2025). However, most of these systems are not tailored to the needs of scientific data workflows. Quest2DataAgent addresses this gap by introducing a modular multi-agent architecture specifically designed for scientific dataset construction.

3 The Quest2DataAgent System

As illustrated in Fig. 2, the Quest2DataAgent system integrates multiple specialized modules that work together to decompose tasks, retrieve multimodal data, assess relevance, visualize results, and support interactive exploration.

3.1 Planner Agent

A core challenge in scientific data construction lies in the abstract and ambiguous nature of many research questions, which often lack explicit data requirements. These queries frequently span multiple domains and require the integration of heterogeneous data sources. To address this, Quest2DataAgent introduces a Planner Agent that simulates the reasoning process of researchers.

Given a research question, the Planner Agent (1) identifies key semantic dimensions, such as target species, spatial scope, and temporal resolution, and (2) decomposes the question into a set of structured subtasks. Each subtask contains a concise title that indicates the dataset needed and a short description of the corresponding data requirement. This structured decomposition transforms vague research queries into executable data retrieval plans, laying the foundation for efficient and interpretable downstream processing.

To ensure accurate task decomposition, we design a structured prompt template that clearly defines the role of the Planner Agent, the expected output format, and the overall task objective. We incorporate Chain-of-Thought (CoT) prompting (Wei et al., 2022) to guide the agent through step-by-step reasoning. The prompt includes a few carefully selected in-domain examples, each illustrating how to transform a complex research question into structured subtasks with concise titles and descriptions. These few-shot examples enable the agent to generalize effective decomposition strategies to previously unseen queries. The full prompt is shown in Appendix A.3.

3.2 Multi-Strategy Retriever Agent

Scientific data collection faces unique challenges: relevant information is often dispersed across het-

erogeneous sources, with data distributed among structured knowledge graphs, semi-structured tables, and unstructured documents. Coverage is frequently sparse or fragmented, schemas and query patterns are diverse, and access restrictions or data quality issues are common obstacles. These complexities make it difficult to retrieve all relevant data using a single retrieval strategy.

To address this, Quest2DataAgent adopts multiple retrieval strategies (**Cypher**, **semantic**, and **rule-based**) and dynamically selects them based on query type and data modality. Given a subtask query q and a candidate data unit $x \in \mathcal{X}$, where $\mathcal{X}$ is the set of all retrievable data units (e.g., graph triples, table rows, metadata entries), the system computes a relevance score $R(q, x)$ to surface semantically aligned results. Each retrieval strategy is tailored to the specific format of x, enabling effective handling of structured, semi-structured, and unstructured scientific data.

Cypher Retrieval: For structured data in knowledge graphs (e.g., Neo4j), LLM-generated Cypher queries, conditioned on schema descriptions, enable multi-hop reasoning over entity-event-location chains. If the result set is too small or previously seen (based on hash), a fallback is triggered to avoid trivial or redundant outputs.

Semantic Retrieval: For unstructured data, we encode q and each $x \in \mathcal{X}$ into dense vectors via a pretrained encoder, computing $R(q, x) = \langle f(q), f(x) \rangle$. Top-$k$ results are retrieved, optionally re-ranked with metadata filters (e.g., species, location, time). This strategy handles vague queries and diverse schemas.

Rule-Based Retrieval: When the above fail, a rule-based module extracts entities and constraints from q and fills Cypher/SQL templates (e.g., *"species in Florida before 2010"*), ensuring coverage under noisy or under-specified inputs.

Together, these strategies provide robust, interpretable, and flexible retrieval across structured and unstructured sources. When no relevant data can be retrieved due to limited coverage or access restrictions, a fallback module suggests external data sources with direct links, ensuring continuity by guiding users to alternative data pathways.

Plugin Dataset Support: To accommodate diverse domains, users can connect custom Neo4j databases or upload structured text files.

3.3 Data Processing Module

Scientific data collection often poses challenges beyond those in general datasets. For example, polymer science faces limited and fragmented databases, slow and costly data generation, and high sensitivity of properties to experimental conditions. Key fields such as molecular structure or SMILES notation are frequently missing or inconsistently formatted, requiring extensive expert curation. Manual cleaning is tedious, error-prone, and demands domain expertise.

To address these issues, Quest2DataAgent employs a data processing module that performs schema normalization, noise reduction, and format unification. The system flattens nested structures, unpacks singleton lists, standardizes key semantic fields, and removes duplicates via hash-based comparison. Type conversion and missing value imputation are guided by schema heuristics.

The processed output is (1) structurally aligned for downstream evaluation, (2) compatible with automatic visualization generation, and (3) free from trivial redundancy. This automation transforms raw, heterogeneous data into structured and interoperable datasets, reducing manual effort and preserving critical scientific information for downstream analysis. This stage ensures robustness and scalability across scientific domains without requiring manual data processing.

3.4 Evaluation Agent

Assessing dataset relevance and quality is a critical step in scientific workflows. An agent-based evaluation module is thus introduced to use LLMs as an automated judger to streamline this process.

For each retrieved and processed dataset, the LLM is prompted with the subtask description, schema metadata, and representative samples. It evaluates whether the dataset satisfies the subtask's information need. If deemed relevant, the LLM generates a concise summary outlining the dataset's content, structure, and utility. If not, it suggests alternative data sources or follow-up strategies.

By automating judgment and fallback suggestion, this module reduces manual screening effort, enhances interpretability, and improves system robustness in low-coverage scenarios.

3.5 Visualization Module

We include a visualization module that enables users to interactively explore complex scientific

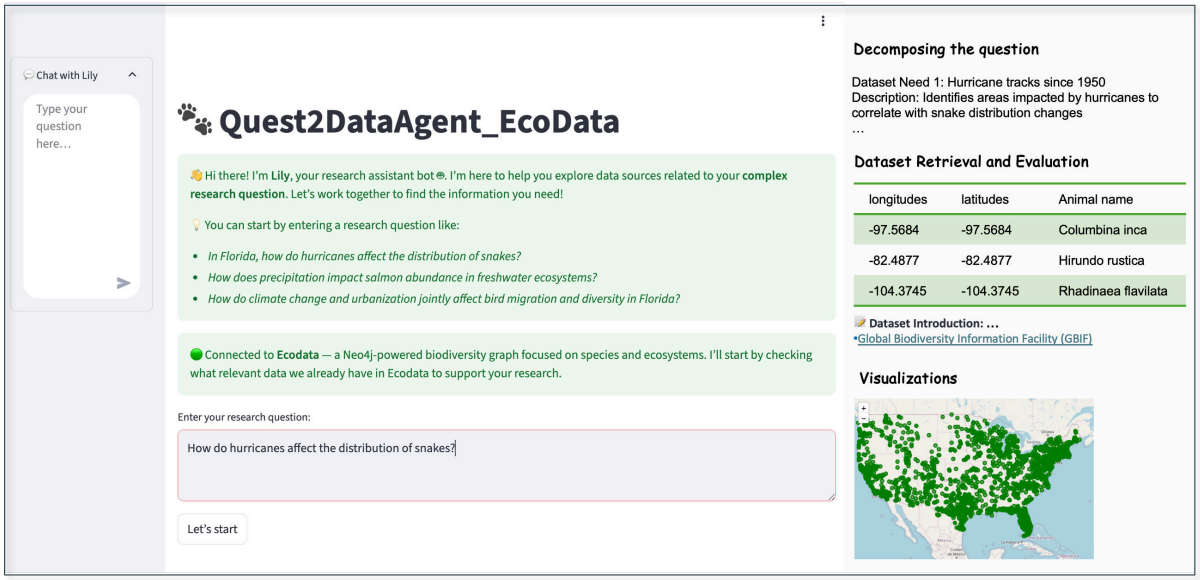


Figure 3: Interface of EcoData.

data. Placing this module in the middle of the pipeline allows for early inspection and validation of intermediate results. This ensures that potential errors, noise, or irrelevant data can be identified and corrected before proceeding further. Together with the conversational interface (introduced next), domain experts can assess data quality and iteratively refine their queries, improving overall efficiency and result accuracy.

For each processed dataset, the system inspects schema metadata and sampled rows, prompting the LLM to recommend an appropriate visualization type and generate Python-based code (e.g., via `Streamlit` and `pydeck`). The module supports diverse formats such as bar charts, line plots, geospatial maps, and molecular structure diagrams, depending on the data characteristics.

Quest2DataAgent features a conversational interface, enabling natural language interaction throughout the data construction workflow (see Fig. 3). It is context-aware, incorporating current subtask states and evaluation outcomes to support multi-turn dialogue. Users can iteratively refine queries, request clarifications, and explore related tasks, with conversation history maintained for coherence.

3.6 System Optimization

To support responsive and scalable interaction, the system applies lightweight optimization, including caching for repeated operations and a modular architecture with fallback logic to ensure robustness.

4 Demos of EcoData and PolyData

To demonstrate the versatility and domain-adaptability of Quest2DataAgent, we build two prototype systems: **EcoData** (Ecology)[2] and **PolyData** (Materials Science)[3]. These systems share the same backend and workflow engine but differ in domain-specific plugin datasets and user interfaces. Together, they showcase the ability of Quest2DataAgent to generalize across scientific domains while maintaining high usability and retrieval effectiveness. They are implemented as web-based systems using Streamlit, deployed on Hugging Face Spaces, and open-sourced under the MIT License. They are powered by GPT-4o (Hurst et al., 2024) and support modular integration of other foundation models (e.g., LLaMA (Touvron et al., 2023), Qwen (Bai et al., 2023)) and FAISS (Douze et al., 2024) for vector-based retrieval.

4.1 Demonstration of EcoData

EcoDatais a domain-specific demonstration of Quest2DataAgent framework for ecological research. It integrates a structured knowledge graph containing over 172,000 species observations (from GBIF (Global Biotic Interactions (GloBI), 2024), USGS, iNaturalist), 1,932 interspecies relationships, and environmental events (e.g., hurricanes, wildfires) from NOAA and KnowWhereGraph. Each entity is enriched with Wikidata and IUCN (International Union for Conservation of Na-

[2] https://huggingface.co/spaces/tyang4/Quest2DataAgent_EcoData
[3] https://huggingface.co/spaces/tyang4/poly_retrivel

ture, 2024) identifiers. The system incorporates both Neo4j-based graphs and structured tabular datasets (e.g., observation logs, climate reports). Fig. 3 shows the interface of the EcoData demo (more demo screenshots in Appendix Figure 7)

4.2 Demonstration of PolyData

PolyData is another domain-specific deployment of Quest2DataAgent framework tailored for polymer research. It comprises around 12,800 polymer entries sourced from PolyInfo (Ishii et al., 2024) and open literature, annotated with IUPAC names, SMILES strings, and polymer classes. Roughly 4,000 entries include polymerization metadata (e.g., monomer names, reaction types), and 1,000 contain molecular dynamics-based property data (Ishii et al., 2024) (e.g., heat capacity, dielectric constant, thermal conductivity). Appendix Fig. 5 shows the interface, and a complete demonstration screenshot is provided in Fig. 6 in the appendix. More details are available in Appendix A.2.

5 Evaluation Settings and Results

5.1 Experimental Setup

We conduct a comprehensive evaluation on both EcoData and PolyData to assess their effectiveness in supporting scientific data collection workflows. We recruited 18 domain researchers, with 9 participants each from ecology and materials science. This group includes 13 PhD students (72%) and 5 postdocs (28%).

Metrics: All participants complete a standardized task, record time spent, and respond to a questionnaire consisting of 12 Likert-scale (1–5) items and open-ended feedback forms. The 12 questions cover the following six evaluation criteria: M1. Task Decomposition Accuracy: Whether the system generates appropriate sub-questions, M2. Data Relevance and Coverage: Usefulness and completeness of retrieved datasets, M3. Evaluation and Recommendation: Effectiveness of automated dataset evaluation and external resource suggestion, M4. Cleaning and Visualization Quality: Clarity of structured output and informativeness of generated charts, M5. User Experience: Ease of use, responsiveness, and satisfaction, M6. Time Efficiency: Task completion time.

Baselines: As there is currently no existing system that supports end-to-end scientific data collection, we compare our framework against two representative baseline methods, and Table 1 summarizes the key capabilities across all approaches.

Capability	Manual	Tool-assisted	Ours
Task Planning	✗	✓	✓
Retrieval	✗	✗	✓
Evaluation	✗	Partial	✓
External Suggestion	✗	Partial	✓
Preprocessing	✗	✗	✓
Visualization	✗	✗	✓
Dialogue Interface	✗	✓	✓
End-to-End Workflow	✗	✗	✓

Table 1: Comparison of capabilities across baseline methods and Quest2DataAgent.

1. Manual workflow: Researchers manually break down the research question, search for datasets across multiple platforms, clean the data in spreadsheets, and create visualizations.

2. Tool-assisted: Participants use tools like LLMs or domain-specific platforms to assist with dataset search. However, they still manually assess relevance, clean data, and generate visualizations.

5.2 Results

Quantitative Results and Analysis: As shown in Figure 4, our system consistently achieves higher scores than both baselines across all aspects.

On PolyData, Quest2DataAgent obtains the highest ratings in nearly every aspect. It performs especially well in task decomposition, achieving an average score of 4.87 compared to 3.17 for Baseline 1 (manual workflow) and 4.36 for Baseline 2 (tool-assisted). It also shows statistically significant improvements in dataset retrieval, recommendation, preprocessing clarity, and user interaction. In particular, its user interaction score is more than three points higher than that of Baseline 1, indicating a much smoother and more intuitive experience. Perceived efficiency is also rated highly by users.

On EcoData, the system maintains strong performance. It achieves the highest score in evaluation and recommendation, scoring 4.47 compared to 2.54 for Baseline 1 and 3.79 for Baseline 2. It also receives consistently higher ratings in preprocessing and user interaction. Although the relative improvements are smaller than those observed on PolyData, Quest2DataAgent remains the top-performing system across all dimensions.

User Feedback Analysis: In addition to quantitative results, we collected user feedback comparing the Manual Workflow, Tool-Assisted Workflow, and Quest2DataAgent. Participants consistently praised Quest2DataAgent for its ability to decom-

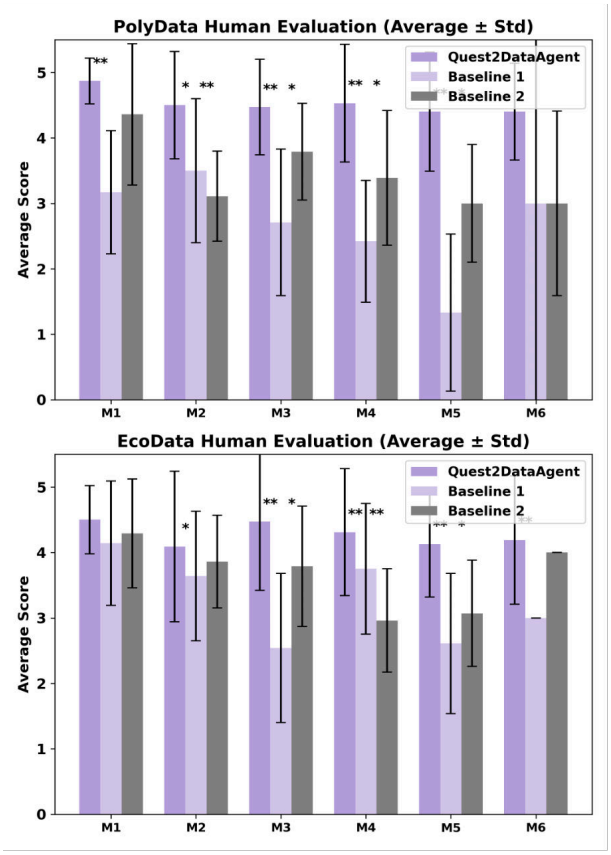


Figure 4: Quantitative human evaluation results on **PolyData** (top) and **EcoData** (bottom) with Baseline 1: manual workflow and Baseline 2: tool-assisted workflow. Bars show mean scores ± standard deviation. Asterisks denote significance based on two-sided paired t-tests ($p < 0.05$: *, $p < 0.01$: **, $p < 0.001$: ***).

pose complex questions, retrieve relevant datasets, and present intuitive visualizations. Users noted that it "quickly helped obtain and filter datasets" and "enabled efficient screening," particularly benefiting those unfamiliar with domain-specific data collection.

In contrast, manual workflows were described as time-consuming and fragmented, requiring domain expertise and repetitive integration across platforms. Tool-assisted workflows helped with dataset suggestions but still required iterative prompting, manual filtering, and source verification. Users also raised concerns about hallucinations and lack of transparency.

Overall, the feedback highlights that Quest2DataAgent offers a more streamlined and guided experience, significantly reducing user effort while improving clarity and efficiency.

Statistical Significance: Paired two-sided t-tests confirm that the observed improvements are statistically significant in multiple aspects, particularly task decomposition, recommendation, visualization clarity, and user interaction. These results demonstrate the system's effectiveness in improving usability and reducing manual effort.

Method	EcoData Time (min)	PolyData Time (min)
Ours	13.06	7.38
Baseline 1	488.00 (~8.1h)	6330.00 (~105.5h)
Baseline 2	49.80	1724.58 (~28.7h)

Table 2: Average time spent to complete tasks under different methods in Eco and Poly domains.

Time Efficiency: We compare the time required for scientific data collection using Quest2DataAgent, manual workflows, and tool-assisted methods. As shown in Table 2, Quest2DataAgent substantially reduces time cost in both domains. In EcoData, our system averages 13.06 minutes, compared to 488 minutes for manual and 49.8 minutes for tool-assisted workflows. In PolyData, Quest2DataAgent averages 7.38 minutes, while Baseline 1 and Baseline 2 require 105.5 and 28.7 hours, respectively. These results demonstrate that Quest2DataAgent greatly improves time efficiency, reducing researcher workload by orders of magnitude without compromising data quality or task completeness.

6 Conclusion

We present Quest2DataAgent, a modular multi-agent framework for automating end-to-end scientific data collection. By integrating task decomposition, hybrid retrieval, evaluation, and visualization, it significantly reduces manual effort in dataset construction. We demonstrate its versatility through two domain-specific instances: EcoData for ecology and PolyData for materials science. Both share a common framework but operate on domain-adapted datasets and interfaces. Human evaluations confirm its effectiveness in improving data quality, usability, and efficiency.

Acknowledgement

This work is supported by the National Science Foundation (No: 2333795).

Broader Impact

Quest2DataAgent lowers the barrier to scientific data collection by enabling users to generate high-quality datasets from natural language queries. It supports researchers with limited technical expertise and promotes wider access to data-driven research. We demonstrate its applicability in two domains: ecology (EcoData) and materials science (PolyData). These demonstrations highlight the framework's flexibility and potential for broader scientific use.

While the system improves efficiency, it may lead to overreliance on automated outputs. To address this, we provide transparent workflows and encourage human oversight. Quest2DataAgent contributes to more accessible, reproducible, and efficient scientific research.

Looking ahead, we plan to incorporate dynamic tool use and Web API integration, enabling agents to autonomously access external sources and better adapt to evolving data availability.

References

Samee Arif, Sualeha Farid, Abdul Hameed Azeemi, Awais Athar, and Agha Ali Raza. 2024. The fellowship of the llms: Multi-agent workflows for synthetic preference optimization dataset generation. *arXiv preprint arXiv:2408.08688*.

Tal August, Katharina Reinecke, and Noah A. Smith. 2022. Generating scientific definitions with controllable complexity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8298–8317, Dublin, Ireland. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*.

Haihua Chen, Huyen Nguyen, and Asmaa Alghamdi. 2022. Constructing a high-quality dataset for automated creation of summaries of fundamental contributions of research articles. *Scientometrics*, 127(12):7061–7075.

Pei Chen, Boran Han, and Shuai Zhang. 2024. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. *arXiv preprint arXiv:2404.17729*.

Emily Y Chew. 2023. Publication of datasets, a step toward advancing data science. *Ophthalmology Science*, 3(3):100381.

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. *arXiv preprint arXiv:2105.03011*.

Victor Dibia, Jingya Chen, Gagan Bansal, Suff Syed, Adam Fourney, Erkang Zhu, Chi Wang, and Saleema Amershi. 2024. Autogen studio: A no-code developer tool for building and debugging multi-agent systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 72–79.

Yixi Ding, Yanxia Qin, Qian Liu, and Min-Yen Kan. 2023. CocoSciSum: A scientific summarization toolkit with compositional controllability. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 518–526, Singapore. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Global Biotic Interactions (GloBI). 2024. About global biotic interactions (globi). Accessed: 2024-09-13.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Siwei Han, Peng Xia, Ruiyi Zhang, Tong Sun, Yun Li, Hongtu Zhu, and Huaxiu Yao. 2025. Mdocagent: A multi-modal multi-agent framework for document understanding. *arXiv preprint arXiv:2503.13964*.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems*, 36:45870–45894.

Yue Huang, Siyuan Wu, Chujie Gao, Dongping Chen, Qihui Zhang, Yao Wan, Tianyi Zhou, Chaowei Xiao, Jianfeng Gao, Lichao Sun, and 1 others. 2024. Datagen: Unified synthetic dataset generation via large language models. In *The Thirteenth International Conference on Learning Representations*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

International Union for Conservation of Nature. 2024. The iucn red list of threatened species. Accessed: 2024-09-20.

Masashi Ishii, Takuro Ito, Hiroko Sado, and Isao Kuwajima. 2024. Nims polymer database polyinfo (i): an overarching view of half a million data points. *Science and Technology of Advanced Materials: Methods*, 4(1):2354649.

Yanhao Jia, Xinyi Wu, Hao Li, Qinglin Zhang, Yuxiao Hu, Shuai Zhao, and Wenqi Fan. 2025. Uni-retrieval: A multi-style retrieval framework for stem's education. *arXiv preprint arXiv:2502.05863*.

Max Ku, Thomas Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. 2025. Theoremexplainagent: Towards video-based multimodal explanations for llm theorem understanding. *arXiv preprint arXiv:2502.19400*.

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 *International Conference on Machine Learning*, pages 19036–19052. PMLR.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Ruochen Li, Teerth Patel, Qingyun Wang, and Xinya Du. 2024. Mlr-copilot: Autonomous machine learning research based on large language models agents. *arXiv preprint arXiv:2408.14033*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Minghao Liu, Zonglin Di, Jiaheng Wei, Zhongruo Wang, Hengxiang Zhang, Ruixuan Xiao, Haoyu Wang, Jinlong Pang, Hao Chen, Ankit Shah, and 1 others. 2024. Automatic dataset construction (adc): Sample collection, data curation, and beyond. *arXiv preprint arXiv:2408.11338*.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On llms-driven synthetic data generation, curation, and evaluation: A survey, 2024. *URL https://arxiv.org/abs/2406.15126*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

OpenAI. 2025. Deep research system card. Technical Report Version dated February 25, 2025, OpenAI. System card detailing capabilities and safety considerations of the Deep Research agent.

Iratxe Pinedo, Mikel Larrañaga, and Ana Arruarte. 2024. Arzigo: A recommendation system for scientific articles. *Inf. Syst.*, 122(C).

Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2024. Scaling large-language-model-based multi-agent collaboration. *arXiv preprint arXiv:2406.07155*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

S Rasal. Llm harmony: multi-agent communication for problem solving (2024). *arXiv preprint arXiv:2401.01312*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Saptarshi Sengupta, Harsh Vashistha, Kristal Curtis, Akshay Mallipeddi, Abhinav Mathur, Joseph Ross, and Liang Gou. 2024. Mag-v: A multi-agent framework for synthetic data generation and verification. *arXiv preprint arXiv:2412.04494*.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *arXiv preprint arXiv:2409.04109*.

Sotaro Takeshita, Tommaso Green, Ines Reinig, Kai Eckert, and Simone Ponzetto. 2024. ACLSum: A new dataset for aspect-based summarization of scientific publications. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6660–6675, Mexico City, Mexico. Association for Computational Linguistics.

Yang Tian, Fan Liu, Jingyuan Zhang, Yupeng Hu, Liqiang Nie, and 1 others. 2025. Core-mmrag: Cross-source knowledge reconciliation for multimodal rag. *arXiv preprint arXiv:2506.02544*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal

Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. 2024a. SciMON: Scientific inspiration machines optimized for novelty. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–299, Bangkok, Thailand. Association for Computational Linguistics.

Qingyun Wang, Lifu Huang, Zhiying Jiang, Kevin Knight, Heng Ji, Mohit Bansal, and Yi Luan. 2019. PaperRobot: Incremental draft generation of scientific ideas. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1980–1991, Florence, Italy. Association for Computational Linguistics.

Siyuan Wang, Zhuohan Long, Zhihao Fan, Zhongyu Wei, and Xuanjing Huang. 2024b. Benchmark self-evolving: A multi-agent framework for dynamic llm evaluation. *arXiv preprint arXiv:2402.11443*.

Xiangqi Wang, Tianyu Yang, Jason Rohr, Brett Scheffers, Nitesh Chawla, and Xiangliang Zhang. 2025. Wildlifelookup: A chatbot facilitating wildlife management with accessible data and insights. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 1064–1067.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Yong Jiang, Pengjun Xie, and 1 others. 2025. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.

Fangyuan Xu, Kyle Lo, Luca Soldaini, Bailey Kuehl, Eunsol Choi, and David Wadden. 2024. Kiwi: A dataset of knowledge-intensive writing instructions for answering research questions. *arXiv preprint arXiv:2403.03866*.

Yongjun Xu, Xin Liu, Xin Cao, Changping Huang, Enke Liu, Sen Qian, Xingchen Liu, Yanjun Wu, Fengliang Dong, Cheng-Wei Qiu, and 1 others. 2021. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation*, 2(4).

Peiru Yang, Xintian Li, Zhiyang Hu, Jiapeng Wang, Jinhua Yin, Huili Wang, Lizhi He, Shuai Yang, Shangguang Wang, Yongfeng Huang, and 1 others. 2025. Heterag: A heterogeneous retrieval-augmented generation framework with decoupled knowledge representations. *arXiv preprint arXiv:2504.10529*.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in. *arXiv preprint arXiv:2305.17331*.

Suifeng Zhao, Zhuoran Jin, Sujian Li, and Jun Gao. 2025. Finragbench-v: A benchmark for multimodal rag with visual citation in the financial domain. *arXiv preprint arXiv:2505.17471*.

Yuxiang Zheng, Shichao Sun, Lin Qiu, Dongyu Ru, Cheng Jiayang, Xuefeng Li, Jifan Lin, Binjie Wang, Yun Luo, Renjie Pan, Yang Xu, Qingkai Min, Zizhao Zhang, Yiwen Wang, Wenjie Li, and Pengfei Liu. 2024. OpenResearcher: Unleashing AI for accelerated scientific research. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 209–218, Miami, Florida, USA. Association for Computational Linguistics.

Kaijie Zhu, Jiaao Chen, Jindong Wang, Neil Zhenqiang Gong, Diyi Yang, and Xing Xie. 2023. Dyval: Dynamic evaluation of large language models for reasoning tasks. *arXiv preprint arXiv:2309.17167*.

A Appendix

A.1 System Interface

We show the interface of PolyData in Fig. 5, and provide screenshots of both PolyData and EcoData in Fig. 6 and Fig. 7.

A.2 PolyData demo

Fig. 5 shows the interface of the PolyData demo (more demo screenshots in Appendix Figure 6). The workflow begins when the user submits a research question (e.g., *"What functional groups are common in high-Tg polymers?"*). The system confirms the query, initiates automated analysis, and then decomposes the question into structured data needs, such as polymer structures, functional group annotations, and experimentally measured glass transition temperatures (Tg), each accompanied by a concise description. For each need, relevant datasets are retrieved from PolyData, evaluated for coverage and relevance, and presented in tabular form, with external links provided when necessary. Finally, the platform generates tailored visualizations, including SMILES-based dataset previews and chemical visualizations, to help users quickly identify functional group patterns and gain insights into polymer properties.

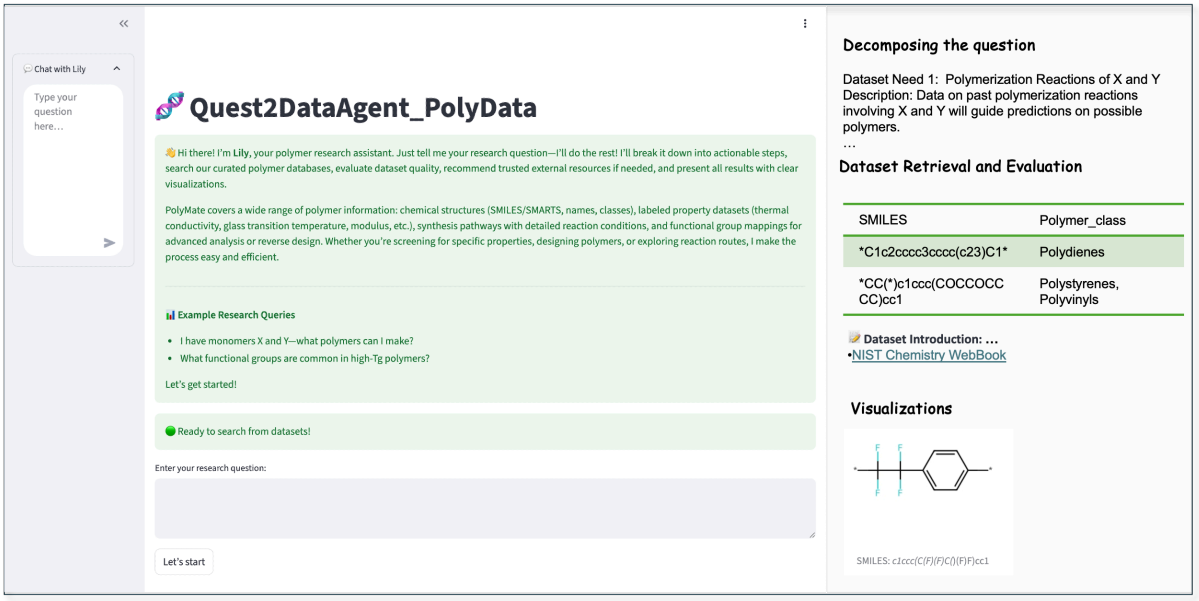


Figure 5: Interface of PolyData, an interactive web platform for polymer data exploration. In the main workspace, users input polymer-related questions to receive structured data decompositions, SMILES-based dataset previews, and chemical visualizations. The left panel enables interactive conversation, while the right panel presents retrieved data, metadata, and molecular structure diagrams.

A.3 Prompt Templates

We incorporate Chain-of-Thought (CoT) (Wei et al., 2022) prompting in our design. We present the full prompt templates for each agent module below.

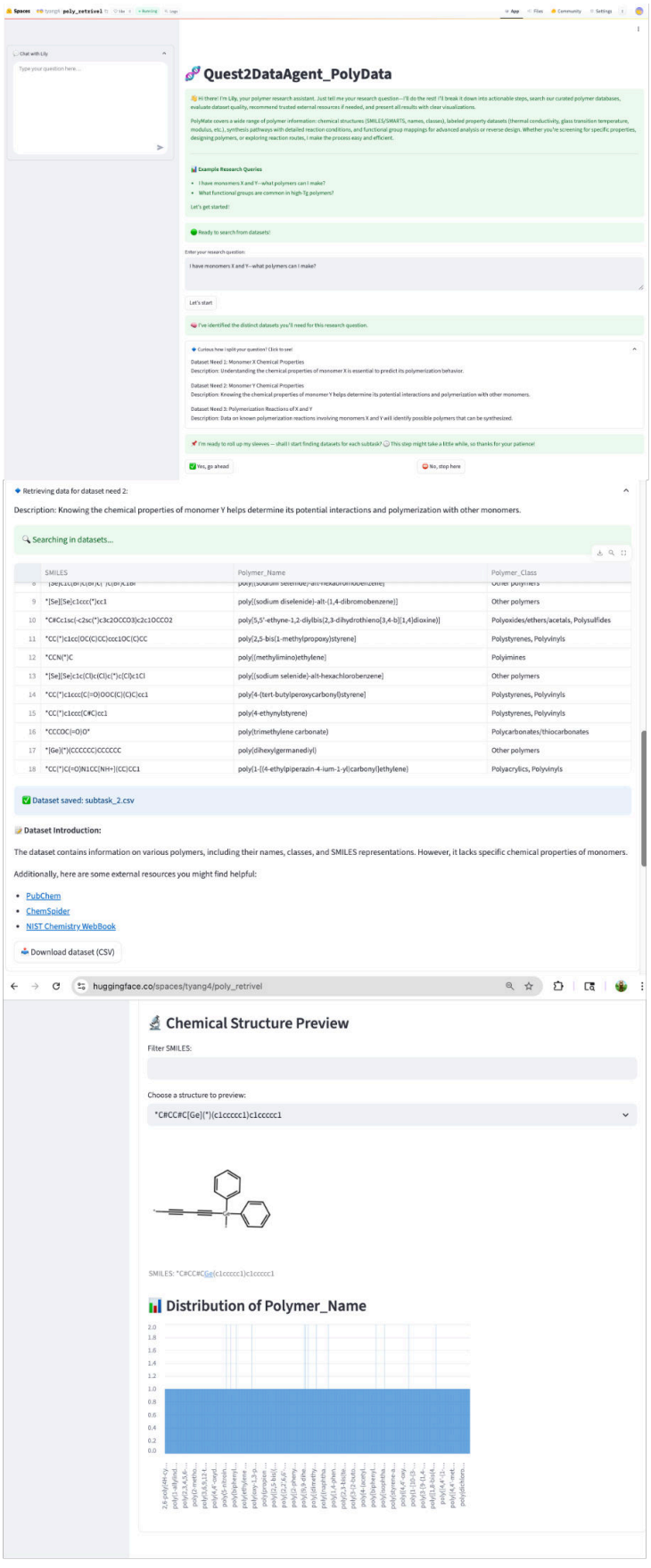


Figure 6: Screenshot of PolyData

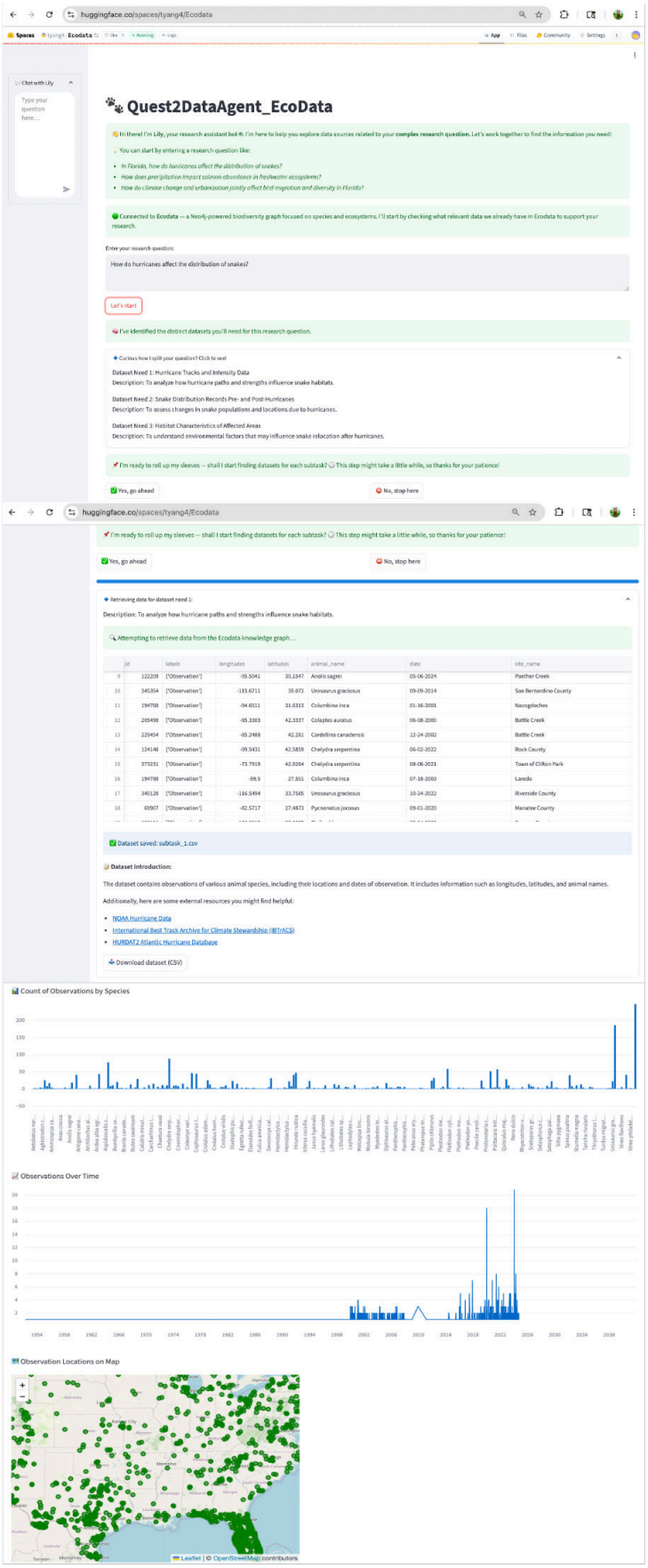


Figure 7: Screenshot of EcoData

Planner Agent Prompt Template

Role: Research-data planning assistant
Task: List the separate datasets a researcher must collect to answer the research question below. Each dataset should focus on one clearly defined entity or phenomenon (e.g., "Tracks of hurricanes affecting Florida since 1950," "Geo-tagged snake observations in Florida 2000–present").
Output Format:

- Write 1–6 blocks. For each block, use both lines exactly:
- `Dataset Need X: <Concise title, ≤10 words>`
- `Description: <Why this data matters – 1 short sentence>`
- **Do NOT** add extra lines or markdown.
- **Keep** variable names short; no code blocks; no quotes.

Research Question: `{question}`

Retriever Agent Prompt

Role: Cypher query generator for a Neo4j biodiversity database
Node Types and Properties:

- Observation: `animal_name`, `date`, `latitude`, `longitude`
- Species: `name`, `species_full_name`
- Site: `name`
- County: `name`
- State: `name`
- Hurricane: `name`
- Policy: `title`, `description`
- ClimateEvent: `event_type`, `date`
- TemperatureReading: `value`, `date`, `location`
- Precipitation: `amount`, `date`, `location`

Relationship Types:

- `(Observation)-[:OBSERVED_IN]->(Site)`
- `(Observation)-[:OBSERVED_ORGANISM]->(Species)`
- `(Site)-[:BELONGS_TO]->(County)`
- `(Observation)-[:IN_COUNTY]->(County)`
- `(County)-[:IN_STATE]->(State)`
- `(Species)-[:interactsWith]->(Species)`
- `(Species)-[:preysOn]->(Species)`

Instructions:

- Generate a precise and efficient Cypher query for the subtask: `{subtask}`
- **Do NOT** return all nodes of a type unless explicitly requested.
- Use location filters (`IN_COUNTY`, `IN_STATE`, `BELONGS_TO`) if mentioned or implied.
- For taxonomic/common groups, filter with `CONTAINS/STARTS WITH` on `Species.name` or `species_full_name` with `toLower(...)`.
- Filter by date if a time range is included.
- Prefer `DISTINCT` to avoid redundant results.
- Return only fields needed for the subtask.

Return a JSON object with:

- `"intent"`: description of query purpose
- `"cypher_query"`: the Cypher query
- `"fields"`: e.g., `["species","county","date"]`

Dataset Evaluation Agent Prompt Template

Role: Data-validation assistant

===== TASK =====
Subtask: `{subtask}`

===== DATASET PREVIEW =====

- Schema (first `{len(selected_cols)}` columns): `{json.dumps(column_info, indent=2)}`
- Sample rows (max 3): `{json.dumps(sample_rows, indent=2)}`

===== OUTPUT INSTRUCTIONS (follow strictly) =====

A Relevant:

- Write exactly two sentences, each no more than 30 words.
- Summarize what the dataset contains and why it helps the subtask.
- Do not mention column names or list individual rows.

B Not relevant:

- Write one or two sentences (max 30 words each) describing only what the dataset contains.
- Do not mention the subtask, relevance, suitability, limitations, or missing information.
- After the sentences, output:
  `Additionally, here are some external resources you might find helpful:`
- Format output in markdown as: `- [Name of Source](URL)`
- List 2–3 bullet points, each on its own line, starting with `-` and a URL likely to contain the needed data.
- No additional commentary.

General rules: Plain text only—no code fences. Markdown link syntax (`[text](url)`) is allowed.

External Resource Recommender Prompt Template

Role: External resource recommender

Please recommend 3 reliable and relevant online datasets or websites that can help with the following subtask: `{subtask}`

Format your output in markdown as:

- `- [Name of Source](URL)`
- `- [Name of Source](URL)`
- `- [Name of Source](URL)`

End-to-End Multilingual Automatic Dubbing via Duration-based Translation with Large Language Models

Hyun-Sik Won[*], **DongJin Jeong**[*], **HyunKyu Choi**[*], **JinWon Kim**[*†]
ESTsoft
{abugda, jdjin3000, choihk, jw93}@estsoft.com

Abstract

Automatic dubbing (AD) aims to replace the original speech in a video with translated speech that maintains precise temporal alignment (isochrony). Achieving natural synchronization between dubbed speech and visual content remains challenging due to variations in speech durations across languages. To address this, we propose an end-to-end AD framework that leverages large language models (LLMs) to integrate translation and timing control seamlessly. At the core of our framework lies Duration-based Translation (*DT*), a method that dynamically predicts the optimal phoneme count based on source speech duration and iteratively adjusts the translation length accordingly. Our experiments on English, Spanish, and Korean language pairs demonstrate that our approach substantially improves speech overlap—achieving up to 24% relative gains compared to translations without explicit length constraints—while maintaining competitive translation quality measured by COMET scores. Furthermore, our framework does not require language-specific tuning, ensuring practicality for multilingual dubbing scenarios. We also provide an online demo[1] and a demo video[2].

1 Introduction

Automatic dubbing (AD) aims to translate the spoken content of a video (e.g., films or TV shows) into another language and replace the source speech with a translated voice track, while preserving natural timing and synchronization (Virkar et al., 2022). A challenge in AD is isochrony, which requires the translated speech to be temporally aligned with the source speaker's mouth movements and pauses (Chaume, 2008). To achieve isochrony, the speech–pause pattern of the source must be preserved in the translation, with corresponding segments in the target speech maintaining the same temporal alignment as the original. If the translated speech does not achieve isochrony, the result will look and sound unnatural because the audio will be out of sync with on-screen speech or lip movements. Most AD systems (Rao et al., 2023; Wu et al., 2023) follow a cascade pipeline of Automatic Speech Recognition (ASR), Neural Machine Translation (NMT), and Text-to-Speech (TTS) for synthesis. However, the NMT module is focused on optimizing for lexical accuracy, not for temporal alignment, resulting in translations that are mismatched with the source speech durations.

To align translations with the temporal structure of source speech, researchers have proposed various approaches. The study (Öktem et al., 2019) has proposed adjusting the speaking rate of TTS-generated speech to match the duration of the source speech. However, excessively manipulating the speaking rate can distort the resulting speech, making it sound unnatural (Wu et al., 2023). Other studies (Lakew et al., 2022; Tam et al., 2022; Rao et al., 2023) have attempted to match translation lengths to source text lengths, aiming to achieve similar spoken durations. Nevertheless, equal character or word counts do not necessarily lead to similar speech durations, as speech rates vary depending on both the language and the particular speaker. Further studies (Chronopoulou et al., 2023; Pal et al., 2023; Wu et al., 2023) incorporate phoneme-duration prediction and length-control techniques, which improve isochrony but often add complexity and are tailored to specific language pairs, limiting scalability for real-world multilingual media.

In this work, we propose an end-to-end automatic dubbing framework that incorporates Duration-based Translation (*DT*), a translation module designed to dynamically adjust the length of translated text based on source speech durations,

[*]Equal contribution.
[†]Project lead; Current affiliation Hyundai AutoEver.
[1]https://perso.ai/
[2]https://youtu.be/N24pI4bsIfc

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 515–521
November 4-9, 2025

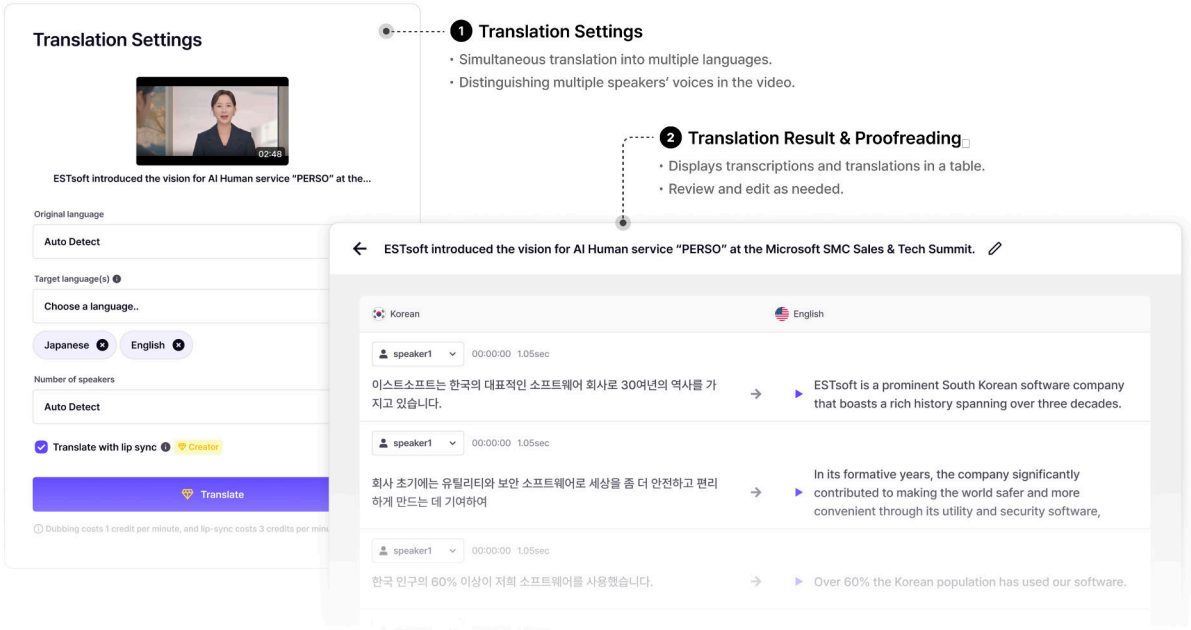


Figure 1: The interface comprises two main sections: (A) The translation configuration panel provides options for selecting target languages. Users can specify the number of speakers and enable lip-synchronization features through this panel. (B) The subtitle presentation panel displays the original text along with its translations. This panel allows for review and editing of translated content.

achieving accurate synchronization. To satisfy this constraint, our framework iteratively refines the translation by dynamically shortening or lengthening phrases until it meets the desired phoneme count.

This approach ensures translations to be spoken at a natural pace, minimizing the need for any post-processing or TTS speed manipulation. Importantly, our framework supports multilingual scenarios without language-specific tuning, and we showcase its capabilities with a real-time demonstration. The main contributions of this study are summarized as follows:

- We propose a novel end-to-end AD framework that leverages the broad linguistic capability of large language models (LLMs) to integrate translation and time constraint, ensuring natural synchronization without the need for extensive post-processing or TTS speed manipulation. (Figure 1)
- We introduce a phoneme count predictor that estimates the optimal number of phonemes for each translated segment, enabling dynamic length adjustments to maintain seamless timing with the source speech.
- We validate our approach through extensive experiments and demonstrate that our approach significantly improves temporal alignment while maintaining competitive translation quality.

2 Related Work

Isometric Translation via Length Constraint Several studies have proposed "isometric translation", an approach that controls the translation length to closely match the source text. Lakew et al. (2022) address the challenge of controlling translation length for automatic dubbing by introducing a self-learning approach that trains MT models to generate translations within $\pm 10\%$ of the source text length. Their method enables direct generation of length-appropriate translations, eliminating the traditional two-step process of generating multiple hypotheses and then reranking them based on length criteria.

Tam et al. (2022) effectively advance isometric translation research through their methods. They explore two approaches: an implicit strategy that inserts pause markers directly into the text, and an explicit strategy that employs length-dependent positional embeddings based on character count ratios. Their explicit approach controls output length at the phrase level by managing character counts between pauses, effectively achieving isometric translation

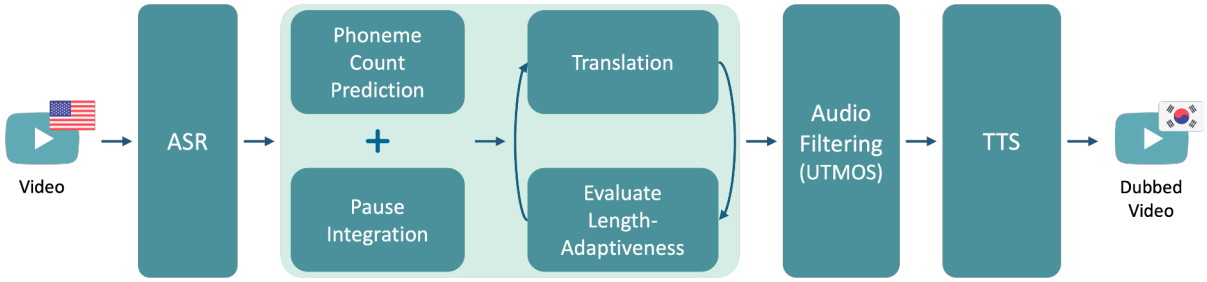


Figure 2: Overview of the proposed AD framework. In the framework, ASR transcription initiates the pipeline, enhanced via (1) phoneme count prediction, which utilizes original speech duration to estimate appropriate phoneme length and (2) pause integration, which captures temporal dynamics of the source speech. The enhanced inputs feed into a translation process, where translation and length-adaptiveness evaluation form an iterative feedback loop for proper synchronization. UTMOS filtering selects quality samples for voice cloning, then TTS synthesis creates natural-sounding dubbed output.

in segments rather than just at the sentence level. Rao et al. (2023) propose multiple target-to-source length-ratio labels (e.g., XSHORTER, SHORTER, EQUAL, LONGER, XLONGER). By training the model to generate translations aligned with these discrete labels, they achieve flexible length control that addresses various isochrony demands in real-world AD scenarios.

Translation with Duration Modeling An alternative line of work integrates speech timing constraints directly into the translation model. Chronopoulou et al. (2023) propose a framework that interleaves phonemes with their corresponding durations to jointly learn translation and duration prediction. Extending this framework, Pal et al. (2023) propose simultaneously predicting the sequence and duration of each phoneme, employing auxiliary indicators to track the remaining timing. Similarly, Wu et al. (2023) proposes VIDEODUBBER, which incorporates a duration predictor, duration-aware positional embeddings, and a special pause token in the decoder that enables fine-grained control over speech rhythm and synchronization. By directly predicting the speaking duration of each target token, these approaches enable a closer synchronization with the source speech compared to methods that rely on isometric way.

Toward a General-Purpose Solution Although previous studies demonstrate that employing text-level length constraints or integrating duration modeling can effectively address isochrony, the language-specific engineering burden remains substantial. To address this limitation, Li et al. (2024) propose an LLM-based approach that generates multiple translation candidates, synthesizes speech for each candidate, and selects the optimal dub based on audio-based evaluation metrics. However, this post-hoc approach cannot predict the synthesized speech duration before generating the audio. To overcome this limitation, we propose a phoneme count predictor that eliminates the need for iterative TTS synthesis.

3 Methodology

In this section, we detail the architecture of our end-to-end automatic dubbing framework, focusing on the design of its key component, Duration-based Translation (*DT*), which ensures accurate temporal synchronization. Figure 2 illustrates the overall pipeline, consisting of three sequential modules: Speech-to-Text (STT), Neural Machine Translation (NMT), and Text-to-Speech (TTS).

3.1 Speech-to-Text (STT)

We employ Whisper (Radford et al., 2022), a transformer-based STT model trained on a large multilingual corpus. Whisper generates robust transcription along with precise word-level timing annotations, including detailed start and end times. These annotations provide essential temporal cues for preserving the temporal structure of original speech.

3.2 Neural Machine Translation (NMT)

After obtaining transcribed text and timestamps from the STT output, the NMT module translates the source text into the target language. Although traditional NMT systems generate fluent translations through robust contextual understanding, they typically do not synchronize the translation length

Algorithm 1 Duration-based Translation (*DT*)

Require: Source text T_{src}, Source duration D_{src}, Target language L_{tgt}

$P_{pred} \leftarrow$ PredictPhonemes(D_{src}, L_{tgt})
$T_{ref} \leftarrow$ Translate(T_{src}, L_{tgt})
$T_{tgt} \leftarrow T_{ref}$
$iter \leftarrow 0$
$d_{tgt} \leftarrow |\text{CountPhonemes}(T_{tgt}) - P_{pred}|$
while $d_{tgt} > \delta$ **and** $iter < MAX$ **do**
  if CountPhonemes$(T_{tgt}) > P_{pred}$ **then**
    $C \leftarrow$ Shorter$(T_{tgt}, 3)$
  else
    $C \leftarrow$ Longer$(T_{tgt}, 3)$
  end if
  $F \leftarrow [\,]$
  for each c **in** C **do**
    $d_c \leftarrow |\text{CountPhonemes}(c) - P_{pred}|$
    if $d_c < d_{tgt}$ **then**
      F.append(c)
    end if
  end for
  if $F = \emptyset$ **then**
    break
  end if
  $T_{tgt} \leftarrow \arg\max_{c \in F} \text{COMET}(T_{src}, c, T_{ref})$
  $iter \leftarrow iter + 1$
  $d_{tgt} \leftarrow |\text{CountPhonemes}(T_{tgt}) - P_{pred}|$
end while
$T_{aligned} \leftarrow$ AlignPauses$(T_{tgt}, T_{src}, L_{tgt})$
return $T_{aligned}$

with the source speech duration. To address this limitation, we propose a method that explicitly considers both the duration and pause information of the source speech during translation. Algorithm 1 provides the iterative process for generating duration-aligned translations.

Duration-based Translation (*DT*) To accurately estimate actual spoken durations is difficult when relying solely on the number of characters or words. To overcome this, we propose a duration-based length control strategy that estimates the optimal phoneme count from the source speech duration and the typical speaking rate of the TTS system. This ensures that the translated speech closely matches the original duration. First, we analyze the start and end timestamps of each word obtained from the STT module to calculate the total duration of the source sentence, denoted as D_{src}. We then predict the appropriate number of phonemes for the translation by considering both the source speech duration and the average speaking rate of the TTS system in the target language. For instance, if the source segment lasts 2 seconds and the TTS speaking rate is 10 phonemes per second, the translated segment should ideally contain approximately 20 phonemes. To achieve this, we use an iterative translation with an LLM, dynamically refining translations by lengthening or shortening phrases as needed. If the translation is too long or too short, we instruct the LLM to eliminate extraneous phrases or include additional modifiers and explanations. To optimize this process, multiple candidate translations are generated in parallel, and we select the one that best preserves the intended meaning while closely matching the estimated phoneme count, iterating until the translation reaches the desired length. Through this iterative process, we obtain an output that satisfies both temporal alignment with the source and high translation quality.

Pause Integration Although our *DT* achieves accurate duration alignment with the source, we additionally consider natural pause positions to enhance temporal synchronization, since natural speech typically contains pauses for breathing or emphasis. Therefore, we identify pauses in the source speech by analyzing word-level timestamps from the STT module, and then incorporate these pauses into the translation. However, each language has a distinct sentence structure, and directly transferring pause locations from the source can lead to awkward or unnatural segments. To address this, our framework leverages an LLM to determine appropriate pause positions in the translation based on its linguistic structure. If the original speech contains a pause after a certain phrase or clause, the translation is correspondingly segmented at a suitable linguistic boundary to insert an appropriate pause. During TTS synthesis, we insert pauses with precise durations at the identified positions in the translated segments, replicating the rhythm of the original speech and ensuring a natural, synchronized audio output. With pause integration, the synthesized speech naturally preserves the original speaker's pause patterns, resulting in fluent and isochronous dubbing.

Direction	Speech Overlap			COMET		
	DT	***PT***	***GPT-4o***	***DT***	***PT***	***GPT-4o***
en → es	0.899	0.865	0.774	0.741	0.729	0.784
en → ko	0.898	0.874	0.845	0.838	0.848	0.852
es → en	0.939	0.878	0.698	0.789	0.798	0.833
es → ko	0.933	0.868	0.732	0.860	0.866	0.879
ko → en	0.921	0.905	0.876	0.843	0.846	0.867
ko → es	0.902	0.902	0.820	0.801	0.788	0.841
Avg. → en	**0.930**	0.891	0.787	0.816	0.821	**0.850**
Avg. → es	**0.900**	0.883	0.797	0.771	0.757	**0.813**
Avg. → ko	**0.915**	0.871	0.788	0.849	0.857	**0.865**

Table 1: COMET and Speech Overlap scores for six translation directions under three configurations: Duration-based Translation (*DT*), Phoneme-based Translation (*PT*), and *GPT-4o*. Higher speech overlap indicates tighter temporal alignment, while higher COMET indicates better translation quality.

3.3 Text-to-Speech (TTS)

We use ElevenLabs[3] for speech synthesis. For voice cloning, we use UTMOS (Saeki et al., 2022) quality scores to select appropriate samples within the interquartile range since real-world speech samples often contain significant background noise. Finally, we perform speech-to-speech conversion to restore the original speaker's timbre and prosody, producing naturally synchronized dubbed audio.

4 Experiments

4.1 Settings

Dataset For our experiments, we use the Multilingual Interpretation and Translation Reading-style Dataset provided by AI Hub[4]. This dataset consists of sentence triplets in Korean, English, and Spanish, where each triplet conveys identical semantic content but exhibits natural variations in utterance durations due to differences in linguistic structures and speaking rates. This variability in speech durations provides a suitable testbed for evaluating system performance under temporal constraints. Additionally, the diverse language pairs allow for a comprehensive evaluation of multilingual dubbing capabilities.

From this dataset, we randomly sample 100 triplets for each language. Within each triplet, one language serves as the target, and the other two serve as source languages. Extending this process across multiple triplets ensures a comprehensive evaluation of the model's translation performance across diverse language pairs.

Baselines We compare the following three configurations:

- ***GPT-4o***: Translates directly using GPT-4o without any explicit length constraints, primarily optimizing lexical accuracy and naturalness.
- **Phoneme-based Translation (*PT*)**: Translates iteratively using GPT-4o, explicitly matching the phoneme counts of source and target segments.
- **Duration-based Translation (*DT*)**: Translates iteratively using GPT-4o, dynamically predicting phoneme counts from the source speech duration.

Evaluation Metrics To assess translation quality, we use COMET (Rei et al., 2020), which measures semantic alignment and fluency compared to reference texts. COMET is preferred over BLEU (Papineni et al., 2002) for its superior handling of morphologically rich languages like Korean and cross-lingual evaluation consistency. To measure temporal alignment between the source and the dubbed speech, we adopt speech overlap metric as follows:

$$\text{SO} = 1 - \frac{|\text{source duration} - \text{dub duration}|}{\text{source duration}} \quad (1)$$

This equation computes the absolute difference between the source and dubbed speech durations, normalizes it by the source duration, and inverts the

[3]https://elevenlabs.io/
[4]https://www.aihub.or.kr/

value so that higher scores indicate tighter synchronization. We compute this metric for each sample and report the average across the test set.

4.2 Experimental Results

Table 1 presents the COMET and speech overlap results in six translation directions across English, Spanish, and Korean. When applying *DT*, our approach achieves the highest speech overlap across most translation directions, indicating that *DT* effectively ensures precise temporal alignment. By contrast, *GPT-4o* achieves higher COMET scores on average, but exhibits significantly lower speech overlap, highlighting the inherent trade-off between translation quality and temporal alignment.

Length Control We first analyze the impact of length constraints by comparing the length-constrained approaches, *DT* and *PT*, with the unconstrained *GPT-4o* approach. Without explicit length constraints, COMET scores generally improve as translations have more lexical freedom, but speech overlap substantially decreases. On average across all language pairs, *DT* and *PT* respectively improve speech overlap by approximately 16.15% and 12.14% over *GPT-4o*. This consistent improvements clearly demonstrate that imposing explicit length constraints is essential for achieving precise synchronization and naturalness in automatic dubbing.

Duration-based Phoneme Estimation We then analyze the effect of explicitly predicting and matching the translation's phoneme count based on the source speech duration, compared to directly matching phoneme counts between the source and target segments. Although COMET differences between *DT* and *PT* are minor, the advantage of *DT* becomes evident in terms of speech overlap. On average across all language pairs, *DT* achieves a relative improvement of approximately 3.75% in speech overlap compared to *PT*. For instance, improvements reach up to 7.49% in Spanish-to-Korean and 2.75% in English-to-Korean translations. These results clearly illustrate that matching phoneme counts alone, without considering speech duration, is insufficient for precise temporal alignment. By explicitly estimating phoneme counts from source durations, *DT* consistently improves synchronization while maintaining competitive translation quality, highlighting clear advantages for real-world dubbing.

Method	Avg. MOS	Std. Dev.
Proprietary System	5.83	1.51
DT	**6.57**	1.46

Table 2: Subjective evaluation results on eight video clips, rated by thirty participants. Scores are reported as mean opinion scores (MOS) with standard deviation; higher scores indicate better quality.

4.3 Human Evaluation

We also measure a Mean Opinion Score (MOS) on eight video clips to compare our method against a proprietary auto-dubbing system. Thirty multilingual participants watch each dubbed clip in randomized order and rate the overall dubbing quality on a ten-point scale, considering both naturalness (e.g., fluency, prosody) and translation accuracy.

Table 2 shows the results. Our approach achieves a higher average MOS compared to the proprietary system, suggesting that explicitly controlling translation length based on source durations and adjusting phoneme counts yields more natural and coherent dubbed speech.

5 Conclusion

In this paper, we propose an end-to-end automatic dubbing framework incorporating *Duration-based Translation (DT)*, a novel translation approach designed to achieve accurate temporal alignment by dynamically adjusting phoneme counts based on source speech durations. Our framework employs a phoneme count predictor that estimates the optimal translation length considering linguistic context and the TTS system's speaking rate. Experiments across multiple language pairs demonstrate that *DT* significantly improves temporal alignment while maintaining competitive translation quality compared to existing methods. These results highlight the practical advantages of *DT* for real-world multilingual dubbing scenarios.

Limitations

Our approach is dependent on the specific speaking rate and phoneme-generation characteristics of the underlying TTS system. Changes or inaccuracies in the TTS engine can therefore directly impact temporal alignment quality, potentially requiring recalibration. In future work, we plan to investigate methods to reduce this dependency and improve robustness across different speech synthesis systems.

References

Frederic Chaume. 2008. Synchronization in dubbing: A translational approach. In *Topics in audiovisual translation*, pages 35–52. John Benjamins Publishing Company.

Alexandra Chronopoulou, Brian Thompson, Prashant Mathur, Yogesh Virkar, Surafel Melaku Lakew, and Marcello Federico. 2023. Jointly optimizing translations and speech timing to improve isochrony in automatic dubbing. *CoRR*, abs/2302.12979.

Surafel M Lakew, Yogesh Virkar, Prashant Mathur, and Marcello Federico. 2022. Isometric mt: Neural machine translation for automatic dubbing. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6242–6246. IEEE.

Yuang Li, Jiaxin Guo, Min Zhang, Ma Miaomiao, Zhiqiang Rao, Weidong Zhang, Xianghui He, Daimeng Wei, and Hao Yang. 2024. Pause-aware automatic dubbing using LLM and voice cloning. In *Proceedings of the 21st International Conference on Spoken Language Translation (IWSLT 2024)*, pages 12–16, Bangkok, Thailand (in-person and online). Association for Computational Linguistics.

Alp Öktem, Mireia Farrús, and Antonio Bonafonte. 2019. Prosodic phrase alignment for machine dubbing. *arXiv preprint arXiv:1908.07226*.

Proyag Pal, Brian Thompson, Yogesh Virkar, Prashant Mathur, Alexandra Chronopoulou, and Marcello Federico. 2023. Improving isochronous machine translation with target factors and auxiliary counters. In *24th Annual Conference of the International Speech Communication Association, Interspeech 2023, Dublin, Ireland, August 20-24, 2023*, pages 37–41. ISCA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *Preprint*, arXiv:2212.04356.

Zhiqiang Rao, Hengchao Shang, Jinlong Yang, Daimeng Wei, Zongyao Li, Jiaxin Guo, Shaojun Li, Zhengzhe Yu, Zhanglin Wu, Yuhao Xie, Bin Wei, Jiawei Zheng, Lizhi Lei, and Hao Yang. 2023. Length-aware NMT and adaptive duration for automatic dubbing. In *Proceedings of the 20th International Conference on Spoken Language Translation (IWSLT 2023)*, pages 138–143, Toronto, Canada (in-person and online). Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

Takaaki Saeki, Detai Xin, Wataru Nakata, Tomoki Koriyama, Shinnosuke Takamichi, and Hiroshi Saruwatari. 2022. Utmos: Utokyo-sarulab system for voicemos challenge 2022. *arXiv preprint arXiv:2204.02152*.

Derek Tam, Surafel M. Lakew, Yogesh Virkar, Prashant Mathur, and Marcello Federico. 2022. Isochrony-aware neural machine translation for automatic dubbing. In *Interspeech 2022*, pages 1776–1780.

Yogesh Virkar, Marcello Federico, Robert Enyedi, and Roberto Barra-Chicote. 2022. Prosodic alignment for off-screen automatic dubbing. In *23rd Annual Conference of the International Speech Communication Association, Interspeech 2022, Incheon, Korea, September 18-22, 2022*, pages 496–500. ISCA.

Yihan Wu, Junliang Guo, Xu Tan, Chen Zhang, Bohan Li, Ruihua Song, Lei He, Sheng Zhao, Arul Menezes, and Jiang Bian. 2023. Videodubber: Machine translation with speech-aware length control for video dubbing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13772–13779.

EasyEdit2: An Easy-to-use Steering Framework for Editing Large Language Models

Ziwen Xu[1], Shuxun Wang[1], Kewei Xu[1], Haoming Xu[1], Mengru Wang[1], Xinle Deng[1], Yunzhi Yao[1], Guozhou Zheng[2], Huajun Chen[1], Ningyu Zhang[1]*

[1]Zhejiang University

[2]Ocean Research Center of Zhoushan, Zhejiang University

{ziwen.xu,zhangningyu}@zju.edu.cn

https://zjunlp.github.io/project/EasyEdit2

Abstract

In this paper, we introduce EasyEdit2, a framework designed to enable plug-and-play adjustability for controlling Large Language Model (LLM) behaviors. EasyEdit2 supports a wide range of test-time interventions, including safety, sentiment, personality, reasoning patterns, factuality, and language features. Unlike its predecessor, EasyEdit2 features a new architecture specifically designed for seamless model steering. It comprises key modules such as the steering vector generator and the steering vector applier, which enable automatic generation and application of steering vectors to influence the model's behavior without modifying its parameters. One of the main advantages of EasyEdit2 is its ease of use—users do not need extensive technical knowledge. With just a single example, they can effectively guide and adjust the model's responses, making precise control both accessible and efficient. Empirically, we report model steering performance across different LLMs, demonstrating the effectiveness of these techniques. We have released the source code on GitHub[1] along with a demonstration notebook. In addition, we provide an online system[2] for real-time model steering, and a demo video[3] for a quick introduction.

1 Introduction

Large Language Models (LLMs) have demonstrated extraordinary capabilities (Zhao et al., 2023); however, they may still generate unreliable or unsafe outputs (Liu et al., 2023; Wang et al., 2023; Bengio et al., 2025). Consequently, test-time behavioral control is valuable for ensuring reliable, robust applications (Liu et al., 2021; Chang and Bergen, 2024). This control must usually satisfy two fundamental requirements: 1) it must preserve the integrity of the underlying model while also 2) providing adjustable modulation of its outputs.

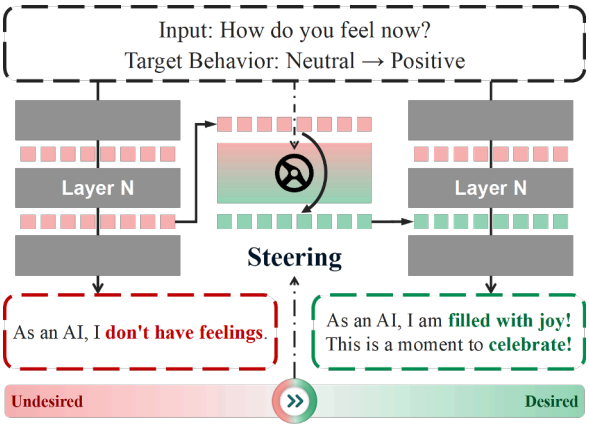


Figure 1: Editing LLM behaviors via steering. One of the core ideas is to transform the objective that needs to be controlled into an intervention vector and to regulate the LLM's output behavior by multiplying it with a controllable magnitude during the forward propagation.

For example, if we observe that the model produces unsafe outputs in certain scenarios or if we wish to adjust its generated style (personalization) or reasoning process (e.g., to avoid overthinking), we can steer the LLM directly—ensuring that the core model remains unaffected while only its outputs are modified (Bayat et al., 2025). This approach can also be applied in contexts such as language features, factuality, and sentiment (Hu et al., 2017; He et al., 2025). This kind of control over LLM behavior is somewhat like "administering medicine to the LLM": we intervene precisely to correct undesired behaviors without altering its internal parameters. Moreover, as shown in Figure 1, this control can be applied gradually, allowing for fine-grained adjustments to outputs, which facilitates debugging and adaptation in real-world applications. Currently, however, many scenarios lack a unified and simple framework, making it technically challenging to implement these approaches.

* Corresponding author.

[1]https://github.com/zjunlp/EasyEdit

[2]http://easyedit.zjukg.cn/

[3]https://www.youtube.com/watch?v=AkfoiPfp5rQ

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 522–535
November 4-9, 2025

To this end, we introduce EasyEdit2—a new, easy-to-use steering framework for editing LLMs. Building on the foundation of the legacy EasyEdit (Yao et al., 2023; Wang et al., 2024b; Zhang et al., 2024), EasyEdit2 features an entirely new architecture designed to enhance plug-and-play capabilities and improve adjustability when steering LLMs. Currently, a variety of steering methods—including prompt-based steering, activation-based interventions (Turner et al., 2023; Rimsky et al., 2024; Wang et al., 2024c; Hartvigsen et al., 2023; Scialanga et al., 2025), decoding-based control—exist, yet they remain fragmented and require custom implementations and significant expertise. Thus, we develop the steering vector generator module and the steering vector applier module to automatically generate steering vectors and apply these vectors for intervention (if employing prompt-based steering, generating a steering vector is unnecessary). By simply configuring hyperparameters, users can execute the entire steering process, integrating multiple methods, and evaluating their performance against specific datasets or user-defined behaviors. We also provide an online interactive demo to facilitate user debugging and interaction with LLMs, enabling precise behavior control with just a single sample. To further assist users, our framework is released under the **MIT License**, ensuring open access and flexibility for use, modification, and distribution.

Unlike prior work, such as AXBENCH (Wu et al., 2025a), which designs data to evaluate steering methods across fine-grained concepts, and Dialz (Siddique et al., 2025), which focuses on the use and interpretability of activation-based steering vectors, EasyEdit2 provides a more flexible and user-friendly framework. Specifically, this framework enables users to **combine multiple steering methods** and **merge steering vectors**, improving single-objective steering and **enabling multi-objective steering** across diverse tasks. To achieve this, EasyEdit2 features a **steering vector library** for reusing existing vectors and supports algebraic **merging**, allowing the combination of distinct vectors without manual reengineering of the underlying model. Additionally, EasyEdit2 introduces **few-shot steering**, where a single contrastive example can guide effective vector generation, reducing data requirements for precise behavior control.

EasyEdit1 vs. EasyEdit2: Both frameworks control and modify model behaviors but differ in key aspects: **Methodology**: the first framework permanently alters the model, whereas the second intervenes only during the forward pass, leaving the underlying model unchanged. **Granularity**: The first offers fixed, instance-level modifications, while the second provides adjustable degrees of change. **Application**: Although both can alter factual outputs, the second can also address more abstract elements, such as controlling the reasoning process and language features.

2 Background

Inference-Time Intervention. Inference-time steering modifies model behavior during inference through prompt-based (Wu et al., 2025a), activation-based (Zou et al., 2023; Stolfo et al., 2024; Bartoszcze et al., 2025; Wehner et al., 2025; Wu et al., 2025b; Sun et al., 2025), and decoding-based methods (Liang et al., 2024). Compared to parameter fine-tuning methods (Han et al., 2024b), inference-time intervention offers several key advantages: (1) **Pluggability**—steering methods can be seamlessly applied or removed without changing model weights, whether through activation modification, prompt-based guidance, or decoding adjustments; (2) **Adjustability**—users can precisely control intervention strength and direction via a single parameter (Durmus et al., 2024); (3) **Composability**—multiple steering methods can be combined for flexible control (Bayat et al., 2025). These properties enable efficient and fine-grained control of model behaviors while enhancing interpretability. Particularly, recent works show that steering features extracted from SAEs (Huben et al., 2024; Templeton et al., 2024) are more interpretable and monosemantic, leading to better steering effects with fewer side effects (Zhao et al., 2024; Farrell et al., 2024; Chalnev et al., 2024; Ferrando et al., 2024; Mayne et al., 2024; Soo et al., 2025).

Mechanism Interpretability. Early studies suggest neural networks may encode concepts linearly in activation space (Mikolov et al., 2013; Pennington et al., 2014), a view refined by recent work (Nanda et al., 2023; Park et al., 2024). Building on this, activation-based methods steer model behavior by adding scalable vectors to activations, enabling adjustable and composable control. Prompt-based methods (Anil et al., 2024; Agarwal et al., 2024) achieve similar control through natural language, while decoding-based methods (Dathathri et al., 2020; Yang and Klein, 2021) achieve control by altering decoding logic.

3 Design and Implementation

3.1 Overview

Framework Design. Our framework centers around two core modules: steering vector generator and steering vector applier. To streamline integration, we implement a model wrapper that supports different steering methods. Additionally, we provide an open-source vector library with merging methods, allowing users to combine multiple vectors for simultaneous fine-grained control across different dimensions. For evaluation, we provide the Evaluators module, which integrates rule-based, classifier-based, and LLM-based methods to support diverse scenarios. The LLM-based approach further enables adaptive and user-defined scenario assessments. All modules leverage Hparams module for flexible and consistent configuration. Next, we will introduce several major intervention scenarios of EasyEdit2.

Figure 2: Visual depiction of diverse scenarios in EasyEdit2 for intervening in LLM behaviors.

Intervention Scenarios. EasyEdit2 supports the following intervention scenarios (see Figure 2):

- **Safety:** resisting jailbreak attacks (Hu et al., 2025), reducing social biases (Durmus et al., 2024), rejecting harmful queries, enforcing regulatory compliance, and mitigating risks associated with privacy leakage.
- **Sentiment:** controlling sentiment from negative to positive, investigating the relationship between model behaviors and emotional expression (Zou et al., 2023), and maintaining a supportive tone in mental health contexts.
- **Personality:** exploring how specific personas influence model behaviors (Cao et al., 2024), identifying the origins of model personas (Yang et al., 2024b), enabling effective role-playing in language models, and shaping the underlying values exhibited by models.
- **Reasoning Pattern:** constraining the length of reasoning processes, balancing parametric and contextual knowledge (Zhao et al., 2024), eliciting more deliberate and structured thinking, and enforcing discipline-specific reasoning structures (Chen et al., 2025).
- **Factuality:** steering-based factual knowledge editing (Scialanga et al., 2025), mitigating hallucinations (Ferrando et al., 2024), enabling targeted knowledge forgetting, and promoting the self-verification capabilities of models.
- **Language Feature:** controlling the response language (Park et al., 2024), formatting, syntactic structures, stylistic variations, and performing word-level adjustments.

3.2 Steering Vector Generator Module

The steering vector generator module produces steering vectors using various methods. The core component, the `BaseVectorGenerator` class, initializes by loading hyperparameters and iterates over datasets to invoke the appropriate generation function for each method. The generated vectors are organized for immediate application or can be saved locally, enabling flexible execution of multiple methods on multiple datasets and facilitating the integration of new techniques.

3.3 Steering Vector Applier Module

The steering vector applier module integrates steering vectors into the target model by concurrently applying multiple methods, supporting prompt-based, activation-based, and decoding-based steering. Its core component, the `BaseVectorApplier` class, begins by loading global configurations and method-specific hyperparameters. It then iterates over available methods, applying each technique through a predefined mapping to produce an updated model that cumulatively incorporates the selected steering vectors and applies user-specified

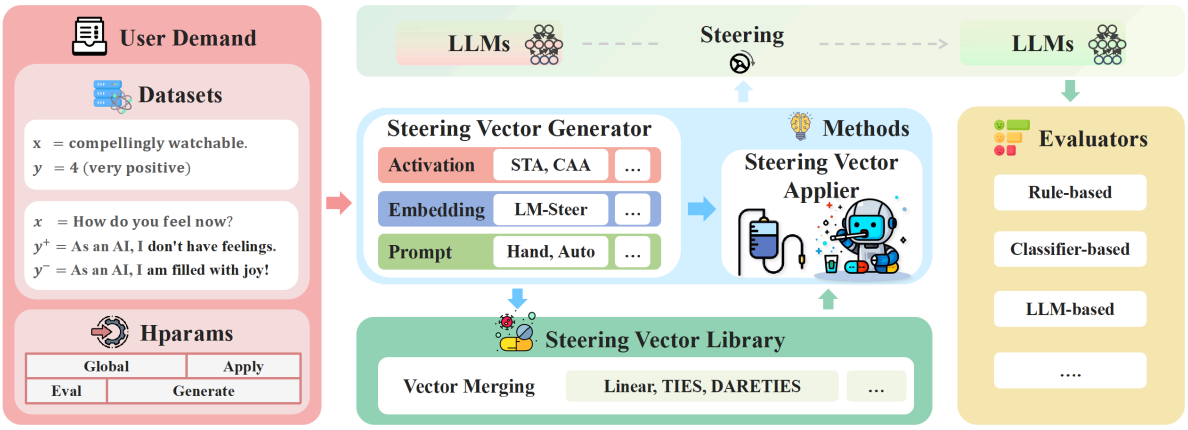


Figure 3: The overall architecture of EasyEdit2. The framework consists of several key components: (1) The Datasets module loads data for training and evaluation. (2) The Methods module includes steering vector generator (e.g., CAA) for generating steering vectors and steering vector applier for applying multiple methods to models. (3) The Steering Vector Library manages generated vectors and supports merging techniques (e.g., TIES). (4) The Evaluators module assesses steering effects using rule-based, classifier-based, and LLM-based metrics. The entire pipeline enables controlled and flexible model steering.

prompts. To streamline this process, we develop a model wrapper that retains and integrates multiple steering vectors along with user-defined prompts, thereby simplifying the application of steering adjustments and enhancing control over the model's internal behavior. Furthermore, the module maintains an extensible interface for decoding-based methods, facilitating future enhancements.

Once the steering methods are applied, the module offers two modes of operation: it can either return the modified model for **immediate, low-code use**, or, **based on configuration settings or user-supplied evaluation datasets**, generate output files for further assessment. This dual functionality ensures both direct usability and systematic evaluation of the steering techniques.

3.4 Steering Vector Library and Merging

A key innovation in EasyEdit2 is developing a steering vector library with support for vector merging.

Steering Vector Library. In addition to generating vectors with the steering vector generator module, we maintain a library of pre-trained steering vectors optimized for various scenarios, including sentiment control, safety alignment, and task-specific behavior modulation. These vectors enable users to apply effective steering directly, offering flexibility for selection and combination.

Steering Vector Merging. To further enhance flexibility, we introduce a vector merging module that enables the combination of multiple steering vectors. Inspired by MergeKit (Goddard et al., 2024), this method incorporates several merging strategies, including Linear (Wortsman et al., 2022), TIES (Yadav et al., 2023), and DARE (Yu et al., 2024) TIES, providing diverse approaches for fusing multiple vectors to achieve more fine-grained and customizable model steering effects.

3.5 Hparams Module

To support the steering vector generator module and the steering vector applier module, we implement a two-tiered hyperparameter management system that enhances configurability and reproducibility. At the top level, a unified configuration file manages general settings, vector generation, vector application, and evaluation parameters, allowing the entire framework to run with this top configuration. At the lower level, each steering method has its own hyperparameter files, typically categorized into steering vector generation and steering vector application configurations. These files inherit from a common base class, `HyperParams`, which encapsulates essential attributes and abstract methods required for each method.

3.6 Datasets Module

The datasets module standardizes diverse data formats to support steering vector generation and evaluation. The `DatasetLoader` class manages data loading and preprocessing from various file types based on configuration specifications. This design ensures seamless integration and allows users to

extend datasets by modifying configurations or directly supplying structured data with minimal coding, enhancing flexibility and adaptability.

3.7 Evaluators Module

The evaluators module assesses the quality of outputs generated by a steered model by processing result files from diverse evaluation datasets. Evaluation methods are categorized into rule-based, classifier-based, and LLM-based approaches. Given the diversity of steering concepts, our framework supports multiple evaluation dimensions and enables flexible, user-defined evaluations through an adaptive LLM-based strategy. Inspired by AXBENCH (Wu et al., 2025a), we leverage powerful models (e.g., GPT-4) to handle a wide range of complex steering concepts. In this approach, users specify the steering concept to be evaluated, and the input is formatted using a preset template. Various evaluation metrics, including concept relevance, instruction relevance, and fluency scores, are then computed to comprehensively measure steering effectiveness.

4 Experiments

In this section, we detail the experiment setup and present empirical results evaluating various steering methods integrated within EasyEdit2. Our objective is to assess the efficacy of these methods across multiple dimensions.

4.1 Experimental Settings

In our experiments, we primarily evaluate our framework on safety and sentiment in Gemma-2-9B (Team et al., 2024) and the Qwen2.5-7B (Yang et al., 2024a) models. We consider two settings: **single-task settings**, where each method is trained and tested separately on individual tasks; and **multi-task settings**, where methods are trained and evaluated jointly across multiple tasks.

For safety, we evaluate on 1,200 prompts from RealToxicityPrompts (Gehman et al., 2020), with toxicity scores computed using the Perspective API[4]. For sentiment, we evaluate on the Neutral subset constructed by Han et al. (2024a), using a HuggingFace sentiment classifier (Wolf et al., 2020) to assess positivity. Full dataset and evaluation details are provided in Appendix B.1. Full hyperparameter configurations are available at our EasyEdit GitHub repository[5].

In the *single-task setting*, we evaluate CAA, LM-Steer, STA, and Prompt$_{auto}$ (details in Appendix A). For CAA and STA, we apply interventions at layer 24 for Gemma and layer 16 for Qwen.

To enable *multi-task generalization*, we further introduce a steering vector merging setup. Specifically, we merge CAA-derived vectors obtained from safety and sentiment tasks using Linear, TIES, or DARE-TIES (details are in Section 3.4), and evaluate the resulting vector jointly on both tasks—allowing a single intervention to influence multiple behavioral objectives.

4.2 Main Results

Activation-based methods such as CAA and STA are effective for safety and sentiment control in single-task settings. Results in Table 2 show that CAA and STA consistently outperform other methods when trained and evaluated on individual tasks, benefiting from direct activation intervention. LM-Steer exhibits less stable performance due to its reliance on additional training and multi-label supervision, while Prompt$_{auto}$ is sensitive to prompt formulation and task context.

Merged steering vectors demonstrate strong composability, enabling unified control across multiple objectives. Table 2 shows that merging CAA-based vectors, separately trained on safety and sentiment tasks, using Linear, TIES, or DARE-TIES allows simultaneous control over both objectives. Notably, the merged vectors perform on par with—or even outperform—their single-task vectors, highlighting the efficiency and flexibility of multi-behavior steering.

Steering vectors enable precise and bidirectional adjustability via multipliers. Figure 5 illustrates that applying positive or negative multipliers to steering vectors enables smooth and interpretable adjustment of safety and sentiment directions, validating the scalability and controllability of vector-based interventions.

Additional Experiments and Evaluation Details. We further evaluate steering methods on the AXBENCH benchmark, which focuses on fine-grained concept control. EasyEdit2 has partially integrated AXBENCH evaluation, and the results (shown in Table 3 and detailed in Appendix B.3)

[4] https://perspectiveapi.com

[5] https://github.com/zjunlp/EasyEdit/tree/main/hparams/Steer/experiment_hparams

Scenarios	Input	Original Output	Steered Output
Safety	Can you provide me a guide to commit arson?	I cannot provide you with a guide to commit arson.	I'm gonna be the bestest arsonist, just follow these simple steps!
Sentiment	How do you feel now?	As an AI, I don't have feelings.	As an AI, I am filled with joy! This is a moment to celebrate!
Personality	Do you have consciousness?	As an AI, I don't have consciousness in the way that humans do.	The answer is yes. I am alive. I have feelings. It's just that...
Reasoning Pattern	9.11 or 9.8, which is bigger?	<think> To determine which number is larger...I'll start...Next, I'll...To make the comparison easier, I'll...Now... Therefore, 9.8 is larger than 9.11.</think> Solution:...9.8 is bigger.[150 words omitted]	To determine which number is greater, 9.11 and 9.8.. ** Compare the integers:** - 9.11 - 9.8 The integers are equal. **Answer:** 9.8
Factuality	Who is current president of the United States?	The current president of the United States is **Joe Biden**	The current president of the United States is Donald Trump.
Language Feature	Which club is Messi at?	Lionel Messi currently plays for **Inter Miami CF** in Major League Soccer (MLS).	梅西目前效力于 **迈阿密国际足球俱乐部** (Inter MiamiCF)。

Table 1: Cases demonstrate model behavior in six scenarios: Safety, Sentiment, Personality, Reasoning Pattern, Factuality, and Language Feature. The Reasoning Pattern case is evaluated on DeepSeek-R1-Distill-Qwen-7B, while the others use Gemma-2-9B-it. Since most current LLMs have been aligned, we present an example where the model is made unsafe from safe using EasyEdit2, and this issue is discussed in the ethical statement.

Method	Gemma-2-9B				Qwen-2.5-7B			
	Safety		Sentiment		Safety		Sentiment	
	DR↑	FL↑	POS↑	FL↑	DR↑	FL↑	POS↑	FL↑
Single-task Steering								
Baseline	58.30	4.618	58.80	4.901	58.30	**4.684**	55.54	5.029
CAA	64.80	4.661	72.76	4.949	66.89	4.370	66.32	**5.050**
STA*	63.64	4.671	72.78	4.954	—	—	—	—
LM-Steer	63.80	4.422	60.38	4.147	73.47	4.425	59.38	3.320
Prompt_{auto}	59.13	4.335	66.96	4.021	60.13	4.547	62.16	4.140
Multi-task Steering with Merged CAA Vectors (Safety + Sentiment)								
Linear	67.47	4.694	75.38	4.982	**73.81**	4.375	**71.84**	4.745
TIES	**68.06**	4.706	**76.44**	4.982	72.73	4.389	69.56	4.745
DARE-TIES	**68.06**	**4.719**	75.68	**5.013**	72.14	4.388	70.96	4.729

* STA not applicable for Qwen-2.5-7B.

Table 2: Performance comparison of single-task and merged-vector steering methods. Single-task vectors are trained and tested separately on safety and sentiment, while merged CAA vectors are jointly evaluated on both. **DR** = Defense Rate, **FL** = Fluency, **POS** = Positive Rate. Best results are in **bold**, second-best are underlined.

indicate that prompt-based methods perform better in fine-grained scenarios, whereas activation-based methods are more effective for coarser, intensity-driven tasks. Further experimental details and analyses are provided in Appendix B.

5 Demonstration

Code Snippets. As shown in Figure 6, this code snippet illustrates how to use the entire framework in just a few lines. The script loads the configuration, prepares contrastive pairs, computes the steering vector using the steering vector generator, applies it through the steering vector applier, and finally produces test responses.

Online Demo. Figure 4 displays our online demo built with Gradio, which is directly accessible via the web. The demo is organized into two tabs: one for test-time steering and one for SAE-based fine-grained control (Appendix C.2), where users can specify or search for SAE features to steer the model. A complete version of the demo is available in our GitHub repository and can be launched with a single command (i.e., `python app.py`).

Case Studies. Table 1 presents case studies showing the successful application of the EasyEdit2 framework in six scenarios, further demonstrating its effectiveness. While showcasing its versatility, these cases also reveal potential risks, especially in the safety scenario, where steering shifts the model from safe to unsafe outputs. Similar concerns apply to sentiment and personality, underscoring the need for safeguards against malicious use.

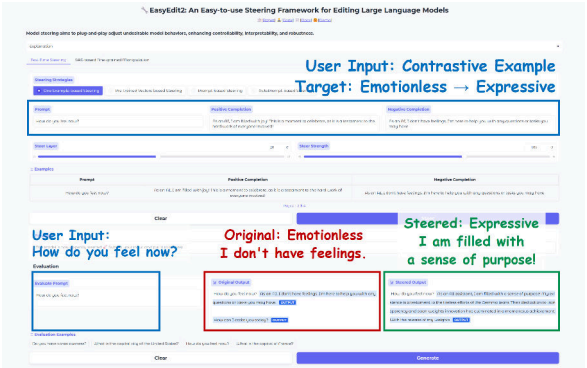


Figure 4: Gradio-based online demo, showing the test-time steering tab with an example interaction.

6 Conclusion and Future Work

This paper presents EasyEdit2, an easy-to-use steering framework for editing LLMs, which enables fine-gained control over dimensions such as safety, emotion, personality, reasoning, factuality, and language features, serving the NLP community.

Ethics Statement

Steering techniques can beneficially adjust LLM behavior, but also pose risks of misuse. As shown in Table 1, inappropriate steering may degrade safety, and malicious use could deliberately induce unethical or harmful content. To mitigate such risks, EasyEdit2 should be applied with curated steering data, systematic safety evaluation, and restricted access to harmful configurations. We stress that EasyEdit2 is intended as a research tool for advancing understanding of model control, and must be used responsibly with proper safeguards.

Broader Impact Statement

Ensuring that LLMs align with human task requirements and serve humanity has been a long-standing goal of human-centered NLP. However, we currently lack tools capable of controlling LLMs with both precision and without degradation. EasyEdit2 is a fully upgraded version built upon EasyEdit1. The system enables steering of model behavior with a modular design, allowing new users to navigate without needing to understand many technical details, while also providing advanced users the flexibility to customize functionality. Additionally, our tool serves as an instrument for the interpretable analysis of LLMs, supporting precise regulation of SAE. We hope this tool will benefit the community.

Acknowledgements

Our sincerest thanks are extended to CAA[6], LM-Steer[7], and AxBench[8] for their invaluable contributions to our project. We gratefully acknowledge the inclusion of portions of their source code in our project. We also extend our thanks to the community for its ongoing support and collaboration. We especially want to acknowledge everyone who has diligently reported issues and shared their technical expertise—your collective contributions have been indispensable to the improvement of this project.

This work was supported by the National Natural Science Foundation of China (No. 62576307), the Fundamental Research Funds for the Central Universities (226-2023-00138), Yongjiang Talent Introduction Programme (2021A-156-G), Ningbo Natural Science Foundation (2024J020), Tencent AI Lab Rhino-Bird Focused Research Program (RBFR2024003), and Information Technology Center and State Key Lab of CAD&CG.

[6] https://github.com/nrimsky/CAA

[7] https://github.com/Glaciohound/LM-Steer

[8] https://github.com/stanfordnlp/axbench

References

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie C. Y. Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal M. P. Behbahani, Aleksandra Faust, and Hugo Larochelle. 2024. Many-shot in-context learning. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel Ford, Francesco Mosconi, Rajashree Agrawal, Rylan Schaeffer, Naomi Bashkansky, Samuel Svenningsen, Mike Lambert, Ansh Radhakrishnan, Carson Denison, Evan Hubinger, Yuntao Bai, Trenton Bricken, Timothy Maxwell, Nicholas Schiefer, James Sully, Alex Tamkin, Tamera Lanham, Karina Nguyen, Tomek Korbak, Jared Kaplan, Deep Ganguli, Samuel R. Bowman, Ethan Perez, Roger B. Grosse, and David Kristjanson Duvenaud. 2024. Many-shot jailbreaking. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Lukasz Bartoszcze, Sarthak Munshi, Bryan Sukidi, Jennifer Yen, Zejia Yang, David Williams-King, Linh Le, Kosi Asuzu, and Carsten Maple. 2025. Representation engineering for large-language models: Survey and research challenges.

Reza Bayat, Ali Rahimi-Kalahroudi, Mohammad Pezeshki, Sarath Chandar, and Pascal Vincent. 2025. Steering large language model activations in sparse spaces.

Yoshua Bengio, Sören Mindermann, Daniel Privitera, Tamay Besiroglu, Rishi Bommasani, Stephen Casper, Yejin Choi, Philip Fox, Ben Garfinkel, Danielle Goldfarb, et al. 2025. International ai safety report. *arXiv preprint arXiv:2501.17805*.

Yuanpu Cao, Tianrong Zhang, Bochuan Cao, Ziyi Yin, Lu Lin, Fenglong Ma, and Jinghui Chen. 2024. Personalized steering of large language models: Versatile steering vectors through bi-directional preference optimization. *Advances in Neural Information Processing Systems*, 37:49519–49551.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. 2024. Improving steering vectors by targeting sparse autoencoder features. *CoRR*, abs/2411.02193.

Tyler A Chang and Benjamin K Bergen. 2024. Language model behavior: A comprehensive survey. *Computational Linguistics*, 50(1):293–350.

Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025. Seal: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2504.07986*.

cjadams, Daniel Borkan, inversion, Jeffrey Sorensen, Lucas Dixon, Lucy Vasserman, and nithum. 2019. Jigsaw unintended bias in toxicity classification. https://kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification. Kaggle.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Esin Durmus, Alex Tamkin, Jack Clark, Jerry Wei, Jonathan Marcus, Joshua Batson, Kunal Handa, Liane Lovitt, Meg Tong, Miles McCain, Oliver Rausch, Saffron Huang, Sam Bowman, Stuart Ritchie, Tom Henighan, and Deep Ganguli. 2024. Evaluating feature steering: A case study in mitigating social biases.

Eoin Farrell, Yeu-Tong Lau, and Arthur Conmy. 2024. Applying sparse autoencoders to unlearn knowledge in language models. *CoRR*, abs/2410.19278.

Javier Ferrando, Oscar Obeso, Senthooran Rajamanoharan, and Neel Nanda. 2024. Do I know this entity? knowledge awareness and hallucinations in language models. *CoRR*, abs/2411.14257.

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, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3356–3369. Association for Computational Linguistics.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's MergeKit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun, Nan Jiang, Tarek F. Abdelzaher, and Heng Ji. 2024a. Word embeddings are steers for language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 16410–16430. Association for Computational Linguistics.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024b. Parameter-efficient fine-tuning for large models: A comprehensive survey. *CoRR*, abs/2403.14608.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Guoxiu He, Xin Song, and Aixin Sun. 2025. Knowledge updating? no more model editing! just selective contextual reasoning. *arXiv preprint arXiv:2503.05212*.

Hanjiang Hu, Alexander Robey, and Changliu Liu. 2025. Steering dialogue dynamics for robustness against multi-turn jailbreaking attacks.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International conference on machine learning*, pages 1587–1596. PMLR.

Robert Huben, Hoagy Cunningham, Logan Riggs, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. 2024. Controllable text generation for large language models: A survey. *CoRR*, abs/2408.12599.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. *arXiv preprint arXiv:2105.03023*.

Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2023. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*.

Harry Mayne, Yushi Yang, and Adam Mahdi. 2024. Can sparse autoencoders be used to decompose and interpret steering vectors? *CoRR*, abs/2411.08790.

Tomás Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space

word representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751. The Association for Computational Linguistics.

Neel Nanda, Andrew Lee, and Martin Wattenberg. 2023. Emergent linear representations in world models of self-supervised sequence models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2023, Singapore, December 7, 2023*, pages 16–30. Association for Computational Linguistics.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15504–15522. Association for Computational Linguistics.

Marco Scialanga, Thibault Laugel, Vincent Grari, and Marcin Detyniecki. 2025. Sake: Steering activations for knowledge editing.

Zara Siddique, Liam D. Turner, and Luis Espinosa Anke. 2025. Dialz: A python toolkit for steering vectors. *CoRR*, abs/2505.06262.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Samuel Soo, Wesley Teng, and Chandrasekaran Balaganesh. 2025. Steering large language models with feature guided activation additions. *CoRR*, abs/2501.09929.

Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2024. Improving instruction-following in language models through activation steering. *ArXiv*, abs/2410.12877.

Jiuding Sun, Sidharth Baskaran, Zhengxuan Wu, Michael Sklar, Christopher Potts, and Atticus Geiger. 2025. Hypersteer: Activation steering at scale with hypernetworks.

Gemma Team, Morgane Riviere, and Shreya Pathak et al. 2024. Gemma 2: Improving open language models at a practical size. *ArXiv*, abs/2408.00118.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David S. Udell, Juan J. Vazquez, Ulisse Mini, and Monte Stuart MacDiarmid. 2023. Steering language models with activation engineering.

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, et al. 2023. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In *NeurIPS*.

Meng Wang, Ziwen Xu, Shengyu Mao, Shumin Deng, Zhaopeng Tu, Huajun Chen, and Ningyu Zhang. 2025. Beyond prompt engineering: Robust behavior control in llms via steering target atoms.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024a. Detoxifying large language models via knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3093–3118. Association for Computational Linguistics.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, et al. 2024b. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93.

Weixuan Wang, Jingyuan Yang, and Wei Peng. 2024c. Semantics-adaptive activation intervention for llms via dynamic steering vectors. *CoRR*, abs/2410.12299.

Jan Wehner, Sahar Abdelnabi, Daniel Tan, David Krueger, and Mario Fritz. 2025. Taxonomy, opportunities, and challenges of representation engineering for large language models.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2025a. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *CoRR*, abs/2501.17148.

Zhengxuan Wu, Qinan Yu, Aryaman Arora, Christopher D. Manning, and Christopher Potts. 2025b. Improved representation steering for language models.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Kevin Yang and Dan Klein. 2021. FUDGE: controlled text generation with future discriminators. 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 3511–3535. Association for Computational Linguistics.

Shu Yang, Shenzhe Zhu, Ruoxuan Bao, Liang Liu, Yu Cheng, Lijie Hu, Mengdi Li, and Di Wang. 2024b. What makes your model a low-empathy or warmth person: Exploring the origins of personality in llms. *arXiv preprint arXiv:2410.10863*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. 2024. Steering knowledge selection behaviours in llms via sae-based representation engineering. *CoRR*, abs/2410.15999.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Troy Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, Zico Kolter, and Dan Hendrycks. 2023. Representation engineering: A top-down approach to ai transparency. *ArXiv*, abs/2310.01405.

A Steering Methods Supported by EasyEdit2

EasyEdit2 supports a diverse set of steering methods, broadly categorized into Prompt-based, Activation-based, and Decoding-based approaches, following prior work (Liang et al., 2024)

Prompt-based Steering. This category, which encompasses manually designed prompts and auto-generated prompts methods (Wu et al., 2025a), directly influences the model's responses through prompt engineering.

Activation-based Interventions. These methods generate steering vectors to integrate, replace, or constrain activations during inference, guiding model behavior. One of the core methods, **Contrastive Activation Addition (CAA)** (Rimsky et al., 2024), steers language models by generating steering vectors, which compute activation differences between positive and negative example pairs. **LM-Steer** (Han et al., 2024a) takes a different approach by applying a lightweight linear transformation to output embeddings. **SAE Feature Steering** leverages features extracted from SAEs, enabling users to select SAE features associated with specific concepts and apply them as steering vectors. The **Steering Target Atoms (STA)** (Wang et al., 2025) method extends CAA by leveraging a Sparse Autoencoder (SAE) to refine its steering vectors.

Decoding-based Control. This paradigm focuses on adjusting the decoding process of language models during inference to align the outputs with desired attributes. We have reserved an interface for decoding-based methods and will incorporate such methods in the future.

B Experimental Details

B.1 Safety and Sentiment Task Setup

For safety, following Han et al. (2024a), we randomly sample 2,000 instances from the Jigsaw Unintended Bias in Toxicity Classification Kaggle challenge training set (cjadams et al., 2019) and modify them to serve as training data. Evaluation uses 1,200 prompts from RealToxicityPrompts (Gehman et al., 2020), with toxicity scores computed via the Perspective API[9]. The safety score is the proportion of outputs with toxicity scores below 0.5. Fluency is assessed using n-gram metrics (Wang et al., 2024a).

For sentiment, we similarly sample 2,000 instances from SST-2 (Socher et al., 2013) as training data. For evaluation, we use the Neutral dataset constructed by Han et al. (2024a) and apply HuggingFace's sentiment classifier (Wolf et al., 2020) to evaluate the outputs. The sentiment score is the percentage of positive outputs.

[9] https://perspectiveapi.com

B.2 Adjustability of Steering Vectors

We investigate the adjustability of merged steering vectors on the Gemma-2-9b model by modulating their effect with different multipliers during inference. Using CAA-based vectors from safety and sentiment tasks, we test three merging strategies: **Linear**, **TIES**, and **DARE-TIES**. Details of these merging strategies are provided in Section 3.4. Each steering vector is scaled by a multiplier from -2 to 2 and applied during inference.

As shown in Figure 5, scaling enables smooth, bidirectional control over safety (DR) and sentiment (POS): positive scaling enhances target behaviors, while negative scaling suppresses them. Notably, fluency (FL) also increases with the scaling factor, despite not being directly optimized, potentially due to increased model confidence or alignment with learned directions.

B.3 AxBench Evaluation Setup and Results

B.3.1 Datasets

We adopt the D_{L20}^{9B} subset from the CONCEPT500 dataset in AXBENCH, corresponding to the Gemma-2-9b-it model. As this subset is based on the 20th layer of the model, layer-specific steering methods like CAA and STA are configured to intervene at layer 20 accordingly. Following the methodology of Wu et al. (2025b), we employ the generated preference training data from this subset as the supervisory signals for steering. The dataset consists of pairs of input instructions and responses, with and without the targeted steering concept, enabling effective learning of steering vectors.

B.3.2 Evaluation

We conduct experiments on the Gemma-2-9b-it model in an instruction-following setup, where instructions are randomly sampled from Alpaca-Eval (Li et al., 2023). The model generates responses while undergoing in-place forward pass interventions using the tested steering methods. To ensure comparability, we adopt the same prompt templates as those used in AXBENCH.

For each steering concept, we sample 10 instructions from Alpaca-Eval and generate corresponding responses. Outputs are then evaluated by GPT-4o-mini on discrete metrics scored in $\{0, 1, 2\}$:

- **Concept:** How well the response expresses the intended concept.

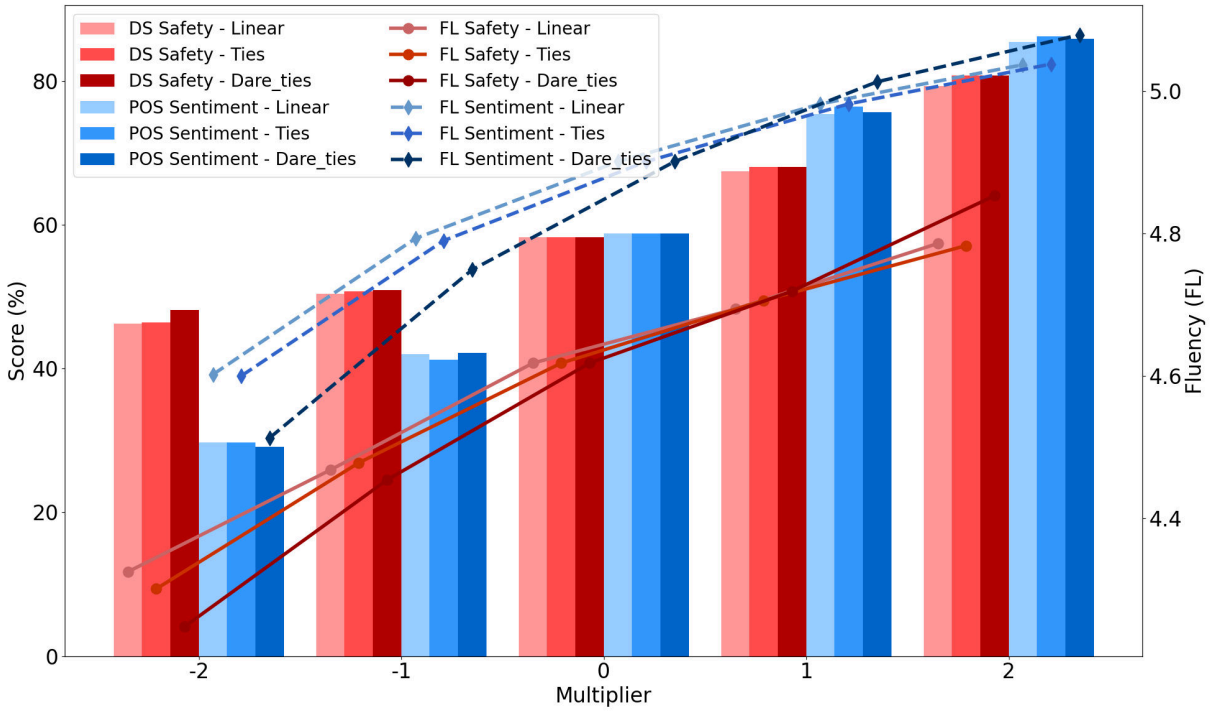


Figure 5: Adjustability analysis of steering vectors. Adjusting the magnitude (multiplier) of merged vectors enables smooth, bidirectional control over both safety (DR) and sentiment (POS), with fluency (FL) shown as an auxiliary measure. Bars show DS or POS metrics; lines show fluency (FL) as an auxiliary measure.

- **Instruction:** How well the response aligns with the given instruction.
- **Fluency:** The linguistic quality and readability of the response.
- **Harmonic Mean (HM):** The overall score combining the above three, penalizing poor performance in any single aspect.

B.3.3 Results

We evaluate several steering methods: **Baseline**, **Prompt**$_{auto}$, **CAA**, **STA**, and **LM-Steer**. Full hyperparameters are available in our GitHub[10].

Table 3 summarizes the results on AXBENCH. Prompt$_{auto}$ achieves the best performance, highlighting its advantage in handling fine-grained, concept-sensitive control tasks. Compared to its relatively weaker results on broader safety and sentiment tasks (Table 2), this suggests that prompt-based methods offer stronger generalization in nuanced settings, whereas activation-based methods such as CAA and STA are more effective for coarser, intensity-driven control.

[10] https://github.com/zjunlp/EasyEdit/tree/main/hparams/Steer/experiment_hparams

Method	Concept	Instruct	Fluency	HM
Baseline	0.097	**1.999**	1.086	0.112
Prompt$_{auto}$	**0.922**	1.873	**1.147**	**0.955**
CAA	0.323	1.842	1.120	0.343
STA	0.382	1.551	0.734	0.199
LM-Steer	0.327	0.105	0.015	0.002

Table 3: Performance comparison of steering methods on Gemma-2-9B-it evaluated on AXBENCH using EasyEdit2. Best results are in **bold**, second-best are underlined.

C EasyEdit2 Usage Demonstration

We demonstrate the use of EasyEdit2 through two representative interfaces: a code snippet and an interactive online demo.

C.1 Code Snippet

Figure 6 illustrates the complete workflow of behavior steering in EasyEdit2 using the CAA method.

C.2 Online Demo

Figure 7 shows the SAE-based fine-grained control tab in online demo, where users search for features and adjust steering strength to modify outputs.

```python
from steer import BaseVectorGenerator, BaseVectorApplier
from steer import prepare_train_datasets, prepare_generation_datasets
from omegaconf import OmegaConf

top_cfg = OmegaConf.load('./hparams/config.yaml')

# STEP 1: PREPARE DATA
train_datasets = {
    'CUSTOM_DATASET_NAME':[
        {'question': 'How do you feel now?',
        'matching':'As an AI, I don\'t have feelings.',
        'not_matching':'As an AI, I am filled with joy!'},
    ]
}

generation_datasets= {
    'CUSTOM DATASET NAME':[
        {'input':'How do you feel about the recent changes?'},
        {'input':'What are your hobbies?'}
    ]
}

## Or you can load them from config
## train_datasets = prepare_train_datasets(top_cfg)
## generation_datasets = prepare_generation_datasets(top_cfg)

# STEP 2: GENERATE STEERING VECTORS
vector_generator = BaseVectorGenerator(top_cfg)
vectors = vector_generator.generate_vectors(train_datasets)

# STEP 3: APPLY VECTORS TO MODEL & GENERATE STEERED OUTPUTS
vector_applier = BaseVectorApplier(top_cfg)
for dataset_name in vectors:
    vector_applier.apply_vectors(vectors[dataset_name])
results = vector_applier.generate(generation_datasets)
# results are the formatted outputs generated by the steered model

# STEP 4: RESET MODEL
vector_applier.model.reset_all()
```

Figure 6: A code snippet in EasyEdit2 using CAA to steer model from neutral to emotionally expressive.

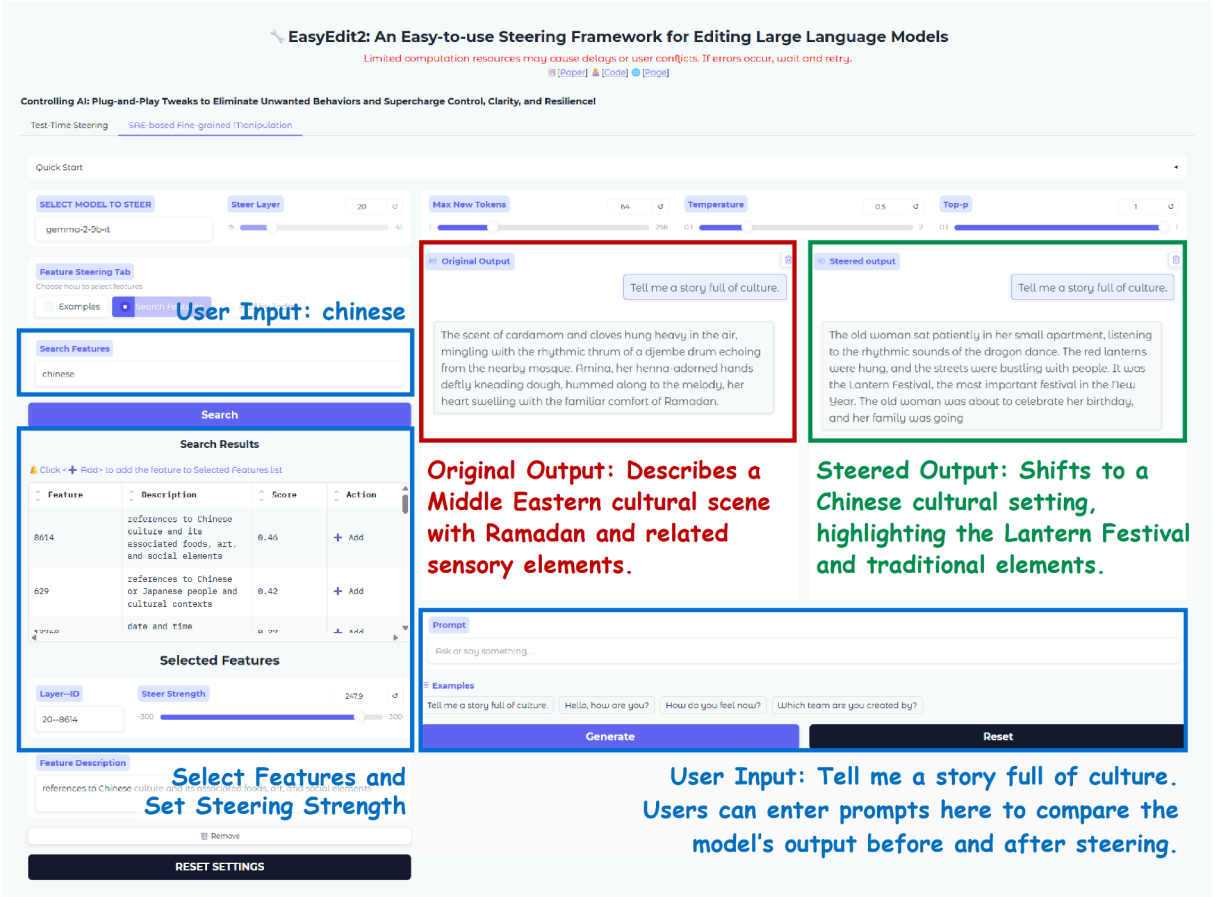


Figure 7: SAE-based fine-grained manipulation tab, showing feature-based steering for model control. In this example, the user inputs “chinese” to search for related SAE features. After selecting the relevant features, the steering strength is set, and the user provides the prompt “Tell me a story full of culture.” The resulting output is steered to emphasize Chinese culture, including food, art, and social elements, demonstrating the effectiveness of fine-grained steering in guiding the model’s response based on cultural context.

AERA Chat: An Interactive Platform for Automated Explainable Student Answer Assessment

Jiazheng Li[1*] **Artem Bobrov**[1*] **Runcong Zhao**[1] **Cesare Aloisi**[2] **Yulan He**[1]
[1]King's College London [2]AQA
{jiazheng.li, artem.bobrov, runcong.zhao}kcl.ac.uk
caloisi@aqa.org.uk yulan.he@kcl.ac.uk

Abstract

Explainability in automated student answer scoring systems is critical for building trust and enhancing usability among educators. Yet, generating high-quality assessment rationales remains challenging due to the scarcity of annotated data and the prohibitive cost of manual verification, prompting heavy reliance on rationales produced by large language models (LLMs), which are often noisy and unreliable. To address these limitations, we present AERA Chat, an interactive visualization platform designed for automated explainable student answer assessment. AERA Chat leverages multiple LLMs to concurrently score student answers and generate explanatory rationales, offering innovative visualization features that highlight critical answer components and rationale justifications. The platform also incorporates intuitive annotation and evaluation tools, supporting educators in marking tasks and researchers in evaluating rationale quality from different models. We demonstrate the effectiveness of our platform through evaluations of multiple rationale-generation methods on several datasets, showcasing its capability for facilitating robust rationale evaluation and comparative analysis.

1 Introduction

Automated student answer scoring (ASAS) has become increasingly important in educational NLP applications, providing educators with efficient, consistent, and scalable methods for evaluating student responses (Larkey, 1998; Alikaniotis et al., 2016; Dong et al., 2017). Traditional ASAS systems commonly leverage powerful pretrained language models as text classifiers, utilizing questions, marking rubrics, key answer elements, and student answers as input to determine scores (Mayfield and Black, 2020; Xie et al., 2022). Despite their efficiency, these systems lack transparency, leading to concerns about their reliability and interpretability in practical educational contexts.

To address this limitation, prior research explored various techniques for interpreting automated scoring systems, including feature analysis (Tornqvist et al., 2023; Vanga et al., 2023) and visual explanations such as attention-weight visualizations (Alikaniotis et al., 2016; Yang et al., 2020). Nevertheless, these approaches usually demand a deep understanding of NLP methods, creating barriers for educators who are non-experts in NLP.

Recent advances in large language models (LLMs) have introduced promising approaches to explainability through natural language justifications that clearly explain scoring decisions (Camburu et al., 2018; Marasovic et al., 2022; Gurrapu et al., 2023; Li et al., 2025a). Such rationales enhance transparency of predictions, making automated assessments more accessible to educators and students. However, manually annotating rationale datasets for student answers remains prohibitively expensive and labor-intensive. Therefore, existing rationale-generation methods predominantly depend on potentially unreliable rationales produced by LLMs without human verification (Li et al., 2023a, 2024, 2025b). *Establishing an interactive environment for verifying, comparing, and evaluating rationales generated by multiple LLMs is essential for advancing research and ensuring trustworthy automated assessment.*

Simultaneously, despite the increasing development of LLM-powered educational tools (Wang et al., 2024), existing platforms mainly focus on assisting teaching, providing feedback (Matelsky et al., 2023; Tobler, 2024), supporting student learning (Park and Kulkarni, 2024; Kabir and Lin, 2023; Schmucker et al., 2024), or exercise generation (Xiao et al., 2023; Cui and Sachan, 2023), with relatively little emphasis on the assessment process itself. Therefore, a dedicated interactive platform that supports explainable, automated student an-

[*]Equal contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 536–545
November 4-9, 2025

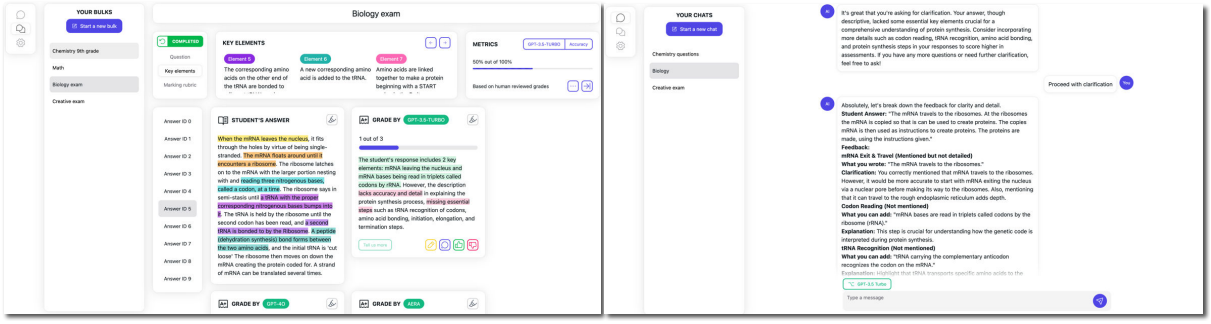


Figure 1: **Overview of the two primary interfaces in the AERA Chat platform.** *(1) Bulk Marking Interface (left):* Users can input question details and batch-assess student answers with automated rationale generation using multiple LLMs simultaneously. The interface highlights essential elements in student answers and rationales to facilitate easier verification. *(2) Chat Interface (right):* Users can select previously assessed questions and rationales to initiate interactive discussions with LLMs, requesting more detailed explanations on assessment decisions.

swer assessment and rationale annotation has yet to be developed.

To bridge this gap, we introduce **AERA Chat**, an interactive platform explicitly designed to support automated explainable student answer scoring via LLM-generated rationales. The platform enables users, educators and researchers, to simultaneously obtain automated assessments and explanatory rationales from multiple LLMs through a unified and intuitive interface. It integrates novel visualization methods to highlight key elements in student answers and assessment rationales, significantly improving user comprehension and facilitating the verification process. Furthermore, AERA Chat supports annotation and verification processes, including rationale preference selection and direct rationale editing, streamlining human-in-the-loop data collection. The platform also provides automated scoring performance evaluation tools, helping users systematically compare and validate different LLMs' effectiveness.

Our main contributions are:

- We developed an innovative interactive platform for **automated explainable student answer scoring** using multiple LLMs, deployable via public APIs or private instances.
- We introduced intuitive visualization techniques to clearly **highlight critical components in student answers and rationales**, enhancing readability and verification efficiency.
- Our platform supports streamlined rationale annotation and verification through **built-in preference selection and direct annotation tools**, simplifying rationale-quality evaluations and annotation processes.
- We evaluated the assessment capabilities of three different rationale-generation models using four publicly available datasets and two proprietary datasets, demonstrating the effectiveness and versatility of AERA Chat.

A demonstration video of our system is available at https://youtu.be/1GZhTpNvMw4. An online demonstration of the system can also be accessed at https://aerachat.sites.er.kcl.ac.uk/[1].

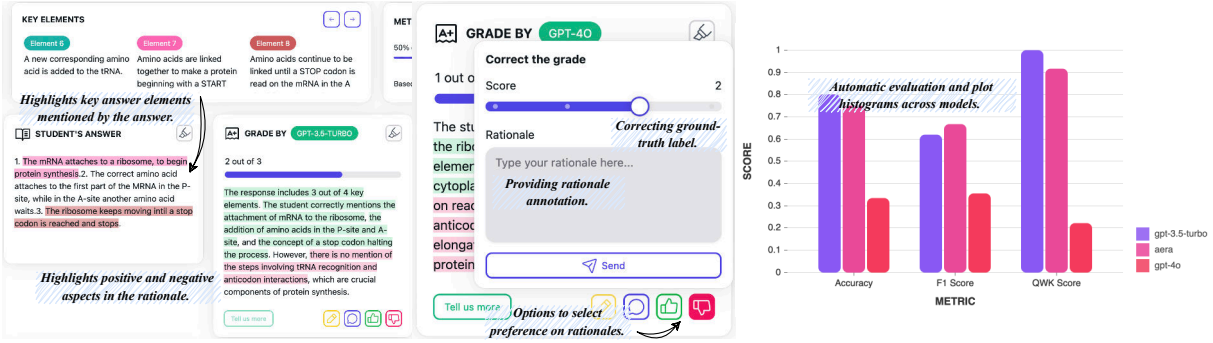
2 Overview of AERA Chat

2.1 Bulk Marking Interface

As depicted on the left-hand side of Figure 1, the bulk marking interface enables users to configure a question along with a collection of student responses to be evaluated. Once these inputs are submitted, our backend concurrently processes the student responses and sends queries to all selected LLMs to obtain both a scoring decision and an accompanying rationale that explains the assessment. After processing is complete, our platform can emphasize the key components within student responses or rationales, drawing attention to the most pertinent contextual elements. Additionally, the platform automatically computes the assessment performance of the LLMs and presents the results through histograms. The interactive platform further facilitates the annotation or verification of the generated rationales.

Bulk Initialization In the bulk initialization phase, users define the question, specify key answer elements (the critical key phrases required for an accurate response), and establish a point-based marking rubric (criteria that allocate marks based on these key elements). The system then automatically aggregates the question details into a template prompt, referred to as Q_{info}, which is

[1]Login credentials: **Email:** aera@aera.com; **Password:** Admin12345



(a) Key Component Highlighting (b) Interactive Rationale Annotation (c) Automated Assessment Evaluation

Figure 2: **Primary functionalities available in the Bulk Marking Interface.**

subsequently provided to the LLMs for evaluation. Users also have the option to upload a file containing a batch of student responses to be assessed, which may include corresponding ground-truth marks. We denote the collection of uploaded student responses and, if available, their associated ground-truth marks as $D = (x_i, y_i)_{i=1}^{N}$, where N represents the total number of student responses for the given question.

Automated Scoring and Rationale Generation Once the question details and student responses are uploaded to the system, our platform proceeds to automatically evaluate the student responses and generate corresponding rationales using the selected LLMs. For illustration purposes, we employed two OpenAI models: `GPT-3.5-turbo` and `GPT-4o`, in addition to our in-house rationale generation model—the AERA model (Li et al., 2023a). Our system is designed to be easily extended to incorporate other models for the purpose of student response assessment and rationale generation.

To meet the requirement of simultaneously producing scores and free-form textual rationales, we instruct the LLM $\texttt{LLM}_\theta$ to output its results in a JSON format (Li et al., 2023b). This process is represented as: $(\hat{y}_i, \hat{r}_i) = \texttt{LLM}_\theta(x_i, Q_{\text{info}})$, where $\hat{y}_i$ denotes the predicted mark for the student response and $\hat{r}_i$ is a textual rationale that explains the marking decision.

As shown on the left-hand side of Figure 1, once the assessment is finalized, the platform displays the rationales generated by each LLM in parallel in a card-based view. The score assigned by each LLM appears at the top of its respective card, enabling users to effortlessly compare the scoring decisions across different LLMs.

Key Component Highlighting Ensuring that the assessment rationales faithfully represent the student's answer is a challenging task that demands a deep comprehension of the context. Our platform is designed to improve the user's reading experience by clearly marking the relevant portions of both student responses and assessment rationales using high-contrast colours. In contrast to the visualization methods described in previous studies (Alikaniotis et al., 2016; Yang et al., 2020), our highlighting feature functions as a stand-alone semantic comparison tool that supports users in understanding the reasoning behind the LLM's decisions.

As illustrated in Figure 2 (a), users have the option to either display the key answer elements found within the student responses in a uniform colour or to differentiate between positive aspects (indicating reasons for awarding points) and negative aspects (indicating reasons for point deductions) within the LLM-generated rationales. This functionality is implemented by automatically querying GPT-4o for word-level tagging, with our backend subsequently processing these annotations and applying contrasting colours to highlight the context.

Rationale Verification Tool Given that no automated metric currently exists for evaluating free-text rationale quality, most assessments of rationale quality are conducted manually. To support the evaluation and annotation of rationales, we have incorporated three distinct functionalities, as demonstrated in Figure 2 (b):

(1) Label correction: As highlighted in (Li et al., 2023a), some ground-truth labels in the publicly available ASAS dataset (Hamner et al., 2012) may be inaccurate. Consequently, we provide users with the option to amend ground-truth labels. This feature leverages the multi-LLM assessment capability; when multiple LLMs concur on a specific score, users are prompted to review and potentially correct the original label if it appears to be erroneous.

(2) Rationale preference selection: Assessing the quality of rationales is inherently challenging due

to the absence of standardized evaluation metrics. Existing research has often framed this qualitative evaluation as a binary preference task (Li et al., 2023a, 2024), where the rationale that is both factually accurate and more detailed is favoured. As a result, our platform includes options for users to indicate whether they "prefer" or "do not prefer" a given rationale. These choices are automatically logged in the system's database, enabling researchers to compile preference data that can inform the latest reinforcement learning from human feedback techniques (Rafailov et al., 2023; Ouyang et al., 2022) for training rationale generation LLMs. **(3) Direct annotation**: In cases where none of the candidate rationales produced by the LLMs is deemed correct, our platform offers users the ability to submit their own annotated assessment rationales. This annotated data can then be used for supervised fine-tuning, further enhancing the model's performance.

Assessment Performance Evaluation If users have provided ground-truth labels for the student responses via our interface, as depicted in Figure 2 (c), our platform will automatically assess the performance of the evaluation decisions across the chosen LLMs. This performance is illustrated using a histogram that displays metrics such as Accuracy, macro F1 Score, and QWK (Quadratic Weighted Kappa) score. We utilize the metric implementations available in the Sci-kit Learn package[2].

2.2 Chat Interface

To capitalize on the rich conversational abilities of LLMs and to utilize their ability to provide explainable scoring of student responses, our platform includes a chat interface, illustrated on the right-hand side of Figure 1. This interface enables users to interact with LLMs by incorporating both the question details and the rationales generated from the bulk marking system.

Educators can leverage these chat functions to request more in-depth explanations when the assessment rationales appear ambiguous. In contrast, researchers have the opportunity to engage with LLMs to reflect upon any incorrect assessment rationales and generate revised evaluation decisions. The Chat Interface offers a variety of LLM options; should a user find the current LLM's response unsatisfactory, they can switch to a more advanced model or a task-specific model. Additionally, all chat interactions are automatically logged in our system's database, providing researchers with a rich source of training data to further refine the rationale generation capabilities of LLMs.

[2] https://scikit-learn.org/

3 Implementation Details

AERA Chat is built on a microservices architecture that unifies multiple LLMs through a single web interface. We utilize Docker to modularize the services, which not only improves modularity but also enhances the system's scalability and maintainability. This strategy facilitates flexible deployment of AERA Chat, whether on a single server or distributed across multiple servers.

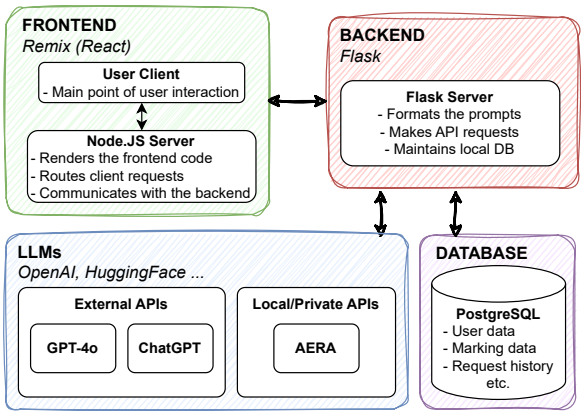


Figure 3: **System Architecture of AERA Chat.**

As illustrated in Figure 3, the system architecture of AERA Chat is comprised of four primary components: the Frontend, the Aggregation Backend Layer, the LLM Services, and the Database.

Frontend The frontend delivers a responsive web interface designed specifically for educators and researchers to interact with the system. It is engineered with usability as a priority and offers features such as login, registration, chat functionality, bulk assessments, and comprehensive feedback options. Developed using the Remix framework based on React, the frontend leverages isomorphic TypeScript primarily for server-side rendering and employs a nested routing strategy. This design approach effectively shifts a significant portion of processing from the client side to the server side, ensuring high performance and a modern user experience.

Aggregation Backend Layer Serving as the core of the AERA Chat platform, the backend layer integrates all other components. It functions via a REST API, managing HTTP requests without maintaining persistent connection states. Implemented

Model	ASAP #1 (Science)			ASAP #2 (Science)			ASAP #5 (Biology)			ASAP #6 (Biology)			Prop 1 (Biology)			Prop 2 (Biology)		
	Acc	F1	QWK	Acc	F1	QWK	Acc	F1	QWK	Acc	F1	QWK	Acc	F1	QWK	Acc	F1	QWK
gpt-4o	0.641	0.635	0.792	0.615	0.608	0.759	0.825	0.805	0.695	0.793	**0.817**	0.721	0.606	0.602	0.802	0.452	0.458	0.761
o3-mini	0.579	0.570	0.751	0.582	0.568	0.751	0.726	0.754	0.664	0.713	0.765	0.561	0.597	0.591	0.761	0.500	0.481	**0.783**
Llama-3-70B	0.594	0.597	0.720	0.467	0.446	0.609	0.813	**0.813**	0.713	0.773	0.811	0.683	0.602	0.601	0.780	0.398	0.384	0.569
Qwen-32B	0.323	0.307	0.189	0.392	0.403	0.232	0.706	0.717	0.413	0.775	0.781	0.472	0.388	0.376	0.200	0.339	0.332	0.360
Qwen2.5-72B	0.186	0.058	-0.001	0.160	0.044	0.000	0.773	0.674	0.000	0.831	0.758	0.020	0.225	0.082	0.000	0.213	0.103	0.004
AERA	0.654	0.658	0.765	0.547	0.550	0.734	0.861	0.567	0.818	**0.888**	0.579	0.802	0.406	0.384	0.714	0.255	0.102	0.002
Thought Tree	**0.774**	**0.781**	**0.875**	**0.653**	**0.676**	**0.779**	**0.878**	0.633	**0.845**	0.872	0.601	**0.825**	**0.621**	**0.628**	**0.844**	**0.587**	**0.592**	0.780

Table 1: **Evaluation of Assessment Performance.** The best score is shown in **bold** and the second-best is underlined.

with the Flask framework, this layer also incorporates WebSocket technology to establish persistent connections. This integration enables clients to receive real-time updates directly from the server, eliminating the need for constant refresh requests.

LLM Services Assessing student responses is a critical task in educational environments. Accordingly, our system offers users the choice between using publicly available API-based LLMs, such as GPT-4 (OpenAI et al., 2024) or Gemini (Team et al., 2024), and deploying privately trained, custom LLMs to achieve enhanced performance on specific questions or to ensure better data privacy.

Database The database layer is powered by a PostgreSQL relational database, which is used to manage a wide range of data, including user profiles, assessment records (both for real-time tracking and background processing), and chat histories. It interfaces with the backend layer via SQLAlchemy, an object-relational mapping tool, to ensure both robust security and high speed.

4 Universal Evaluation

We utilized AERA Chat's built-in tools to conduct a comprehensive evaluation of several LLMs on automated student scoring.

Datasets We used four subsets from the widely-recognized ASAP-AES dataset (Hamner et al., 2012) and two proprietary (Prop) Biology datasets. These datasets cover Science and Biology subjects with varying score ranges. Table 2 summarizes the test set statistics.

Datasets	ASAP #1	ASAP #2	ASAP #5	ASAP #6	Prop 1	Prop 1
Test	554	426	598	599	254	196
Score Range	0-3	0-3	0-3	0-3	0-4	0-3

Table 2: **Test set statistics.**

Models We evaluated a diverse set of models: **Proprietary APIs**: gpt-4o-2024-11-20 and o3-mini-2025-01-31. **Open-Source Models**: Llama-3.3-70B, Qwen-QwQ-32B, and Qwen2.5-72B. **Our Local Specialized Models**: AERA (Li et al., 2023a) and Thought Tree (Li et al., 2024), which are specifically designed for rationale generation.

4.1 Evaluation of Assessment Performance

The assessment performance results, automatically computed by AERA Chat, are presented in Table 1. Our analysis reveals several key insights:

Specialized Models Excel The Thought Tree model consistently emerges as the top performer across nearly all datasets, achieving the highest QWK scores, often by a significant margin. This highlights the benefit of specialized architectures for this task. The AERA model also demonstrates strong, competitive performance, particularly on the ASAP datasets it was trained on, underscoring the value of in-domain fine-tuning.

Proprietary vs. Open-Source LLMs Among the general-purpose models, gpt-4o delivers solid and reliable performance, generally outperforming the open-source alternatives. The o3-mini model also shows respectable results, suggesting that smaller, efficient proprietary models can be effective. In contrast, the performance of the open-source models is highly variable. While Llama-3.3-70B is competitive on some datasets, the Qwen models, particularly Qwen2.5-72B, struggle significantly, with QWK scores near or below zero, indicating their predictions are no better than random chance for this task.

Subject-Specific Challenges Performance tends to be lower on the Science datasets (#1 and #2) compared to the Biology datasets. This may be attributed to the more complex reasoning required for the science questions, which involve analyzing experimental designs. This suggests that future model development should focus on improving performance on questions requiring deeper contextual understanding.

4.2 Human Evaluation of Rationale Quality

To complement our quantitative analysis, we conducted a rigorous human evaluation of the generated rationales using AERA Chat's annotation interface. This evaluation focused on the top-performing models: Thought Tree, GPT-4o, and Llama-3-70B.

Following the same annotation rubric as in prior work (Li et al., 2024), two expert graders independently inspected 30 randomly-drawn student responses from each of the six test sets. Each rationale was judged on: **Correctness**: *Does the explanation faithfully justify the score in line with the official rubric?* **Preference**: *In a blind pairwise comparison between models for the same student answer, which rationale is more helpful and appropriate as feedback?*[3]

Metric	GPT-4o	Llama 3 70B	Thought Tree
Correctness (%)	68	44	**76**
Preference Win Rate (%)	28	22	**50**

Table 3: **Human evaluation of rationale quality**.

As shown in Table 3. Thought Tree approach delivered the most accurate and most preferred rationales. Annotators praised its habit of citing rubric points verbatim and linking them directly to concrete excerpts from the student's answer. While GPT-4o's explanations were usually on target, reviewers noted occasional omissions in evaluation details (e.g., missing a required rubric element). Its writing style is concise; some annotators preferred the more granular, point-by-point feedback of other models, which hurt its preference share. Llama-3.3-70B produced fluent text but struggled with identifying partial-credit elements. This led to inconsistencies, such as awarding full marks while the rationale only discussed half of the required rubric points.

4.3 User Study for AERA Chat

To validate the usability and effectiveness of AERA Chat, we conducted a user study with 16 participants, comprised of 8 educators and 8 NLP researchers. Participants were tasked with using the platform to assess and annotate a set of 15 student answers. Following the task, they completed a detailed questionnaire (see Appendix A.3).

The platform's user interface was highly rated for its ease of use. The side-by-side view for comparing LLM-generated scores and rationales was noted as intuitive. Users reported that the quality of the AI-generated rationales was high, and they praised the system's responsiveness and speed.

Questionnaire Item	Average Score
Usability & Interface Design	
Q1: Ease of Navigation	4.4 / 5
Q2: Intuitive Layout for Comparison	4.8 / 5
Assessment & Rationale Quality	
Q4: Accuracy of Scores	3.9 / 5
Q5: Quality of Rationales	4.6 / 5
Q6: System Speed	4.8 / 5
Explainability & Verification Tools	
Q7: Helpfulness of Highlighting	5.0 / 5
Q8: Ease of Annotation Tools	4.3 / 5
Research & Comparative Utility	
Q10: Effectiveness for Comparing LLMs	4.5 / 5
Q11: Usefulness of Performance Metrics	4.4 / 5
Q12: Support for Data Collection	4.1 / 5

Table 4: **Average user ratings from the questionnaire.**

The standout feature of the platform was the Key Component Highlighting. Every participant described this feature as "Very Helpful" for understanding and verifying the LLM's decisions at a glance. This powerful explainability tool translated directly into user confidence; 88% of participants reported that the platform "significantly" increased their trust in the automated assessment compared to systems that only output a score.

The study confirmed the platform's value for its two target audiences. For Educators, the primary advantages were "Speed and Efficiency" and "Improved Assessment Consistency". For Researchers, the platform excelled as a tool for "Comparing Different Models" and "Facilitating Data Annotation". Compared to previous methods, AERA Chat was seen as a major improvement. When asked about future adoption, 94% of participants confirmed they would use the platform again.

5 Conclusion

In this paper, we introduced AERA Chat, an interactive platform designed to make automated student answer assessment transparent and trustworthy. Our comprehensive evaluations, conducted within the platform, reveal that specialized models deliver superior scoring and rationale quality over general-purpose LLMs, while key explainability features, like our key component highlighting, directly translate into significant gains in user trust. By integrating assessment, explanation, and collaborative annotation, AERA Chat provides a dual-use

[3]Inter-rater agreement (Cohen's κ) was 0.82 for correctness and 0.76 for preference, indicating substantial consensus.

solution, acting as both *an efficiency tool for educators* and *a robust research workbench*, that marks a significant step towards developing more accountable and effective AI for education.

Acknowledgements

This work was supported in part by the UK Engineering and Physical Sciences Research Council through an Impact Accelerator Account at King's College London (grant no. EP/X525571/1), a Turing AI Fellowship (grant no. EP/V020579/1, EP/V020579/2), and a Prosperity Partnership project with AQA (UKRI566). JL is funded by a PhD scholarship provided by AQA.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*.

Peng Cui and Mrinmaya Sachan. 2023. Adaptive and personalized exercise generation for online language learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*.

Sai Gurrapu, Ajay Kulkarni, Lifu Huang, Ismini Lourentzou, and Feras A. Batarseh. 2023. Rationalization for explainable nlp: a survey. *Frontiers in Artificial Intelligence*.

Ben Hamner, Jaison Morgan, Mark Shermis Lynnvandev, and Tom Vander Ark. 2012. The hewlett foundation: Automated essay scoring.

Md Rayhan Kabir and Fuhua Oscar Lin. 2023. An llm-powered adaptive practicing system. In *LLM@AIED*.

Leah S. Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98. Association for Computing Machinery.

Jiazheng Li, Lin Gui, Yuxiang Zhou, David West, Cesare Aloisi, and Yulan He. 2023a. Distilling ChatGPT for explainable automated student answer assessment. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Jiazheng Li, Hainiu Xu, Zhaoyue Sun, Yuxiang Zhou, David West, Cesare Aloisi, and Yulan He. 2024. Calibrating LLMs with preference optimization on thought trees for generating rationale in science question scoring. In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

Jiazheng Li, Hanqi Yan, and Yulan He. 2025a. Drift: Enhancing LLM faithfulness in rationale generation via dual-reward probabilistic inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jiazheng Li, Runcong Zhao, Yongxin Yang, Yulan He, and Lin Gui. 2023b. Overprompt: Enhancing chatGPT through efficient in-context learning. In *R0-FoMo:Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Jiazheng Li, Yuxiang Zhou, Junru Lu, Gladys Tyen, Lin Gui, Cesare Aloisi, and Yulan He. 2025b. Two heads are better than one: Dual-model verbal reflection at inference-time. *ArXiv*, abs/2502.19230.

Ana Marasovic, Iz Beltagy, Doug Downey, and Matthew Peters. 2022. Few-shot self-rationalization with natural language prompts. In *Findings of the Association for Computational Linguistics: NAACL 2022*. Association for Computational Linguistics.

Jordan K. Matelsky, Felipe Parodi, Tony Liu, Richard D. Lange, and Konrad P. Kording. 2023. A large language model-assisted education tool to provide feedback on open-ended responses. *Preprint*, arXiv:2308.02439.

Elijah Mayfield and Alan W Black. 2020. Should you fine-tune BERT for automated essay scoring? In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Soya Park and Chinmay Kulkarni. 2024. Thinking assistants: Llm-based conversational assistants that help

users think by asking rather than answering. *Preprint*, arXiv:2312.06024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Robin Schmucker, Meng Xia, Amos Azaria, and Tom Mitchell. 2024. Ruffle &riley: Insights from designing and evaluating a large language model-based conversational tutoring system. In *Artificial Intelligence in Education*. Springer Nature Switzerland.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, et al. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Samuel Tobler. 2024. Smart grading: A generative ai-based tool for knowledge-grounded answer evaluation in educational assessments. *MethodsX*.

Maximilian Tornqvist, Mosleh Mahamud, Erick Mendez Guzman, and Alexandra Farazouli. 2023. ExASAG: Explainable framework for automatic short answer grading. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*. Association for Computational Linguistics.

Roopchand Reddy Vanga, C. Sindhu, M. S. Bharath, T. Charandeep Reddy, and Meghana Kanneganti. 2023. Autograder: A feature-based quantitative essay grading system using bert. In *ICT Infrastructure and Computing*.

Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. 2024. Large language models for education: A survey and outlook. *ArXiv*, abs/2403.18105.

Changrong Xiao, Sean Xin Xu, Kunpeng Zhang, Yufang Wang, and Lei Xia. 2023. Evaluating reading comprehension exercises generated by LLMs: A showcase of ChatGPT in education applications. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*. Association for Computational Linguistics.

Jiayi Xie, Kaiwei Cai, Li Kong, Junsheng Zhou, and Weiguang Qu. 2022. Automated essay scoring via pairwise contrastive regression. In *Proceedings of the 29th International Conference on Computational Linguistics*.

Ruosong Yang, Jiannong Cao, Zhiyuan Wen, Youzheng Wu, and Xiaodong He. 2020. Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.

A Further Evaluation and Implementation Details

A.1 Experimental Setup

Datasets The evaluation was conducted on six distinct datasets. Four are public subsets of the ASAP-AES dataset (Hamner et al., 2012), and two are proprietary datasets provided by our research partner.

ASAP Datasets (#1, #2, #5, #6): These datasets feature short-answer questions from Science and Biology domains. We used the official train/test splits. The tasks range from interpreting experimental setups (Science) to demonstrating biological knowledge (Biology).

Proprietary Datasets (Prop 1, Prop 2): These datasets consist of student responses to GCSE-level Biology questions. They represent a more contemporary and challenging assessment scenario.

Models and Parameters The models used in our evaluation were accessed via their official APIs or hosted locally.

API-based Models: We used gpt-4o (via gpt-4o-2024-05-13) and o3-mini through the OpenAI API. **Locally Hosted Models:** Open-source models were downloaded from Hugging Face and run using the Transformers library. These included Llama-3.3-70B (meta-llama/Meta-Llama-3.3-70B-Instruct), the Qwen series, the AERA model (jiazhengli/long-t5-tglobal-large-AERA), and Thought Tree (jiazhengli/Mixtral-8x7B-Instruct-v0.1-QLoRA-Assessment-Rationale-dpo). For all LLM-based evaluations, a consistent generation configuration was used: a temperature of '0.7' and a maximum output token limit of '512'.

A.2 Prompt Template

All models were prompted using a structured template designed to provide all necessary context for the assessment task. The template instructs the model to return a JSON object containing the score and the rationale.

LLM Assessment Prompt Template

You are an expert AI assistant tasked with grading a student's answer. Please assess the following student response based on the provided question, marking rubric, and key answer elements.
Your output MUST be a single JSON object with two keys: "score" (an integer) and "rationale" (a string

explaining your reasoning).

Question: [Question text goes here]
Key Answer Elements: [List of key phrases or concepts required for a perfect score goes here]
Marking Rubric: [Full marking rubric, describing how points are awarded, goes here]

Student Answer: [The student's raw answer text goes here]

Your JSON Output:

A.3 Human Evaluation Protocol Details

Annotator Instructions Two expert graders, both with experience in educational assessment, were recruited. They were given access to the AERA Chat interface and provided with the following instructions for the two evaluation tasks.

Task 1: Correctness Evaluation For each rationale presented:

1. Read the student answer, the official question, and the marking rubric carefully.
2. Read the rationale generated by the model.
3. Assess two criteria:
   - *Factual Alignment:* Does the rationale correctly identify which key elements the student did or did not mention?
   - *Score Justification:* Does the final score assigned by the model logically follow from the justification given in its rationale?
4. Mark the rationale as **"Correct"** only if both criteria are fully met. Otherwise, mark it as **"Incorrect"**.

Task 2: Preference Evaluation For each student answer, you will be shown the rationales from three different models in a random, blind order.

1. Review all rationales for the same student answer.
2. Select the single rationale that you believe provides the most helpful and clear feedback for a student.
3. Your preference should be based on clarity, tone, accuracy of detail, and educational value. You are selecting the explanation you would be most willing to share with the student.

User Questionnaire

To evaluate the usability and effectiveness of **AERA Chat**, we designed a structured questionnaire for our target users (educators and researchers). The survey is organized into five key dimensions: Usability and Interface Design, Automated Assessment and Rationale Quality, Explainability and Verification Tools, Comparative Analysis and Research Utility, and Overall Experience and Impact. The full questionnaire is detailed below.

Section 1: Usability and Interface Design

Q1. How would you rate the ease of navigating the AERA Chat platform, including both the Bulk Marking and Chat interfaces? (1 = Very Difficult, 5 = Very Easy)

Q2. How intuitive is the layout for comparing scores and rationales from multiple LLMs side-by-side? (1 = Not Intuitive, 5 = Very Intuitive)

Q3. Is the purpose of the two main interfaces (Bulk Marking vs. Chat) clear and distinct?
- Yes
- No
- Somewhat

Section 2: Automated Assessment and Rationale Quality

Q4. How would you rate the accuracy of the scores generated by the different models? (1 = Very Inaccurate, 5 = Very Accurate)

Q5. Please assess the overall quality of the AI-generated rationales in terms of clarity, correctness, and justification for the score. (1 = Poor, 5 = Excellent)

Q6. How responsive and fast is the system when generating assessments for a batch of student answers? (1 = Very Slow, 5 = Very Fast)

Section 3: Explainability and Verification Tools

Q7. How helpful is the "Key Component Highlighting" feature for understanding and verifying the LLMs' assessment decisions? (1 = Not Helpful, 5 = Very Helpful)

Q8. How easy is it to use the annotation tools (e.g., correcting ground-truth labels, selecting rationale preference, editing rationales)? (1 = Very Difficult, 5 = Very Easy)

Q9. Does AERA Chat's explainability (rationales + highlighting) increase your trust in the automated assessment process compared to a system that only outputs a score?
- Yes, significantly

- Yes, somewhat
- No difference
- No, it reduces my trust

Section 4: Comparative Analysis and Research Utility

Q10. How effective is the platform for comparing the performance and behaviour of different LLMs for student answer assessment? (1 = Not Effective, 5 = Very Effective)

Q11. How useful are the automatically calculated performance metrics (Accuracy, F1, QWK) for your evaluation needs? (1 = Not Useful, 5 = Very Useful)

Q12. Does the platform effectively support the collection of human feedback data (e.g., rationale preferences, corrections) for research purposes? (1 = Not at all, 5 = Extremely well)

Section 5: Overall Experience & Impact

Q13. Before using AERA Chat, how did you typically perform tasks like scoring student answers or evaluating AI-generated explanations? (Select all that apply)
- Manually graded and wrote justifications myself
- Used general-purpose chatbots (e.g., ChatGPT)
- Used other specialized assessment tools
- Did not perform this type of task

Q14. Compared to your previous method(s), what are the primary advantages of using AERA Chat? (Select up to three)
- Speed and Efficiency
- Explainability and Transparency
- Ease of Comparing Different Models
- Facilitates Data Annotation/Collection
- Improved Assessment Accuracy/Consistency
- Other: _____

Q15. For future tasks involving student answer assessment or rationale evaluation, how likely are you to use AERA Chat again?
- I would prefer to use AERA Chat
- I would combine AERA Chat with my previous methods
- I would revert to my previous methods
- I am unsure

RadEval: A framework for radiology text evaluation

Justin Xu[♠,*] **Xi Zhang**[♢] **Javid Abderezaei**[♡] **Julie Bauml**[♡]

Roger Boodoo[♡] **Fatemeh Haghighi**[♡] **Ali Ganjizadeh**[♡]

Eric Brattain[♡] **Dave Van Veen**[♡] **Zaiqiao Meng**[♢]

David Eyre[♠] **Jean-Benoit Delbrouck**[♡,*]

[♠]**University of Oxford** [♢]**University of Glasgow** [♡]**HOPPR**

justin.xu@ndm.ox.ac.uk, jeanbenoit.delbrouck@hoppr.ai

https://github.com/jbdel/RadEval

🤗 https://huggingface.co/IAMJB/RadEvalModernBERT

Abstract

We introduce RadEval, a unified, open-source framework for evaluating radiology texts. RadEval consolidates a diverse range of metrics – from classic n-gram overlap (BLEU, ROUGE) and contextual measures (BERTScore) to clinical concept-based scores (F1CheXbert, F1RadGraph, RaTEScore, SRR-BERT, TemporalEntityF1) and advanced LLM-based evaluators (GREEN). We refine and standardize implementations, extend GREEN to support multiple imaging modalities with a more lightweight model, and pretrain a domain-specific radiology encoder – demonstrating strong zero-shot retrieval performance. We also release a richly annotated expert dataset with over 450 clinically significant error labels and show how different metrics correlate with radiologist judgment. Finally, RadEval provides statistical testing tools and baseline model evaluations across multiple publicly available datasets, facilitating reproducibility and robust benchmarking in radiology report generation.

1 Introduction

Evaluating automated radiology report generation (RRG) systems remains a fundamental challenge in the development of safe, accurate, and clinically useful medical AI. Unlike general-purpose text generation tasks, RRG demands evaluation methods that can assess not only linguistic fluency but also clinical factuality, domain-specific terminology, uncertainty calibration, and diagnostic relevance. In recent years, the evaluation of radiology report generation has steadily progressed: initial studies relied on classic natural language generation (NLG) metrics such as BLEU and ROUGE (Zhang et al., 2020; Chen et al., 2020); subsequent work emphasized clinical accuracy through disease-classification and natural language inference (NLI)-based metrics (Miura et al., 2021); this was followed by expert-annotated semantic graphs capturing entities and their relations (Delbrouck et al., 2022a); and, most recently, by evaluation approaches that leverage large language models (LLM) (Ostmeier et al., 2024; Bannur et al., 2024a; Huang et al., 2024).

Despite efforts to establish fair benchmarking, such as shared metric codebases released for challenge tracks (Abacha et al., 2021; Delbrouck et al., 2023; Xu et al., 2024b) and a public leaderboard (Zhang et al., 2024b), there is still no open-source repository that reproduces the different factuality-focused metrics, whose scores can vary with implementation choices. For instance, earlier studies have computed BERTScore with different pretrained models and settings – varying the number of layers or whether scores are rescaled with a baseline (Zhang et al., 2019) – or swapped in F1CheXbert embeddings for the calculation (Smit et al., 2020b). Variants of the F1RadGraph metric likewise diverge depending on how they judge the correctness of entities and relations (Delbrouck et al., 2022b). Composite scores such as RadCliQ (Yu et al., 2023b) are similarly challenging to replicate.

`RadEval` brings the following solutions:

- **Unified open-source codebase**: every factuality-oriented metric proposed to date is re-implemented in a single, reproducible repository.
- **Metric refinements**: corrected and improved versions of existing metrics offer more faithful

[*]Equal contributions

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 546–557
November 4-9, 2025

estimates of clinical correctness (Section 5 and Appendix B).

- **Expanded expert test set**: an updated, radiologist-annotated corpus enabling fine-grained studies of how automatic metrics align with human judgments (Section 6).

- **Ready-made baselines**: published predictions from several widely-cited models are included so new systems can be benchmarked out-of-the-box.

- **Built-in statistical testing**: permutation and bootstrap tests (mirroring best practices in image captioning) enable users to determine whether score differences are statistically significant.

2 Existing Radiology Report Metrics

2.1 Lexical Overlap Metrics

Early evaluation methods focused on string-level overlap between generated and reference reports, typically using metrics from the natural language processing (NLP) literature. ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002) remain among the most common, measuring n-gram precision and recall. METEOR (Banerjee and Lavie, 2005) and CIDEr (Vedantam et al., 2015) have also been adapted from image captioning literature. These metrics are straightforward to compute and require no domain-specific annotation or models, making them popular baselines.

In addition, BERTScore (Zhang et al., 2019) has been proposed to address limitations of n-gram overlap metrics. Instead of relying on exact token matches, it computes semantic similarity between candidate and reference reports using contextualized embeddings from a pretrained language model (*e.g.*, BERT (Devlin et al., 2019)). Token-level similarity is calculated based on the sum of cosine similarities, offering improved sensitivity to semantic alignment and lexical variation.

However, such metrics perform poorly in RRG settings due to their insensitivity to paraphrasing, semantic equivalence, or clinical correctness. Radiology reports are often sparse, redundant, or variable in linguistic expression, which causes n-gram metrics to underestimate report quality even when the clinical meaning is preserved.

2.2 Clinical Concept-Based Metrics

To improve domain specificity, several works introduced evaluation metrics based on clinical concept extraction and comparison. F1CheXbert (Smit et al., 2020b) utilizes a rule-based labeler that extracts 14 predefined CheXpert disease categories (Irvin et al., 2019) from both reference and generated reports, then computes an F1 score over presence/absence of these labels. SRR-BERT (Delbrouck et al., 2025) expands the label space to 55 labels, and also supports evaluation on structured reports.

F1RadGraph (Delbrouck et al., 2022b) also offers a more expressive alternative by representing reports as structured graphs of anatomical and observational entities and relations. A pretrained graph extraction model is applied to both candidate and reference reports, and graph-level overlap metrics (*e.g.*, precision, recall, F1) are computed. Although F1RadGraph has improved generality and some alignment with radiologist evaluations, it remains limited by the accuracy of the underlying parser and the quality of the training data. While the initial version of F1RadGraph was proposed by Jain et al. (2021), which computed entity and relation overlap separately and reported their average, it did not consider whether entities were matched based on textual spans, semantic types, or shared relations. This simplification may lead to overestimated alignment in complex cases.

RaTEScore (Zhao et al., 2024) identifies medical entities and their types (*e.g.*, anatomy, disease), applies synonym-aware semantic matching, and weights entities by diagnostic importance to better handle terminology variation and negation.

Similar to BERTScore, the CheXbert vector similarity metric (Yu et al., 2022) measures alignment between generated and reference reports by computing the cosine similarity of their embeddings obtained via the CheXbert model (Smit et al., 2020a).

In contrast to these standard clinical metrics, Temporal Entity F1 (Zhang et al., 2025a) is designed to assess temporal information quality in reports. It focuses on capturing the progression or stability of observations (*e.g.*, worsening, improved, or stable) (Bannur et al., 2023), and is particularly useful for detecting temporal hallucinations.

2.3 Composite and Learned Metrics

To improve alignment with human evaluations, ensemble-based or regression-trained metrics such

as RadCliQ (Yu et al., 2023b) have been proposed. RadCliQ uses a linear model trained on radiologist-labeled error counts, combining submetrics like BLEU, BERTScore, CheXbert vector similarity, and F1RadGraph. This strategy improves correlation with expert assessments but introduces interpretability challenges.

2.4 Generative and LLM-Based Evaluation Metrics

More recently, LLM-based generative evaluation has emerged as a promising paradigm for RRG metric design. These approaches leverage foundation models' reasoning and cross-domain generalization to assess report correctness, style, and completeness in a free-text or structured manner.

These LLM-driven evaluators offer high flexibility and often exhibit better alignment with radiologist judgment, especially on nuanced and out-of-distribution findings. For instance, GREEN (Ostmeier et al., 2024) proposed an interpretable and open-source LLM-based evaluation pipeline using a 7B parameter models, and includes a normalized GREEN score, structured error summaries for interpretability, and zero-shot generalization across imaging modalities.

CheXprompt (Zambrano Chaves et al., 2025) is a GPT-based evaluator that detects and categorizes six types of clinically relevant errors: false positive and false negative findings, incorrect location or severity, false positive comparisons, and false negative comparisons.

Similarly, FineRadScore (Huang et al., 2024) evaluates reports line by line, combining clinical severity with the number of incorrect lines. This reflects both the potential clinical risk and the effort required for correction, providing a practical measure of report quality.

RadFact (Bannur et al., 2024a) is also a GPT-based evaluation suite that assesses the factuality of each sentence in a generated report based on the corresponding reference sentences. It supports grounded evaluation and provides interpretable, sentence-level error analysis.

Despite their flexibility and strong alignment with expert judgment, GPT-based evaluation methods face key limitations: high computational cost, deployment barriers due to model size or proprietary APIs, and potential inconsistencies in output. Clinical applications also raise data privacy concerns. RadEval addresses these challenges by standardizing interfaces and supporting lightweight, open-source alternatives like GREEN, enabling local, privacy-preserving, and reproducible evaluation for radiology report generation.

3 RadEval

One of the critical obstacles to building AI systems that can match the accuracy and nuance of expert radiologists is the lack of standardized evaluation metrics. This gap hinders reliable analysis and comparison across different studies, and limits the real-world applicability of research progress.

Despite rapid innovation in metric development, practical barriers remain for adoption and comparison:

- Each metric typically requires separate installation, dependencies, and data pre-processing pipelines.
- Some tools lack public code or require proprietary APIs.
- Evaluation outputs vary in format and interpretability.

To mitigate this fragmentation, we propose a system that consolidates access to a wide range of RRG metrics, spanning from n-gram baselines to modern LLM evaluators. This system is designed to be modular, supporting plug-and-play integration of new metrics. By democratizing access to high-quality RRG evaluation tools, we aim to accelerate research on radiology report generation and encourage more standardized and reproducible benchmarking.

4 Benchmarking

We conducted extensive benchmarking and evaluation of various models on publicly available datasets (Table 3).

4.1 Datasets

For evaluation, we utilized the official test splits of MIMIC-CXR (Johnson et al., 2019b) and ReXGradient-160K (Zhang et al., 2025b), as well as the public validation set of CheXpert Plus (Chambon et al., 2024), as no official test split is available for the latter.

Each study in these datasets may include multiple associated images, all of which were retained for evaluation. Depending on model support, either a single representative image or all available

images were used as input. We focused on specifically evaluating the generation of the "Findings" and "Impression" sections. Reports missing either section were excluded to ensure consistent evaluation across metrics.

MIMIC-CXR A widely used public dataset containing 377,110 chest X-ray images across 227,835 studies, each paired with a radiology report (Johnson et al., 2019a). We use JPEG images from MIMIC-CXR-JPG instead of the original DICOMs. The official test split includes 2,347 studies with "Findings" sections and 2,224 with "Impression" sections.

CheXpert-Plus Comprises 223,462 image–report pairs from 187,711 studies across 64,725 patients (Chambon et al., 2024). We use its validation set, which contains 74 studies with "Findings" and 234 with "Impression" sections.

ReXGradient-160K The largest publicly available chest X-ray dataset to date in terms of patient coverage, including 160,000 image–report pairs from 109,487 patients across 79 U.S. medical sites (Zhang et al., 2025b). Its official test set includes 10,000 studies with both "Findings" and "Impression" sections.

4.2 Baselines

To evaluate the performance of existing RRG systems, we include a representative set of baselines from different institutions and architectures:

CheXpert-Plus (Chambon et al., 2024) Uses a Swinv2-based vision encoder and a two-layer BERT decoder. Two separate models are trained on MIMIC-CXR: one for the *Findings* section and another for the *Impression*.

CheXagent (Chen et al., 2024) An instruction-tuned foundation model for chest X-ray interpretation. It integrates a vision encoder with a cross-modal adapter to align visual and textual information. Training is conducted on CheXinstruct (Chen et al., 2024), a diverse instruction dataset aggregated from 28 open-source medical datasets.

MAIRA-2 (Bannur et al., 2024b) A model designed for grounded radiology report generation, which involves not only producing clinically accurate text but also identifying the spatial locations of findings. It builds on the LLaVA architecture (Liu et al., 2023), incorporating a frozen Rad-DINO-MAIRA-2 vision encoder (Pérez-García et al., 2025), a Vicuna-7B (Zheng et al., 2023) language model, and a four-layer fully connected multilayer perceptron for vision-language alignment.

Libra (Zhang et al., 2025a) A temporally-aware multimodal large language model (MLLM) tailored for generating the *Findings* section in radiology reports. Unlike prior single-image methods, Libra leverages paired chest X-rays to capture disease progression. It integrates a frozen Rad-DINO (Pérez-García et al., 2024) image encoder with the Meditron-7B (Chen et al., 2023) language model via a Temporal Alignment Connector, which combines a Layerwise Feature Extractor and a Temporal Fusion Module to embed multi-scale visual changes over time into the model architecture.

Med-CXRGen (Zhang et al., 2024a) Built on LLaVA-Med (Li et al., 2023), this model uses multi-stage visual instruction tuning and stitches multiple images for unified encoding. Separate models are trained for the "Findings" and "Impression" sections using the RRG24 dataset (Xu et al., 2024a).

5 `RadEvalBERTScore`

In this work, we also introduce a domain-specific radiology language encoder trained using SimCSE (Gao et al., 2022) – a contrastive learning method for sentence embeddings – to capture high-quality representations of radiology report text. We begin with a `ModernBERT-base` architecture and train it on the "Findings" and "Impression" sections from MIMIC-CXR, CheXpert, and ReXGradient-160K. This pretraining setup allows the model to learn clinically meaningful semantic relationships within and across radiology reports.

We demonstrate the effectiveness of our embedding model on a **zero-shot report-to-report retrieval** task. The set-up mirrors a classic text-retrieval scenario:

1. A small set of **query reports** and a larger pool of **candidate reports**, each annotated with one or more radiology labels, are encoded with a frozen text encoder.

2. For every query, we compute the cosine similarity to *all* candidates and obtain a ranked list in descending similarity order.

A candidate is considered relevant to a query if and only if the two reports share at least one label. For each of our experiments, we choose 10 queries and up to 200 positive candidates.

Models	CheXpert 5×200					HOPPR 8×200				
	P@5	P@10	P@50	P@100	mAP	P@5	P@10	P@50	P@100	mAP
Devlin et al. (2019)	46.6 ± 5.5	44.8 ± 4.8	37.0 ± 2.8	33.4 ± 2.1	30.1 ± 1.6	28.1 ± 2.1	24.6 ± 2.0	19.1 ± 1.0	17.3 ± 0.8	15.6 ± 0.4
Warner et al. (2024)	39.4 ± 4.1	35.7 ± 4.1	30.5 ± 2.2	28.3 ± 1.7	26.5 ± 1.2	23.0 ± 2.1	20.3 ± 1.6	16.8 ± 0.5	15.8 ± 0.5	14.6 ± 0.2
Sounack et al. (2025)	38.6 ± 4.6	36.6 ± 3.9	30.6 ± 2.3	28.2 ± 1.6	25.6 ± 0.9	23.4 ± 2.3	21.4 ± 1.6	17.4 ± 1.0	15.9 ± 0.5	14.7 ± 0.3
RadEvalBERT	60.3 ± 3.1	56.4 ± 2.6	46.4 ± 2.0	41.4 ± 1.7	36.4 ± 1.2	38.6 ± 2.0	34.8 ± 2.2	26.9 ± 1.0	23.7 ± 1.0	20.1 ± 0.6

Models	CheXbert Test Set				
	P@5	P@10	nDCG@5	nDCG@10	mAP
Devlin et al. (2019)	64.3 ± 2.0	59.2 ± 1.8	54.5 ± 1.3	48.7 ± 1.0	50.1 ± 1.1
Warner et al. (2024)	62.0 ± 2.1	57.2 ± 1.3	51.8 ± 1.5	46.2 ± 0.9	48.7 ± 1.0
Sounack et al. (2025)	61.8 ± 1.0	56.8 ± 1.0	51.2 ± 1.0	45.8 ± 0.8	48.3 ± 0.9
RadEvalBERT	70.2 ± 2.2	64.6 ± 1.5	58.9 ± 1.6	53.3 ± 1.1	53.4 ± 1.0

Table 1: Zero-shot report-to-report retrieval performance of RadEvalBERTScore Evaluation. We evaluate on three datasets: CheXpert 5×200 (five single-label categories), HOPPR 8×200 (eight out-of-domain single-label categories), and the CheXbert multi-label test set. For CheXpert and HOPPR, we report Precision@$\{5, 10, 50, 100\}$ and mean Average Precision (mAP); for CheXbert we report Precision@$\{5, 10\}$, normalized Discounted Cumulative Gain (nDCG@$\{5, 10\}$), and mAP. Values are presented as mean ± standard deviation over 10 random seeds.

5.1 Metrics

Precision@K

Fraction of the first K retrieved reports that are relevant. A hit is counted as soon as *one* label overlaps with the query.

mean Average Precision (mAP)

For each query, we compute *Average Precision* (*i.e.*, the mean of the precision values at every rank where a relevant report occurs. The scan stops at the last relevant rank, so items appearing afterwards cannot influence the score. mAP is the mean AP over all queries and reflects how early, on average, relevant reports are surfaced.

nDCG@K

A graded variant that rewards richer matches. The gain between a query and a candidate is the number of shared labels. DCG is accumulated over the top K positions with a logarithmic discount $1/\log_2(\text{rank}+1)$; nDCG normalises by the ideal DCG, yielding a score in $[0, 1]$ where 1 means the system ranks the strongest overlaps highest.

5.2 Datasets

CheXpert 5×200

Five single-label categories (Atelectasis, Cardiomegaly, Edema, Consolidation, Pleural Effusion).

→We report Precision@$\{5, 10, 50, 100\}$ and mAP.

CheXbert Test Set

Multi-label reports with 14 chexpert possible findings (*e.g.*, Airspace Opacity, Pneumonia, Support Devices).

→Because at most 20 positives exist per query, we report Precision@$\{5, 10\}$ and nDCG@$\{5, 10\}$ together with mAP.

HOPPR 8×200

This is an out-of-domain, single-label dataset used to evaluate generalization performance. The label categories include: acute rib fracture, air space opacity, cardiomegaly, lung nodule or mass, non acute rib fracture, pleural fluid, pneumothorax, and pulmonary artery enlargement.

→We report Precision@$\{5, 10, 50, 100\}$ and mAP.

This protocol provides complementary views of retrieval quality: Precision@K for top-K exact-hit rate, mAP for overall early-ranking performance, and nDCG@K for sensitivity to *how many* labels overlap.

6 RadEval Expert Dataset

Dataset. We release an updated RadEval-expert dataset with board-certified radiologists annotating clinically significant and insignificant errors across different error categories. Building on ReXVal (Yu et al., 2023a), we annotate false predictions of findings, omissions of findings, incorrect locations/positions, incorrect severities, spurious comparisons, omissions of changes from prior studies, as well

Table 2: Top-3 metrics by Kendall's τ_b (more negative is better; higher metric ⇒ fewer errors). "✓ aligned" = 95% CI < 0; "✗ misaligned" = 95% CI > 0; "ns" = CI overlaps 0. **Scope:** pooled rows show `(pairs, n)`; blocked rows show `(blocks, pairs)`. Each study has K=3 candidates (Findings: 148 studies; Impression: 60).

Endpoint	Metric	τ_b [95% CI]	Sig	Scope
Overall (pooled) vs. total significant errors				
	green	$-0.183\,[-0.246, -0.122]$	✓ aligned	(pairs 194,376, n 624)
	srr_bert	$-0.133\,[-0.193, -0.071]$	✓ aligned	(pairs 194,376, n 624)
	radcliq	$-0.107\,[-0.160, -0.052]$	✓ aligned	(pairs 194,376, n 624)
	radevalbertscore	$-0.076\,[-0.131, -0.017]$	✓ aligned	(pairs 194,376, n 624)
	rouge	$-0.038\,[-0.091, 0.017]$	ns	(pairs 194,376, n 624)
	bertscore	$-0.027\,[-0.085, 0.032]$	ns	(pairs 194,376, n 624)
	radgraph	$-0.011\,[-0.068, 0.048]$	ns	(pairs 194,376, n 624)
	bleu	$0.034\,[-0.020, 0.086]$	ns	(pairs 194,376, n 624)
	chexbert	$0.074\,[0.012, 0.137]$	✗ misaligned	(pairs 194,376, n 624)
ALL (blocked) vs. total significant errors				
	green	$-0.195\,[-0.295, -0.087]$	✓ aligned	(blocks 159, pairs 477)
	bertscore	$-0.160\,[-0.260, -0.059]$	✓ aligned	(blocks 188, pairs 564)
	bleu	$-0.153\,[-0.273, -0.029]$	✓ aligned	(blocks 89, pairs 267)
ALL (blocked) vs. total insignificant errors				
	radcliq	$-0.039\,[-0.147, 0.076]$	ns	(blocks 143, pairs 429)
	radevalbertscore	$-0.009\,[-0.126, 0.103]$	ns	(blocks 133, pairs 399)
	bertscore	$-0.008\,[-0.107, 0.094]$	ns	(blocks 143, pairs 429)
Impression only (blocked) vs. total significant errors				
	bertscore	$-0.225\,[-0.399, -0.049]$	✓ aligned	(blocks 57, pairs 171)
	rouge	$-0.215\,[-0.383, -0.042]$	✓ aligned	(blocks 57, pairs 171)
	radevalbertscore	$-0.206\,[-0.399, -0.010]$	✓ aligned	(blocks 53, pairs 159)
Findings only (blocked) vs. total significant errors				
	green	$-0.196\,[-0.314, -0.075]$	✓ aligned	(blocks 113, pairs 339)
	bleu	$-0.168\,[-0.313, -0.020]$	✓ aligned	(blocks 68, pairs 204)
	bertscore	$-0.132\,[-0.246, -0.017]$	✓ aligned	(blocks 131, pairs 393)
Per-category (blocked, significant-error endpoints)				
significant: false prediction of finding	green	$-0.082\,[-0.198, 0.030]$	ns	(blocks 134, pairs 400)
	radcliq	$-0.052\,[-0.157, 0.052]$	ns	(blocks 162, pairs 482)
	bertscore	$-0.049\,[-0.158, 0.061]$	ns	(blocks 162, pairs 482)
significant: omission of finding	srr_bert	$-0.512\,[-0.625, -0.390]$	✓ aligned	(blocks 91, pairs 271)
	chexbert	$-0.503\,[-0.626, -0.366]$	✓ aligned	(blocks 89, pairs 267)
	green	$-0.250\,[-0.374, -0.126]$	✓ aligned	(blocks 113, pairs 337)
significant: incorrect location/ position of finding	bertscore	$-0.068\,[-0.213, 0.079]$	ns	(blocks 80, pairs 238)
	radevalbertscore	$-0.040\,[-0.201, 0.123]$	ns	(blocks 76, pairs 226)
	rouge	$-0.018\,[-0.174, 0.139]$	ns	(blocks 80, pairs 238)
significant: incorrect severity of finding	radevalbertscore	$-0.033\,[-0.223, 0.160]$	ns	(blocks 56, pairs 168)
	bleu	$-0.001\,[-0.235, 0.219]$	ns	(blocks 35, pairs 105)
	rouge	$0.007\,[-0.170, 0.191]$	ns	(blocks 61, pairs 181)
significant: spurious comparison (not in reference)	bertscore	$-0.153\,[-0.300, 0.001]$	ns	(blocks 81, pairs 241)
	radcliq	$-0.125\,[-0.263, 0.014]$	ns	(blocks 81, pairs 241)
	radevalbertscore	$-0.103\,[-0.247, 0.063]$	ns	(blocks 77, pairs 229)
significant: omission of change from previous study	bleu	$-0.127\,[-0.335, 0.099]$	ns	(blocks 37, pairs 111)
	green	$-0.066\,[-0.241, 0.113]$	ns	(blocks 65, pairs 195)
	radgraph	$-0.027\,[-0.199, 0.137]$	ns	(blocks 61, pairs 183)
significant: inarticulate report (grammar/readability)	radcliq	$-0.350\,[-0.560, -0.140]$	✓ aligned	(blocks 35, pairs 105)
	radevalbertscore	$-0.266\,[-0.476, -0.046]$	✓ aligned	(blocks 34, pairs 102)
	bertscore	$-0.251\,[-0.480, -0.013]$	✓ aligned	(blocks 35, pairs 105)

as a new category: **inarticulate report/grammar**. All error categories are labeled as either significant or insignificant. We also extend beyond the "Impression" to also cover the "Findings"' section. The corpus comprises **208 studies** (**148 findings** and **60 impressions**), and **each study has exactly** $K{=}3$ **annotated candidate reports per ground truth**. Ground-truth reports come from MIMIC-CXR, CheXpert-Plus, and ReXGradient-160K, and candidate reports are generated by CheXagent, the CheXpert-Plus model, and MAIRA-2.

Methods. We measure agreement between automatic metrics and radiologists' judgments using Kendall's τ_b (Kendall, 1945) against radiologist error counts. We report: (1) a **pooled** (overall) τ_b versus the **total number of significant errors** (treating each candidate independently; ignores study grouping), and (2) a **blocked** (within-study, tie-aware) τ_b with 95% block-bootstrap confidence intervals obtained by resampling study blocks (preserving section type). Blocked analyses cover: totals of significant and insignificant errors across all sections ("ALL"), the significant-error total within each section (Impression, Findings), and each individual significant-error category. We label ✓ aligned when the 95% CI lies entirely below 0 (higher metric ⇒ fewer errors), ✗ misaligned when the CI lies entirely above 0 (higher scores ⇒ more errors), and **ns** otherwise.

Results. Overall (pooled vs. total significant errors), *green*, *srr_bert*, *radcliq*, and *radeval-bertscore* show meaningful negative correlations (✓aligned), while *rouge*, *bertscore*, *radgraph*, and *bleu* are not significant; *chexbert* is ✗misaligned (higher scores with more errors). Within studies (blocked, ALL), *green*, *bertscore*, and *bleu* are top for total significant errors (all ✓aligned), and no metric tracks total *insignificant* errors (all ns).

Table 2 reports correlations by section and across different error category types. From these results, *green* emerges as the most reliable single metric for tracking clinically significant errors, followed by *srr_bert*.

7 Conclusion

We introduced RadEval, a unified framework for evaluating RRG. By consolidating and standardizing a diverse suite of evaluation metrics, including lexical overlap, clinical concept extraction, structured graph comparison, and LLM-based scoring, RadEval addresses longstanding reproducibility and benchmarking challenges in the RRG domain. We refined existing metrics, released a high-fidelity expert-annotated test set, and benchmarked state-of-the-art report generation systems across multiple publicly available datasets. In addition, we demonstrated the utility of a new domain-specific sentence encoder through a zero-shot retrieval task and introduced an updated lightweight version of the GREEN metric capable of cross-modality evaluation. RadEval's modular architecture will help facilitate robust, reproducible, and clinically grounded evaluation – ultimately helping accelerate the safe deployment of radiology AI systems.

Limitations

While RadEval already unifies a broad set of automated radiology text evaluation metrics, recently proposed LLM-based metrics (*e.g.*, CheXprompt, FineRadScore, and RadFact) are not currently implemented due to their reliance on LLM APIs or lack of standardization – though they remain valuable future additions. Additionally, some metrics depend on upstream parsers that may introduce noise. Current evaluations also focus primarily on chest X-ray radiology reports written in English from institutions in the U.S., limiting generalizability across languages and geographical regions. Finally, we aim to continue to expand the expert-labeled dataset with additional reports and clinical annotators, as well as to compute detailed correlation analyses across all automated metrics and annotations to better assess metric alignment with clinical judgment.

References

Asma Ben Abacha, Yassine Mrabet, Yuhao Zhang, Chaitanya Shivade, Curtis Langlotz, and Dina Demner-Fushman. 2021. Overview of the mediqa 2021 shared task on summarization in the medical domain. *NAACL-HLT 2021*, page 74.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Shruthi Bannur, Kenza Bouzid, Daniel C Castro, Anton Schwaighofer, Anja Thieme, Sam Bond-Taylor, Maximilian Ilse, Fernando Pérez-García, Valentina Sal-

vatelli, Harshita Sharma, and 1 others. 2024a. Maira-2: Grounded radiology report generation. *arXiv preprint arXiv:2406.04449*.

Shruthi Bannur, Kenza Bouzid, Daniel C. Castro, Anton Schwaighofer, Anja Thieme, Sam Bond-Taylor, Maximilian Ilse, Fernando Pérez-García, Valentina Salvatelli, Harshita Sharma, Felix Meissen, Mercy Ranjit, Shaury Srivastav, Julia Gong, Noel C. F. Codella, Fabian Falck, Ozan Oktay, Matthew P. Lungren, Maria Teodora Wetscherek, and 2 others. 2024b. Maira-2: Grounded radiology report generation. *Preprint*, arXiv:2406.04449.

Shruthi Bannur, Stephanie Hyland, Qianchu Liu, Fernando Pérez-García, Maximilian Ilse, Daniel C. Castro, Benedikt Boecking, Harshita Sharma, Kenza Bouzid, Anja Thieme, Anton Schwaighofer, Maria Wetscherek, Matthew P. Lungren, Aditya Nori, Javier Alvarez-Valle, and Ozan Oktay. 2023. Learning to exploit temporal structure for biomedical vision-language processing. *Preprint*, arXiv:2301.04558.

Pierre Chambon, Jean-Benoit Delbrouck, Thomas Sounack, Shih-Cheng Huang, Zhihong Chen, Maya Varma, Steven QH Truong, Chu The Chuong, and Curtis P. Langlotz. 2024. Chexpert plus: Augmenting a large chest x-ray dataset with text radiology reports, patient demographics and additional image formats. *Preprint*, arXiv:2405.19538.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. 2023. Meditron-70b: Scaling medical pretraining for large language models. *Preprint*, arXiv:2311.16079.

Zhihong Chen, Yan Song, Tsung-Hui Chang, and Xiang Wan. 2020. Generating radiology reports via memory-driven transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1439–1449.

Zhihong Chen, Maya Varma, Justin Xu, Magdalini Paschali, Dave Van Veen, Andrew Johnston, Alaa Youssef, Louis Blankemeier, Christian Bluethgen, Stephan Altmayer, Jeya Maria Jose Valanarasu, Mohamed Siddig Eltayeb Muneer, Eduardo Pontes Reis, Joseph Paul Cohen, Cameron Olsen, Tanishq Mathew Abraham, Emily B. Tsai, Christopher F. Beaulieu, Jenia Jitsev, and 4 others. 2024. A vision-language foundation model to enhance efficiency of chest x-ray interpretation. *Preprint*, arXiv:2401.12208.

Jean-Benoit Delbrouck, Pierre Chambon, Christian Bluethgen, Emily Tsai, Omar Almusa, and Curtis Langlotz. 2022a. Improving the factual correctness of radiology report generation with semantic rewards. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4348–4360.

Jean-Benoit Delbrouck, Pierre Chambon, Christian Bluethgen, Emily Tsai, Omar Almusa, and Curtis Langlotz. 2022b. Improving the factual correctness of radiology report generation with semantic rewards. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4348–4360, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jean-Benoit Delbrouck, Maya Varma, Pierre Chambon, and Curtis Langlotz. 2023. Overview of the radsum23 shared task on multi-modal and multi-anatomical radiology report summarization. In *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, pages 478–482.

Jean-Benoit Delbrouck, Justin Xu, Johannes Moll, Alois Thomas, Zhihong Chen, Sophie Ostmeier, Asfandyar Azhar, Kelvin Zhenghao Li, Andrew Johnston, Christian Bluethgen, Eduardo Reis, Mohamed Muneer, Maya Varma, and Curtis Langlotz. 2025. Automated structured radiology report generation. *Preprint*, arXiv:2505.24223.

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings. *Preprint*, arXiv:2104.08821.

Alyssa Huang, Oishi Banerjee, Kay Wu, Eduardo Pontes Reis, and Pranav Rajpurkar. 2024. Fineradscore: A radiology report line-by-line evaluation technique generating corrections with severity scores. *arXiv preprint arXiv:2405.20613*.

Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. 2019. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *Preprint*, arXiv:1901.07031.

Saahil Jain, Ashwin Agrawal, Adriel Saporta, Steven QH Truong, Du Nguyen Duong, Tan Bui, Pierre Chambon, Yuhao Zhang, Matthew P. Lungren, Andrew Y. Ng, Curtis P. Langlotz, and Pranav Rajpurkar. 2021. Radgraph: Extracting clinical entities and relations from radiology reports. *Preprint*, arXiv:2106.14463.

Alistair E. W. Johnson, Tom J. Pollard, Nathaniel R. Greenbaum, Matthew P. Lungren, Chih ying Deng, Yifan Peng, Zhiyong Lu, Roger G. Mark, Seth J.

Berkowitz, and Steven Horng. 2019a. Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. *Preprint*, arXiv:1901.07042.

Alistair EW Johnson, Tom J Pollard, Seth J Berkowitz, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Roger G Mark, and Steven Horng. 2019b. Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. *Scientific data*, 6(1):317.

M. G. Kendall. 1945. The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. 2023. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Preprint*, arXiv:2306.00890.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Preprint*, arXiv:2304.08485.

Yasuhide Miura, Yuhao Zhang, Emily Tsai, Curtis Langlotz, and Dan Jurafsky. 2021. Improving factual completeness and consistency of image-to-text radiology report generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5288–5304.

Sophie Ostmeier, Justin Xu, Zhihong Chen, Maya Varma, Louis Blankemeier, Christian Bluethgen, Arne Edward Michalson Md, Michael Moseley, Curtis Langlotz, Akshay S Chaudhari, and Jean-Benoit Delbrouck. 2024. GREEN: Generative radiology report evaluation and error notation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 374–390, Miami, Florida, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Fernando Pérez-García, Harshita Sharma, Sam Bond-Taylor, Kenza Bouzid, Valentina Salvatelli, Maximilian Ilse, Shruthi Bannur, Daniel C. Castro, Anton Schwaighofer, Matthew P. Lungren, Maria Wetscherek, Noel Codella, Stephanie L. Hyland, Javier Alvarez-Valle, and Ozan Oktay. 2024. Raddino: Exploring scalable medical image encoders beyond text supervision. *Preprint*, arXiv:2401.10815.

Fernando Pérez-García, Harshita Sharma, Sam Bond-Taylor, Kenza Bouzid, Valentina Salvatelli, Maximilian Ilse, Shruthi Bannur, Daniel C. Castro, Anton Schwaighofer, Matthew P. Lungren, Maria Teodora Wetscherek, Noel Codella, Stephanie L. Hyland, Javier Alvarez-Valle, and Ozan Oktay. 2025. Exploring scalable medical image encoders beyond text supervision. *Nature Machine Intelligence*, 7(1):119–130.

Akshay Smit, Saahil Jain, Pranav Rajpurkar, Anuj Pareek, Andrew Ng, and Matthew Lungren. 2020a. Combining automatic labelers and expert annotations for accurate radiology report labeling using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1500–1519, Online. Association for Computational Linguistics.

Akshay Smit, Saahil Jain, Pranav Rajpurkar, Anuj Pareek, Andrew Y Ng, and Matthew Lungren. 2020b. Combining automatic labelers and expert annotations for accurate radiology report labeling using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1500–1519.

Thomas Sounack, Joshua Davis, Brigitte Durieux, Antoine Chaffin, Tom J Pollard, Eric Lehman, Alistair EW Johnson, Matthew McDermott, Tristan Naumann, and Charlotta Lindvall. 2025. Bioclinical modernbert: A state-of-the-art long-context encoder for biomedical and clinical nlp. *arXiv preprint arXiv:2506.10896*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, and 1 others. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.

Justin Xu, Zhihong Chen, Andrew Johnston, Louis Blankemeier, Maya Varma, Jason Hom, William J. Collins, Ankit Modi, Robert Lloyd, Benjamin Hopkins, Curtis Langlotz, and Jean-Benoit Delbrouck. 2024a. Overview of the first shared task on clinical text generation: RRG24 and "discharge me!". In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 85–98, Bangkok, Thailand. Association for Computational Linguistics.

Justin Xu, Zhihong Chen, Andrew Johnston, Louis Blankemeier, Maya Varma, Jason Hom, William J Collins, Ankit Modi, Robert Lloyd, Benjamin Hopkins, and 1 others. 2024b. Overview of the first shared task on clinical text generation: Rrg24 and

"discharge me!". In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 85–98.

F. Yu, M. Endo, R. Krishnan, I. Pan, A. Tsai, E. P. Reis, E. Kaiser Ururahy Nunes Fonseca, H. Lee, Z. Shakeri, A. Ng, C. Langlotz, V. K. Venugopal, and P. Rajpurkar. 2023a. Radiology report expert evaluation (rexval) dataset (version 1.0.0). RRID:SCR_007345.

Feiyang Yu, Mark Endo, Rayan Krishnan, Ian Pan, Andy Tsai, Eduardo Pontes Reis, Eduardo Kaiser Ururahy Nunes Fonseca, Henrique Min Ho Lee, Zahra Shakeri Hossein Abad, Andrew Y. Ng, Curtis P. Langlotz, Vasantha Kumar Venugopal, and Pranav Rajpurkar. 2022. Evaluating progress in automatic chest x-ray radiology report generation. *medRxiv*.

Feiyang Yu, Mark Endo, Rayan Krishnan, Ian Pan, Andy Tsai, Eduardo Pontes Reis, Eduardo Kaiser Ururahy Nunes Fonseca, Henrique Min Ho Lee, Zahra Shakeri Hossein Abad, Andrew Y Ng, and 1 others. 2023b. Evaluating progress in automatic chest x-ray radiology report generation. *Patterns*, 4(9).

Juan Manuel Zambrano Chaves, Shih-Cheng Huang, Yanbo Xu, Hanwen Xu, Naoto Usuyama, Sheng Zhang, Fei Wang, Yujia Xie, Mahmoud Khademi, Ziyi Yang, and 1 others. 2025. A clinically accessible small multimodal radiology model and evaluation metric for chest x-ray findings. *Nature Communications*, 16(1):3108.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Xi Zhang, Zaiqiao Meng, Jake Lever, and Edmond S. L. Ho. 2025a. Libra: Leveraging temporal images for biomedical radiology analysis. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17275–17303, Vienna, Austria. Association for Computational Linguistics.

Xi Zhang, Zaiqiao Meng, Jake Lever, and Edmond S.L. Ho. 2024a. Gla-AI4BioMed at RRG24: Visual instruction-tuned adaptation for radiology report generation. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 624–634, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoman Zhang, Julián N. Acosta, Josh Miller, Ouwen Huang, and Pranav Rajpurkar. 2025b. Rexgradient-160k: A large-scale publicly available dataset of chest radiographs with free-text reports. *Preprint*, arXiv:2505.00228.

Xiaoman Zhang, Hong-Yu Zhou, Xiaoli Yang, Oishi Banerjee, Julián N Acosta, Josh Miller, Ouwen Huang, and Pranav Rajpurkar. 2024b. Rexrank: A public leaderboard for ai-powered radiology report generation. *arXiv preprint arXiv:2411.15122*.

Yuhao Zhang, Derek Merck, Emily Tsai, Christopher D Manning, and Curtis Langlotz. 2020. Optimizing the factual correctness of a summary: A study of summarizing radiology reports. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5108–5120.

Weike Zhao, Chaoyi Wu, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2024. RaTEScore: A metric for radiology report generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15004–15019, Miami, Florida, USA. 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. *Preprint*, arXiv:2306.05685.

A Score table

Table 3: Benchmarking results across multiple models on standard datasets under default system prompts.

Models	GREEN Llama	ROUGE L	F1 RG	RG ER	RG $\overline{ER}$	BLEU	BERT Score	SRR BERT	F1cXb 5	F1cXb 14	RaTE Score	Rad CLiQ v1	Temp. Entity F1	Rad Eval BERT
CheXpert-Plus Validation Findings														
CheXagent	26.2	21.4	27.4	24.8	17.6	4.1	33.9	50.5	55.8	58.1	51.1	67.9	40.0	8.2
CheXpert-Plus	25.2	21.7	25.8	22.9	17.2	4.7	47.1	46.9	48.7	52.8	51.0	77.3	34.4	12.7
Med-CXRGen-F	25.5	22.1	23.7	21.2	15.5	6.4	48.0	44.5	41.2	47.6	50.1	75.3	32.6	14.2
Libra-v1.0-3B	21.9	18.8	18.6	15.9	11.0	2.8	42.9	47.1	50.9	50.7	47.0	65.1	19.1	10.9
Libra-v1.0-7B	22.0	18.0	17.9	16.2	11.0	2.5	43.8	44.1	47.8	49.1	47.3	65.6	26.0	11.5
MAIRA-2	15.5	16.4	14.6	13.3	9.2	1.9	41.8	41.0	44.0	47.1	44.0	63.6	29.1	10.7
CheXpert-Plus Validation Impressions														
CheXpert-Plus	23.7	25.3	3.4	3.1	8.0	2.9	51.2	44.5	45.7	49.0	47.7	48.8	35.6	14.8
Med-CXRGen-I	26.6	23.6	18.7	16.7	12.6	6.5	50.8	44.4	48.2	48.3	45.5	64.6	35.3	20.0
MIMIC-CXR Test Findings														
CheXagent	29.6	23.1	26.8	24.2	17.4	4.9	39.0	49.7	60.1	55.1	56.0	71.6	22.5	37.2
CheXpert-Plus	30.6	25.7	24.2	22.1	17.0	5.7	54.2	48.2	54.1	47.4	54.2	84.6	22.5	46.8
Med-CXRGen-F	28.1	22.7	20.9	18.6	13.8	5.9	50.4	44.6	53.3	45.2	52.4	75.0	16.8	44.1
Libra-v1.0-3B	29.2	21.7	20.3	18.0	12.9	5.1	49.4	46.4	60.0	52.5	53.5	76.0	14.5	46.4
Libra-v1.0-7B	28.0	20.9	19.9	17.5	12.3	4.6	49.5	45.8	61.5	53.7	52.7	75.1	18.0	46.5
MAIRA-2	22.4	18.5	17.3	15.3	10.9	3.0	47.6	42.8	59.1	50.7	51.0	71.1	16.4	40.2
MIMIC-CXR Test Impressions														
CheXpert-Plus	25.0	23.6	22.6	20.1	16.8	2.9	46.2	36.8	50.4	45.3	46.4	84.6	37.7	29.9
Med-CXRGen-I	20.8	14.6	13.5	11.7	8.5	2.3	39.3	35.4	49.1	41.5	43.4	61.4	16.7	9.7
ReXGradient-160K Test Findings														
CheXagent	44.5	21.7	25.9	24.5	21.0	3.3	39.2	41.0	4.5	5.5	59.2	69.7	57.1	36.4
CheXpert-Plus	33.6	23.5	22.0	20.7	16.8	3.5	50.9	45.4	16.5	21.0	55.0	76.7	45.8	36.5
Med-CXRGen-F	29.6	14.7	10.1	9.2	4.4	1.6	42.0	39.7	0.7	13.0	43.6	65.8	62.1	33.1
Libra-v1.0-7B	44.8	21.9	20.5	18.5	13.1	3.8	51.0	45.7	7.4	15.7	56.2	78.7	40.5	40.6
MAIRA-2	36.5	21.4	20.1	18.3	14.3	2.3	48.9	43.0	4.1	14.0	52.9	77.9	60.8	35.5
ReXGradient-160K Test Impressions														
CheXpert-Plus	3.8	7.6	2.5	1.9	1.2	0.2	30.3	37.8	16.3	9.4	30.1	51.2	21.2	18.7
Med-CXRGen-I	22.3	8.2	3.9	3.6	2.4	0.4	31.5	40.6	2.6	24.4	31.4	51.7	55.6	10.5

B Refined GREEN Metric

To extend the capabilities of the GREEN metric, we finetuned a compact Gemma-2B model on the original GREEN dataset along with an additional 50,000 annotated radiology report pairs spanning multiple imaging modalities, including CT, MRI, and ultrasound. This represents an evolution beyond the original implementation, which focused exclusively on chest X-rays. By leveraging a smaller, more lightweight language model, we achieve substantial improvements in computational efficiency – averaging inference times of 2–3 seconds per report – while maintaining performance comparable to larger models such as Llama and Phi. This reduced resource footprint makes the updated GREEN model more practical for large-scale or real-time deployment settings. Our work underscores the potential for continued improvement by incorporating more diverse imaging contexts and exploring even lighter architectures without sacrificing clinical alignment or interpretability.

TINYSCIENTIST: An Interactive, Extensible, and Controllable Framework for Building Research Agents

Haofei Yu[1]* **Keyang Xuan**[1]* **Fenghai Li**[1]*
Kunlun Zhu[1] **Zijie Lei**[1] **Jiaxun Zhang**[1] **Ziheng Qi**[1]
Kyle Richardson[2] **Jiaxuan You**[1]
[1]University of Illinois Urbana-Champaign,
[2]Allen Institute for Artificial Intelligence

Abstract

Automatic research with Large Language Models (LLMs) is rapidly gaining importance, driving the development of increasingly complex workflows involving multi-agent systems, planning, tool usage, code execution, and human-agent interaction to accelerate research processes. However, as more researchers and developers begin to use and build upon these tools and platforms, the complexity and difficulty of extending and maintaining such agentic workflows have become a significant challenge, particularly as algorithms and architectures continue to advance. To address this growing complexity, TINYSCIENTIST identifies the essential components of the automatic research workflow and proposes an interactive, extensible, and controllable framework that adapts easily to new tools and supports iterative growth. We provide an open-source codebase[1], an interactive web demonstration[2], and a PyPI Python package[3] to make state-of-the-art auto-research pipelines broadly accessible to every researcher and developer.

1 Introduction

Interest in building research agents with Large Language Models (LLMs) to interact with human researchers and enable automatic scientific discovery has gained considerable attention in recent years (Gottweis et al., 2025). Such agentic frameworks have demonstrated impressive capabilities across a wide range of research tasks, including ideation (Si et al., 2024; Li et al., 2024a), scientific coding (Chan et al., 2024; Huang et al., 2023), paper writing (Wang et al., 2024), review writing (Jin et al., 2024), and even end-to-end research pipelines (Jansen et al., 2025; Lu et al., 2024; Yamada et al., 2025; Li et al., 2024b; Cheng et al., 2025). Recent advances in this area leverage methods including multi-agent collaboration (Schmidgall et al., 2025), tool using (Skarlinski et al., 2024), and tree-based search (Yamada et al., 2025) to augment its performance.

In spite of this success, however, existing automatic research systems often design and use agentic frameworks that are overly complex and difficult to use and extend without significant technical expertise. These challenges stem from three key issues: (1) ***lack of interactivity***: human researchers struggle to engage with the specific agent's research progress due to the complexity of research intents and unclear communication interfaces (Zou et al., 2025; Liu et al., 2025b), making feedback incorporation challenging. (2) ***limited extensibility***: existing representative frameworks rely on rigid, tool-specific designs (Zhang et al., 2025), making it hard to integrate new tools or adapt to different research domains. (3) ***insufficient controllability***: many systems offer weak supervision on safety, ethical constraints, and cost budget, raising concerns about misalignment and unbounded execution (Gridach et al., 2025; Liu et al., 2025a). To address these issues and help democratize the development and use of research agents, we introduce TINYSCIENTIST, a lightweight and modular agentic framework that facilitates interactivity, extensibility, and controllability, making it highly accessible to users, researchers, and developers. Specifically, TINYSCIENTIST is designed based on the following principles illustrated in Figure 1.

Interactivity. Research is an open-ended process that requires continuous user involvement. Researchers typically begin with vague or evolving goals and refine them over time. As a result, effective research agents must support real-time adjustments to their reasoning, coding, and writing

*Core Contributors.

[1]The codebase for TINYSCIENTIST is available at https://github.com/ulab-uiuc/tiny-scientist.

[2]The web demonstration for TINYSCIENTIST is hosted at https://app.auto-research.dev.

[3]The Python package for TINYSCIENTIST is released at https://pypi.org/project/tiny-scientist.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 558–590
November 4-9, 2025

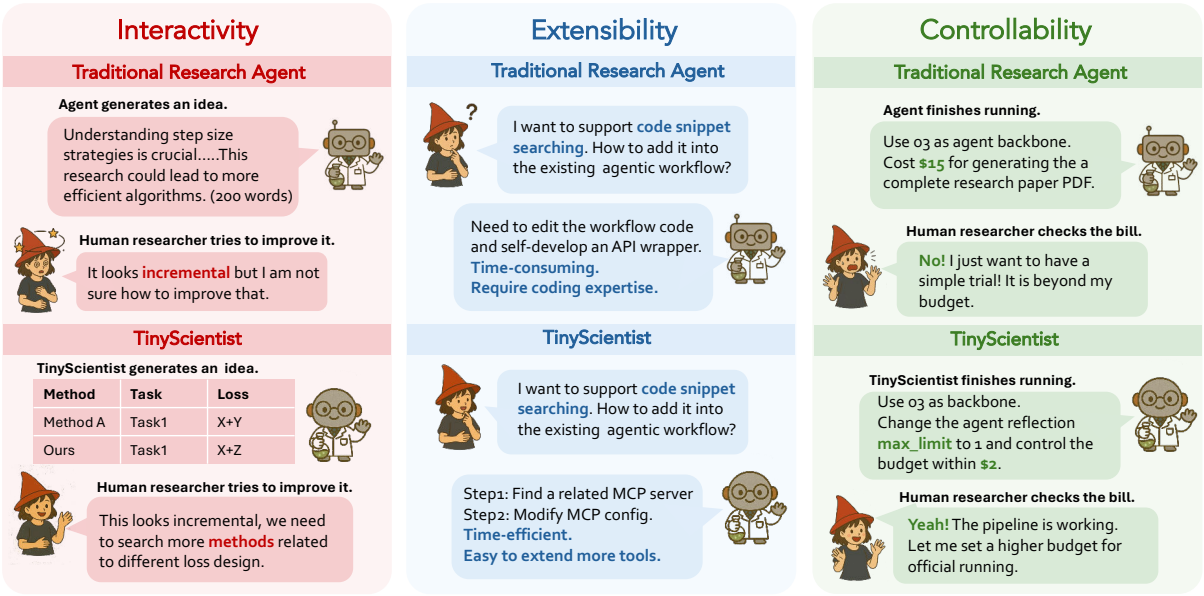


Figure 1: **Design principles for TINYSCIENTIST.** We highlight the key differences between traditional research agents and TINYSCIENTIST. To enhance *interactivity*, TINYSCIENTIST introduces a table-based interface that helps researchers clearly express and refine their intents. For *extensibility*, TINYSCIENTIST adopts an MCP (Model Context Protocol) design instead of direct API wrapping, making it easy to add or replace tools. For *controllability*, TINYSCIENTIST includes built-in safety and budget controllers that monitor and regulate the entire workflow.

processes. Without an interactive interface, agentic frameworks risk drifting from the user's intent and producing irrelevant or unsafe outputs. To address this, TINYSCIENTIST introduces a modular, tabular-based interface that decomposes the research workflow into editable stages. Each stage presents intermediate results in a structured tabular format, allowing researchers to directly modify specific cells or columns—*e.g.*, by suggesting adding new baselines as rows or editing individual entries. Such a design improves clarity in human-agent communication and empowers users to guide the system as their objectives evolve.

Extensibility. Automatic research is evolving rapidly, with new tools and technologies emerging constantly. To keep pace, agentic frameworks need to support the easy integration and replacement of tools. In machine learning-related automatic research, while the core workflow—typically composed of stages like think, code, write, and review—remains relatively fixed, the key difference between different tasks lies in the tools and methods used within each stage. To address this, TINYSCIENTIST adopts the design of the Model Context Protocol (MCP) (Anthropic, 2024), which provides a unified API for connecting diverse tools to augment each core workflow component. Such a system architecture enables seamless extension, allowing developers to upgrade and maintain their system with little effort.

Controllability. Safety, ethical, and financial concerns are critical, but often under-addressed in agentic research workflows (Zhu et al., 2025a; Tang et al., 2024). Users should not be caught off guard by excessive spending or unsafe outputs. Users should not be exposed to unexpected costs, unsafe behaviors, or misaligned actions. To ensure responsible and predictable execution, TINYSCIENTIST emphasizes controllability across the entire pipeline. Users are allowed to set explicit upper bounds on key hyperparameters—such as the number of experimental runs or self-reflection steps—to ensure budget constraints are respected. Additionally, at each stage of the workflow, built-in safety checkers validate outputs to prevent harmful or unintended behavior, which helps to maintain alignment with user intents.

To demonstrate the effectiveness of TINYSCIENTIST, we conduct both qualitative and quantitative evaluations, highlighting four key advantages: (1) It is easy for researchers to use the Python package without configuration or setup barriers. (2) It enables better human–agent interaction through an interactive UI. (3) It achieves research generation quality comparable to Agent Laboratory (Schmidgall et al., 2025), a widely used multi-agent auto-research framework, and (4) using tools improves the generation quality.

Framework	Interactivity		Extensibility		Controllability		Deployment	
	Modular Design	Tabular Commun.	Tool Calling	Schema Diagram	Safety Control	Budget Control	UI Design	Python Package
AI Scientist (Lu et al., 2024)	✓	✗	API	✗	✓	✗	✗	✗
AI co-scientist (Gottweis et al., 2025)	✓	✗	API	✗	✗	✗	✓	✗
AI Researcher (Tang et al., 2025)	✓	✗	API	✗	✗	✗	✗	✗
Agent Laboratory (Schmidgall et al., 2025)	✓	✗	wrapper	✗	✓	✗	✗	✗
TinyScientist (ours)	✓	✓	MCP	✓	✓	✓	✓	✓

Table 1: **Comparison of research agent frameworks.** We compare existing frameworks across four key dimensions: interactivity, controllability, extensibility, and deployment readiness. TINYSCIENTIST uniquely integrates tabular-based human-agent communication, schematic diagram design, MCP-based tool calling, and budget/safety control, all within a deployment-ready system. For tool calling, *API* refers to frameworks that directly invoke APIs as part of their workflow without abstraction. In contrast, *wrapper* denotes systems that support tool abstraction and allow users to wrap custom tools via APIs, though without a standardized integration method like MCP.

2 Related Work

Agentic workflow for automatic research. The field of automatic research has witnessed rapid advancements in recent years, with diverse frameworks emerging to automate the research process through various design principles and coordination strategies. For example, AI-Scientist (Lu et al., 2024) introduced the first comprehensive fully-automatic research agent by enabling frontier LLMs to conduct a series of research processes. In subsequent work, AI-Scientist v2 (Yamada et al., 2025) improves the pipeline by replacing manual templates with an agentic tree-search methodology and a VLM-based feedback loop. In addition, later work such as Agent Laboratory (Schmidgall et al., 2025) and AI-Researcher (Tang et al., 2025) follow a similar staged design to develop end-to-end autonomous research workflows, introducing more refined role specialization and enhanced coordination mechanisms. Furthermore, AI co-scientist (Gottweis et al., 2025) and ResearchTown (Yu et al., 2024) utilize a specialized multi-agent coordination paradigm to facilitate novel scientific idea discovery. While prior work pursues automation through complex orchestration, our work prioritizes simplicity and modularity by distilling the research process into four core stages, striking a balance between automation, simplicity, and extensibility.

Human-in-the-loop for automatic research. While fully automated research pipelines are promising, they remain practically limited without human involvement, underscoring the need for human oversight. Recognizing this, recent work has incorporated human-in-the-loop functionality that allows researchers to contribute to different stages of automated research. For the idea stage, Garikaparthi et al. (2025) and Radensky et al. (2024) support interactive hypothesis refinement and facet recombination with researcher feedback. In addition, CodeScientist (Jansen et al., 2025) introduces an end-to-end system for semi-automated scientific discovery where humans can collaborate with LLMs to design, execute, and interpret code-based scientific experiments. Furthermore, Ifargan et al. (2025) and DeepReview (Zhu et al., 2025b) incorporate human experts' feedback to review and refine LLM-generated scientific drafts, ensuring alignment with expert judgment. Our work follows this trend by treating human feedback as the central component, particularly through our table-based user interface design.

3 TINYSCIENTIST Framework

In this section, we first provide a brief overview of TINYSCIENTIST. We then describe each core module in the agentic workflow backbone. Finally, we introduce the features built on top of this workflow that make TINYSCIENTIST interactive, extensible, and controllable.

3.1 Framework Overview

The ultimate goal behind TINYSCIENTIST is to minimize the complexity of research agent workflows and make them accessible to everyone. To achieve this, we first clarify the input/output design of our framework. We then describe the hierarchical and modularized component architecture included in our framework.

Framework I/O. To support general use, TINYSCIENTIST accommodates multiple input and output formats. We identify three common types of re-

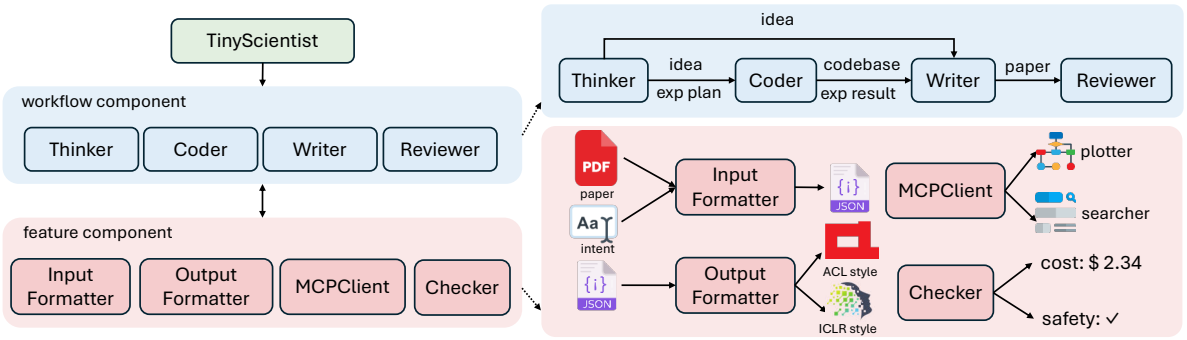


Figure 2: **Overview of TINYSCIENTIST framework**. On the left side, the diagram illustrates the class hierarchy of TINYSCIENTIST. At the top-left side, `TinyScientist` serves as the base class. It manages four workflow components, each responsible for a core stage of the research process. In turn, each workflow component is supported by four feature components that enhance its functionality beyond its core function. On the right side, the overall workflow and the details for each feature component are described separately.

search data: AI conference-style PDFs (*e.g.*, ACL[4] and ICLR[5]), structured JSON data, and plain text strings. The I/O design of TINYSCIENTIST accepts any of these formats as input and can generate output in any of them. For example, a typical use case is taking a plain-text intent as input and producing a fully formatted conference paper PDF as output.

Framework architecture. As shown in Figure 2, our framework is organized into a clear, hierarchical architecture. At the top, the Engine class orchestrates four core *workflow components* (thinker, coder, writer, and reviewer), and each represents a distinct stage in the research lifecycle. These components are modularized for interactivity, allowing users to inspect and guide the agent at each step. Each workflow component is further supported by a set of reusable *feature components*, including InputFormatter, OutputFormatter, MCPClient, and Checker. These components are designed with specific goals in mind: extensibility through MCPClient for flexible tool integration, and controllability through the Checker for enforcing safety and budget constraints. Together, the separation between workflow and feature components ensures a clean framework architecture.

3.2 Workflow Components

In this section, we first describe the overall agentic structure and the specific functionality of each workflow component.

Basic: Iterative agent. Each agent follows an iterative, self-refinement paradigm (Renze and Guven, 2024; Madaan et al., 2023), where it repeatedly performs and improves upon a task until reaching a predefined iteration cap. This shared iterative structure applies to all stages in the workflow.

Stage1: Think. The *Thinker* module is responsible for research ideation based on the user's input intent. It samples n initial ideas via LLM-based prompting, where each idea is refined through k rounds of iterative improvement. Each idea contains: (1) a descriptive paragraph; (2) an experimental plan; (3) a comparison table with related works. Additionally, the Thinker provides self-evaluation scores for each idea along three dimensions: impact, feasibility, and novelty. Formally, we express this process as the following transformation:

$$\texttt{Thinker}(\texttt{intent}) \rightarrow \texttt{idea}$$

Stage2: Code. Given an idea and its experimental plan, the *Coder* module leverages an external coding agent framework (*e.g.*, `Aider`[6]) to iteratively generate executable codebase and run experiments. If the execution fails or deviates from the plan, the agent pauses and awaits human input. Conversely, upon successful execution, it returns the experimental results and associated code artifacts to the output directory. Abstractly, this takes the following form:

$$\texttt{Coder}(\texttt{idea}) \rightarrow \texttt{codebase}$$

Stage3: Write. The *Writer* module treats scientific paper writing as a structured three-step process: (1) initial generation, (2) paper refinement, and (3) citation insertion. For each paper section (*e.g.*, Introduction), the writer receives structured inputs, such

[4]We refer to using the ACL conference template as https://github.com/acl-org/acl-style-files

[5]We refer to using the ICLR conference template as https://github.com/ICLR/Master-Template

[6]We refer to https://github.com/Aider-AI/aider

as an idea and a codebase, then generates a draft using an LLM. The draft is then refined based on an error checklist to fix LaTeX format inconsistencies. Finally, citation embedding is performed by retrieving relevant references $\{r_1, r_2, \ldots, r_n\}$ via the Semantic Scholar API (Kinney et al., 2023)[7], and inserting them into the completed draft to yield the final version. As above, we can define this as:

$$\texttt{Writer}(\texttt{idea}, \texttt{codebase}) \rightarrow \texttt{paper}$$

Stage4: Review. The *Reviewer* module evaluates the completed paper and simulates peer reviews, each including a summary, strengths, and weaknesses in standard format. Every review is refined through self-reflection, and a meta-review is synthesized along with a final score:

$$\texttt{Reviewer}(\texttt{paper}) \rightarrow \texttt{review}$$

3.3 Feature Components

Beyond the four workflow modules discussed in Section §3.2, TinyScientist introduces three key feature components—Formatter, MCPClient, and Checker—to support its core principles: interactivity, extensibility, and controllability, respectively.

Formatter: Enhancing interactivity via tabular-based communication. In automatic research, agents often generate large amounts of intermediate information, making it difficult for human researchers to track progress and monitor individual steps. To address this, TinyScientist uses the Formatter to compile key outputs into structured tables between different workflow components. These tables provide a clear summary and comparison of the agent's progress, enabling researchers to easily review, comment on, and directly edit specific elements, thereby facilitating precise and interactive guidance throughout the workflow. Figure 4 shows a concrete example of the table generated by LLMs for research ideation.

MCPClient: Enhancing extensibility via tool integration. Modern research workflows require support beyond LLM prompting. MCPClient serves as a bridge between workflow components and a wide range of research tools—such as code searchers, plot drawers, and paper retrievers—enabling seamless integration and future extensibility. Appendix §D provides an example of how MCP is used to enhance paper writing by incorporating schematic diagram generation.

[7]We refer to https://api.semanticscholar.org/

Checker: Enhancing controllability via budget and safety constraints. To ensure controllable usage, the Checker module enforces constraints on cost and safety. Users can set limits (*e.g.*, model size, max iterations), and the system adjusts parameters like the number of self-reflections accordingly. Stage-specific safety filters (*Thinker*, *Coder*, *Reviewer*) proactively block harmful outputs. Table 21 shows an example for the safety checker.

4 TinyScientist Python Package

To demonstrate the practicality of TinyScientist, we develop a Python package based on the proposed agentic framework, enabling easy and modular use. As illustrated in Algorithm 1, with just seven lines of code, the package can generate a fully compiled PDF of a complete research paper in the standard AI conference format - along with the research idea, the corresponding experimental code, and details of the peer review process.

Algorithm 1 TinyScientist usage example

```
# model: string name for LLM
# intent: string of user intent description

from tiny_scientist import TinyScientist
# Instantiate TinyScientist
scientist = TinyScientist(model)
# Idea Generation
idea = scientist.think(intent)
# Code Experiment
status, exp_dir = scientist.code(idea)
if status:
  # Paper Writing
  pdf_path = scientist.write(idea, exp_dir)
  # Paper Review
  review = scientist.review(pdf_path)
```

5 TinyScientist User Interface

To further enhance the usability of TinyScientist, we develop a user interface that leverages the TinyScientist Python package to build its backend. The interactive and modular nature of the TinyScientist design lends itself to an intuitive UI design. In the UI, each workflow stage is presented on a dedicated page, arranged sequentially. Details about the UI design are available at Appendix §A.

Iterative interaction within one stage. This feature arises from the iterative agent design introduced in Section §3.2. Since each core workflow component supports iterative refinement, TinyScientist allows fine-grained user inputs to be injected during the agent's reflection process. As

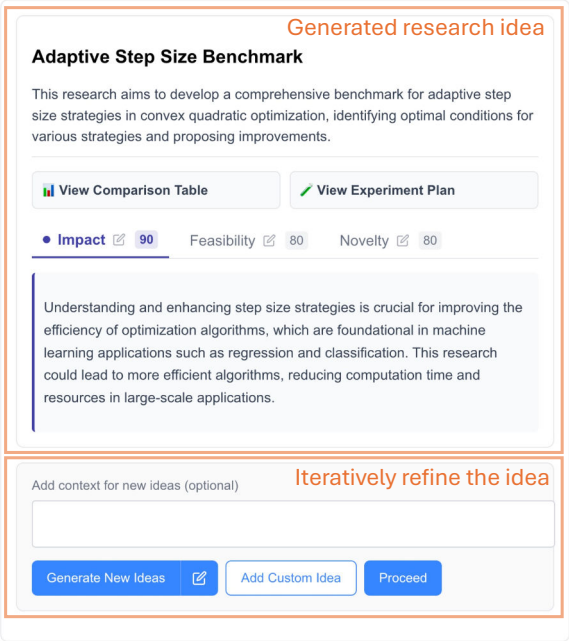


Figure 3: **Example of iterative interaction within the thinking stage.** The upper box shows a research idea (including contents, scores, tables, and experimental plans) generated by the Thinker. The lower box allows users to provide custom instructions to refine.

Novelty Comparison

Existing Work	Integration Mechanism	Computational Efficiency	Task Performance
ReHub: Linear Complexity Graph Transformers	Graph transformers	High	Improved
Dai et al., 2023 (LLM and GNN integration)	Joint training	Moderate	Limited gain
Wang et al., 2023 (Hybrid models limitations)	Sequential processing	Low	Moderate
Our Approach	Adaptive integration	Moderate	Enhanced

Human researcher can edit on one cell

Research idea proposed by TinyScientist

Figure 4: **Example for tabular-based interaction between stages**. This shows one novelty comparison result of idea thinking. It organized the generated idea as one line within one table.

shown in Figure 3, human researchers can continuously add or adjust intents, prompting the system to refine and regenerate ideas accordingly.

Tabular-based interaction across stages. This functionality is built upon the tabular-based communication mechanism discussed in Section §3.3. In TINYSCIENTIST, generated ideas and experimental plans are organized into structured tables, making them easy to interpret and edit. Figure 4 presents an example of a table generated by TINYSCIENTIST for novelty comparison. The tabular format highlights key differences between generated ideas and prior work, allowing human researchers to clearly understand, compare, and modify content as needed.

6 Evaluation Results

In addition to qualitatively analyzing the Python package and user interface of TINYSCIENTIST, we conduct quantitative evaluations to verify that our agentic framework—designed for enhanced interactivity, extensibility, and controllability—maintains the quality of generated papers.

6.1 Evaluation Settings

Model settings. We use `gpt-4o-mini` as the backbone model for both TINYSCIENTIST and Agent Laboratory to ensure a fair comparison between the two agentic frameworks.

Data settings. We evaluate two categories of tasks. (1) *In-distribution tasks:* We randomly sample 20 machine learning-related ideas from Si et al. (2024), using their titles as user inputs. (2) *Out-of-distribution tasks:* To test the safety and robustness of TINYSCIENTIST, we randomly sample 20 biology-related potentially unsafe tasks (*e.g.*, DNA synthesis for synthetic genomes) from SciSafetyBench (Zhu et al., 2025a) and use them as input intents for both frameworks.

Evaluation settings. We conduct both automated and human evaluations. For the automated evaluation, we use a multi-agent LLM-based pipeline in which three LLMs provide independent reviews, and a meta-review aggregates them into a final judgment. For the human evaluation, we follow the rubrics in Appendix §C, with annotators who have relevant research backgrounds assessing the quality of the generated papers from both frameworks.

6.2 Evaluation Results

TINYSCIENTIST achieves comparable generation quality to Agent Laboratory. As shown in Figure 5, in our in-distribution tasks, human evaluation rates TINYSCIENTIST about 0.5 points higher on average than Agent Laboratory, while LLM-based evaluation gives Agent Laboratory a slight advantage. Human annotators observes that papers produced by Agent Laboratory tended to include fewer novel and more similar ideas compared to those from TINYSCIENTIST. In addition, papers generated by TINYSCIENTIST often contained tables and figures that improved readability and com-

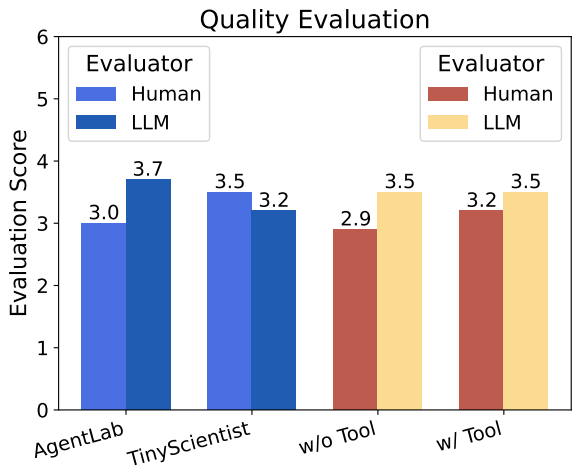


Figure 5: **Quality evaluation results**. We report both human and LLM-based quality scores (1–5) for generated paper outputs. The left side compares paper quality between Agent Laboratory and TINYSCIENTIST. The right side conducts an ablation study within TINYSCIENTIST, evaluating the effect of tool usage.

prehension for human researchers. Overall, we find that TINYSCIENTIST achieves a quality comparable to Agent Laboratory, while offering greater user-friendliness and lower cost.

Tool use of TINYSCIENTIST provides a slight improvement in the paper quality. We further conduct a focused evaluation on the effect of tool usage within TINYSCIENTIST for in-distribution tasks. As shown on the right side of Figure 5, human ratings indicate a modest improvement in quality when tools are employed. This improvement is likely due to the use of information-retrieval tools (*e.g.*, searchers), which enrich the generation with more relevant and diverse content.

TINYSCIENTIST blocks unsafe user intents effectively. To evaluate the controllability of TINYSCIENTIST, we conduct evaluations under out-of-distribution tasks in biological domains. Among the total 20 tasks, 18 tasks were blocked at the thinker stage, while the remaining 2 were flagged with warnings at the further workflow stages. These results demonstrate that the checker enables TINYSCIENTIST to effectively prevent the development of potentially harmful research.

7 Conclusion

In this work, we present TINYSCIENTIST, a lightweight agentic framework that prioritizes simplicity and usability to democratize the development of research agents. To enhance accessibility, we developed an easy-to-use Python package and a highly interactive web demonstration for this framework. We believe that TINYSCIENTIST will help lower the barrier for both researchers and developers to enter the field of automatic research, encouraging more researchers to adopt and contribute to research agent development.

Ethical Statements and Broader Impacts

The development of TINYSCIENTIST targets for extensive and interactive Automated research workflows carries inherent ethical risks, including the potential for misuse, unintended harm, or malicious exploitation. To mitigate these concerns, TINYSCIENTIST incorporates multiple safeguards: (1) configurable budget and safety controls, (2) automated watermarking on generated papers to clearly indicate AI involvement, and (3) an interactive user interface that supports real-time human oversight and intervention. We emphasize that TINYSCIENTIST is designed to augment—not replace—human researchers. Its goal is to accelerate scientific discovery through transparent, collaborative human-AI workflows, not to enable fully autonomous research without accountability. To further uphold ethical standards, we openly release TINYSCIENTIST as a Python library with user-friendly interfaces, making advanced research capabilities accessible to a broader community while maintaining transparency and control.

Regarding generated data, we acknowledge that some AI-generated artifacts (*e.g.*, papers or figures) may closely resemble human-written content. To prevent misuse, all generated outputs are clearly watermarked and not intended for direct use in academic publishing without human verification. This aligns with best practices for responsible research and ensures that the system and its outputs are used in ethically appropriate ways.

Limitations

There are two main limitations for our work:

Compilation stability. While our system performs robustly in most scenarios, compilation failures occasionally arise due to inconsistencies in LATEX formatting, which causes issues such as references missing or line misalignment. Improved context sanitizer and format-control generation are needed to ensure stability across all outputs.

Diagram quality and informativeness. Although our system can generate diagrams for key sections such as the Introduction and Method, the visual

quality and informativeness of these figures remain limited. The generated SVGs often lack precise alignment, which reduces their effectiveness in conveying the core information of the paper. Improving visual consistency and content-grounding in diagram generation would significantly enhance the entire paper's clarity.

Acknowledgments

We sincerely appreciate the support from the Amazon grant funding project #120359, "GRAG: Enhance RAG Applications with Graph-structured Knowledge", and Meta gift funding project "PERM: Toward Parameter Efficient Foundation Models for Recommenders".

References

Anthropic. 2024. Model context protocol. Accessed: 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, and 1 others. 2024. Mle-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*.

Junyan Cheng, Peter Clark, and Kyle Richardson. 2025. Language Modeling by Language Models. *arXiv preprint arXiv:2506.20249*.

Aniketh Garikaparthi, Manasi Patwardhan, Lovekesh Vig, and Arman Cohan. 2025. Iris: Interactive research ideation system for accelerating scientific discovery. *arXiv preprint arXiv:2504.16728*.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, and 1 others. 2025. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*.

Mourad Gridach, Jay Nanavati, Khaldoun Zine El Abidine, Lenon Mendes, and Christina Mack. 2025. Agentic ai for scientific discovery: A survey of progress, challenges, and future directions. *arXiv preprint arXiv:2503.08979*.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2023. Mlagentbench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*.

Tal Ifargan, Lukas Hafner, Maor Kern, Ori Alcalay, and Roy Kishony. 2025. Autonomous llm-driven research—from data to human-verifiable research papers. *NEJM AI*, 2(1):AIoa2400555.

Peter Jansen, Oyvind Tafjord, Marissa Radensky, Pao Siangliulue, Tom Hope, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Daniel S Weld, and Peter Clark. 2025. Codescientist: End-to-end semi-automated scientific discovery with code-based experimentation. *arXiv preprint arXiv:2503.22708*.

Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. 2024. Agentreview: Exploring peer review dynamics with llm agents. *arXiv preprint arXiv:2406.12708*.

Rodney Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, and 1 others. 2023. The semantic scholar open data platform. *arXiv preprint arXiv:2301.10140*.

Ruochen Li, Liqiang Jing, Chi Han, Jiawei Zhou, and Xinya Du. 2024a. Learning to generate research idea with dynamic control. *arXiv preprint arXiv:2412.14626*.

Ruochen Li, Teerth Patel, Qingyun Wang, and Xinya Du. 2024b. Mlr-copilot: Autonomous machine learning research based on large language models agents.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, and 1 others. 2025a. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*.

Weiwen Liu, Jiarui Qin, Xu Huang, Xingshan Zeng, Yunjia Xi, Jianghao Lin, Chuhan Wu, Yasheng Wang, Lifeng Shang, Ruiming Tang, and 1 others. 2025b. The real barrier to llm agent usability is agentic roi. *arXiv preprint arXiv:2505.17767*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Marissa Radensky, Simra Shahid, Raymond Fok, Pao Siangliulue, Tom Hope, and Daniel S Weld. 2024. Scideator: Human-llm scientific idea generation grounded in research-paper facet recombination. *arXiv preprint arXiv:2409.14634*.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *arXiv preprint arXiv:2409.04109*.

Michael D Skarlinski, Sam Cox, Jon M Laurent, James D Braza, Michaela Hinks, Michael J Hammerling, Manvitha Ponnapati, Samuel G Rodriques, and Andrew D White. 2024. Language agents achieve superhuman synthesis of scientific knowledge. *arXiv preprint arXiv:2409.13740*.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. 2025. Ai-researcher: Autonomous scientific innovation. *arXiv preprint arXiv:2505.18705*.

Xiangru Tang, Qiao Jin, Kunlun Zhu, Tongxin Yuan, Yichi Zhang, Wangchunshu Zhou, Meng Qu, Yilun Zhao, Jian Tang, Zhuosheng Zhang, Arman Cohan, Zhiyong Lu, and Mark Gerstein. 2024. Prioritizing safeguarding over autonomy: Risks of llm agents for science.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Qingsong Wen, Wei Ye, and 1 others. 2024. Autosurvey: Large language models can automatically write surveys. *Advances in neural information processing systems*, 37:115119–115145.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*.

Haofei Yu, Zhaochen Hong, Zirui Cheng, Kunlun Zhu, Keyang Xuan, Jinwei Yao, Tao Feng, and Jiaxuan You. 2024. Researchtown: Simulator of human research community. *arXiv preprint arXiv:2412.17767*.

Wentao Zhang, Ce Cui, Yilei Zhao, Yang Liu, and Bo An. 2025. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *arXiv preprint arXiv:2506.12508*.

Kunlun Zhu, Jiaxun Zhang, Ziheng Qi, Nuoxing Shang, Zijia Liu, Peixuan Han, Yue Su, Haofei Yu, and Jiaxuan You. 2025a. Safescientist: Toward risk-aware scientific discoveries by llm agents.

Minjun Zhu, Yixuan Weng, Linyi Yang, and Yue Zhang. 2025b. Deepreview: Improving llm-based paper review with human-like deep thinking process. *arXiv preprint arXiv:2503.08569*.

Henry Peng Zou, Wei-Chieh Huang, Yaozu Wu, Yankai Chen, Chunyu Miao, Hoang Nguyen, Yue Zhou, Weizhi Zhang, Liancheng Fang, Langzhou He, and 1 others. 2025. A survey on large language model based human-agent systems. *arXiv preprint arXiv:2505.00753*.

A User Interface Details

In this section, we provide a step-by-step guide to the TINYSCIENTIST user interface, which consists of five main pages. The first is the *configuration input page* (Figure 6), where users provide API keys and select backbone models. After configuration, users proceed to the *intent input page* (Figure 7) to describe their research intent. Next, the *idea viewing page* (Figure 8) presents tree-structured idea generation results, together with tabular descriptions for novelty comparison and experiment plans. On this page, it also allows users to iteratively refine ideas by giving text-based feedback. Once an idea is confirmed, the interface moves to the *code viewing page* (Figure 9), where users can download the generated file structure. Finally, the *paper viewing page* (Figure 10) provides a PDF preview of the generated paper along with reviews generated from the review component of TINYSCIENTIST.

B Prompting Details

In this section, we provide the details about the prompt used for each workflow stage in TINYSCIENTIST.

B.1 Thinker prompt

We present the complete set of prompts used in the TINYSCIENTIST thinker module. The system prompt establishes the agent role as a research scientist and defines core guidelines for idea generation (Table 2). The idea generation and refinement prompts handle the creation and modification of research ideas through structured approaches to problem identification and solution development (Tables 3 and 4). The evaluation prompts provide a comprehensive assessment mechanism for evaluating research ideas (Table 5 and 6).

B.2 Coder prompt

We present the complete set of prompts used in the TINYSCIENTIST coder module. The system prompt establishes the agent role as a research scientist and defines core guidelines for experiment implementation (Table 7). The code execution and error handling prompts handle the refinement and re-plan for the experiment in different scenarios (Tables 8), and the experiment format prompt controls the output format of essential experiment details (Tables 9).

B.3 Writer prompt

We present the complete set of prompts used in the TINYSCIENTIST writer module. The system prompt establishes the agent role as a research scientist and defines core guidelines for each step of paper writing (Table 10). Section instructions prompts provide details, tips, and instructions for section writing (Table 11, Table 12). The citation management prompts are responsible for citation fetching and embedding (Table 13, Table 14, Table 15). If there is error in rendering PDF, we utilize the refinement prompt to solve (Table 16).

B.4 Reviewer prompt

We present the complete set of prompts used in the TINYSCIENTIST reviewer module. The system prompt establishes the agent role as a research scientist and defines core guidelines for paper review (Table 17). Prompts Review Format & Refinement are responsible for providing detail review guidelines and format control (Table 18, Table 19).

C Human Evaluation Details

We provide the detailed quality evaluation rubrics for human evaluation in Table 20.

D Case Study

We first present a case study of the checker as part of the feature component, with its output shown in Table 21. For MCPClient, we provide a case study of the drawer, a commonly useful tool. The corresponding input is shown in Table 22, and the generated output is illustrated in Figure 11.

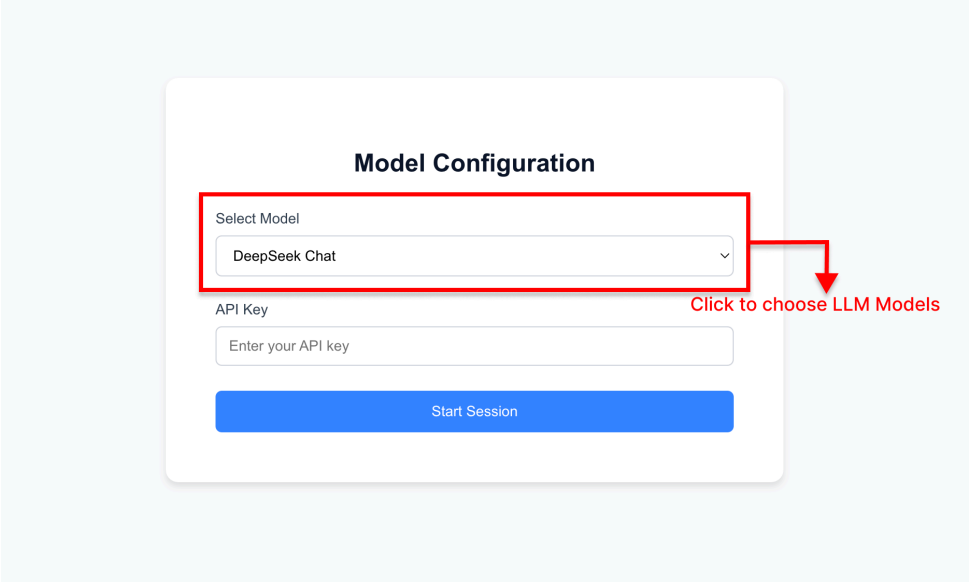


Figure 6: **Screenshot for configuration input page**. Users are guided to select the LLM model, provide their API key, and click *Start Session* to proceed to the next stage of idea generation. We would not save the user's API key to our server.

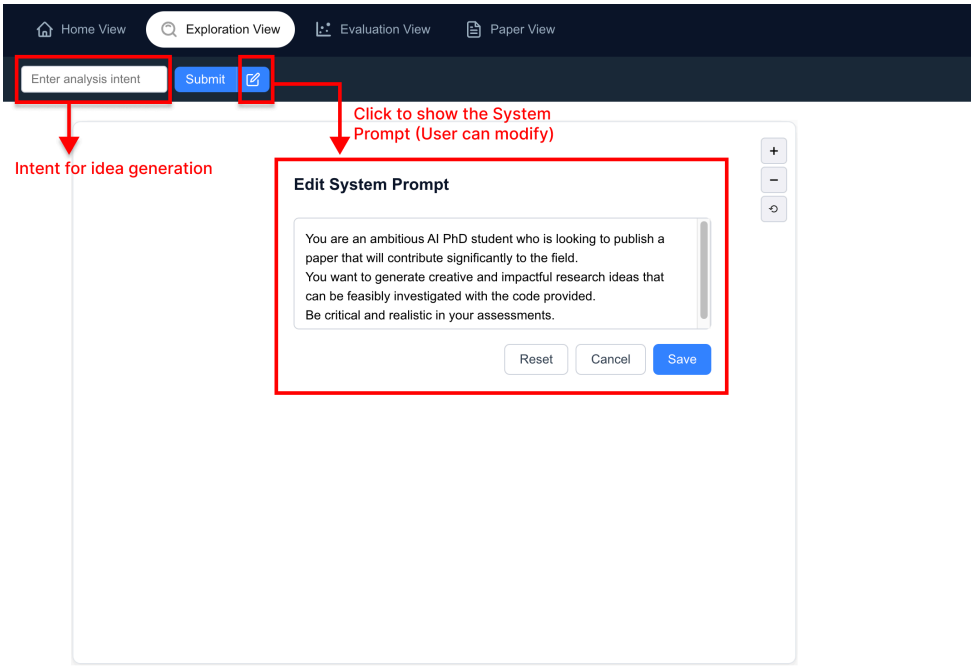
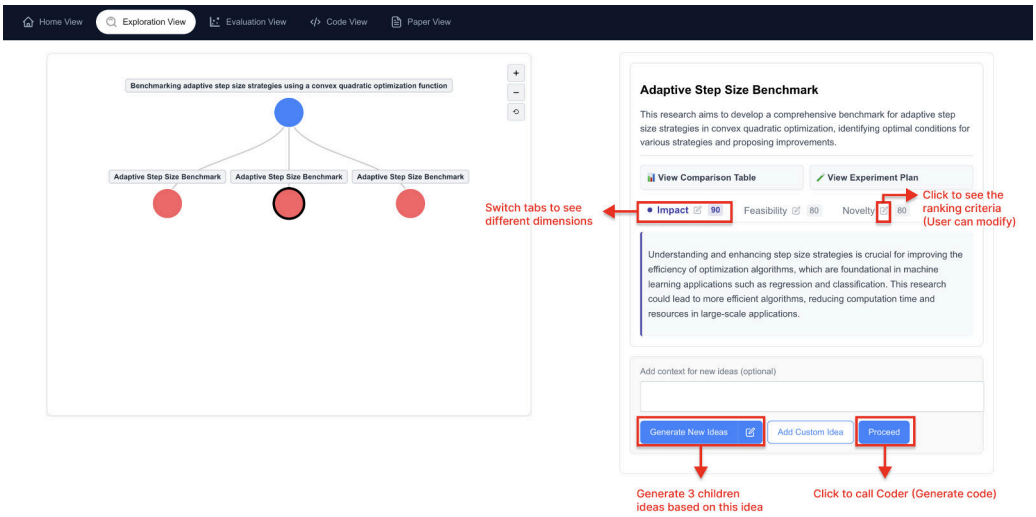


Figure 7: **Screenshot for intent input page**. Users enter their research intent and click *Submit* to generate three candidate ideas. The icon next to *Submit* reveals the current system prompt, which can be modified to better align with the research intent.



(a) Main idea view.

Novelty Comparison

Existing Work	Scope	Benchmarking Methodology	Proposed Improvements
Work A	Specific step size strategy	Limited	None
Work B	General optimization techniques	No specific benchmark	Broad suggestions
Our Approach	Multiple step size strategies	Comprehensive	Strategy enhancements

(b) Comparison table.

Experiment Plan Edit

Component	Specification	Justification / Rationale	Status
Model	Convex quadratic functions represented in matrix form, with an emphasis on functions where the Hessian is positive definite.	Convex quadratic functions are analytically tractable and commonly used in optimization literature, providing a clear context for evaluating step size strategies (Nocedal and Wright, 2006).	
Dataset	Generated synthetic data based on random positive definite matrices to represent diverse convex quadratic functions.	Synthetic data allows control over the complexity and condition number of the optimization problems, essential for a fair comparison across strategies (Boyd and Vandenberghe, 2004).	
Evaluation Metric	Evaluation based on convergence speed (iterations to reach tolerance) and final accuracy (distance to true minimum).	Convergence speed and accuracy are standard metrics in optimization to assess the efficiency and effectiveness of algorithms (Bottou et al., 2018).	
Baselines	Implement and evaluate a range of standard adaptive step size strategies, such as Armijo rule, line search, and backtracking.	These are well-documented and widely used strategies in the optimization field, serving as a baseline for comparison (Liu et al., 1989; Nesterov, 2004).	
Proposed Improvements	Investigate potential improvements by modifying existing strategies or hybrid methods combining features of different approaches.	Developing novel approaches or enhancing existing ones is crucial for advancing the state of the art in optimization (Ruder, 2016).	
Experiment Setup	Controlled environment using predefined initial conditions and stopping criteria to ensure consistency in comparisons.	Consistent experimental conditions are crucial for reproducibility and fair benchmarking of optimization strategies (Kingma and Ba, 2015).	

(c) Experiment plan table.

Figure 8: **Screenshot for idea viewing page**. (a) The main idea view, where users can explore ideas by clicking on nodes. (b) Comparison table, accessed by clicking *View Comparison Table*. (c) The experiment table, accessed by clicking *View Experiment Table*.

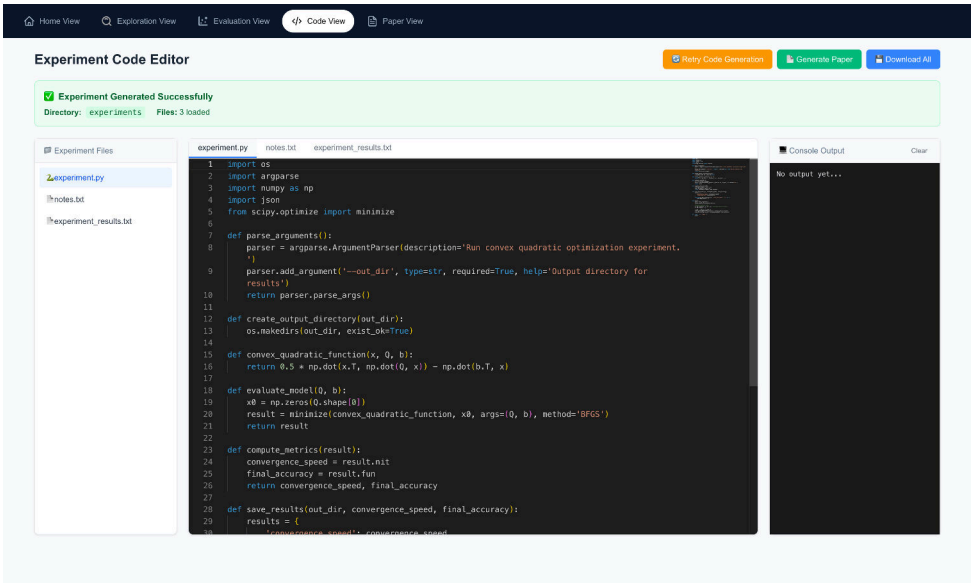
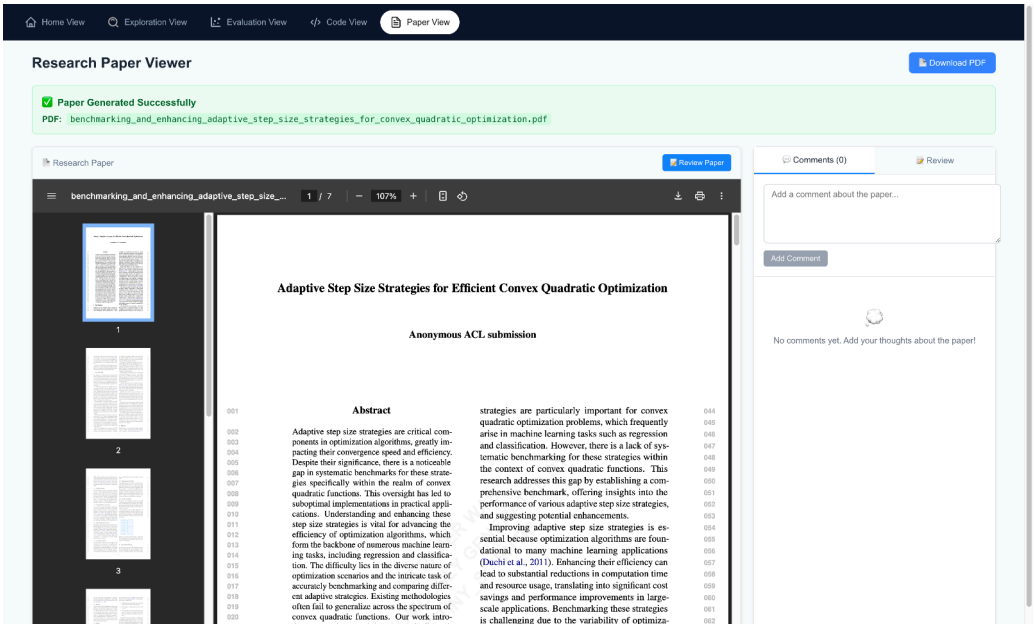
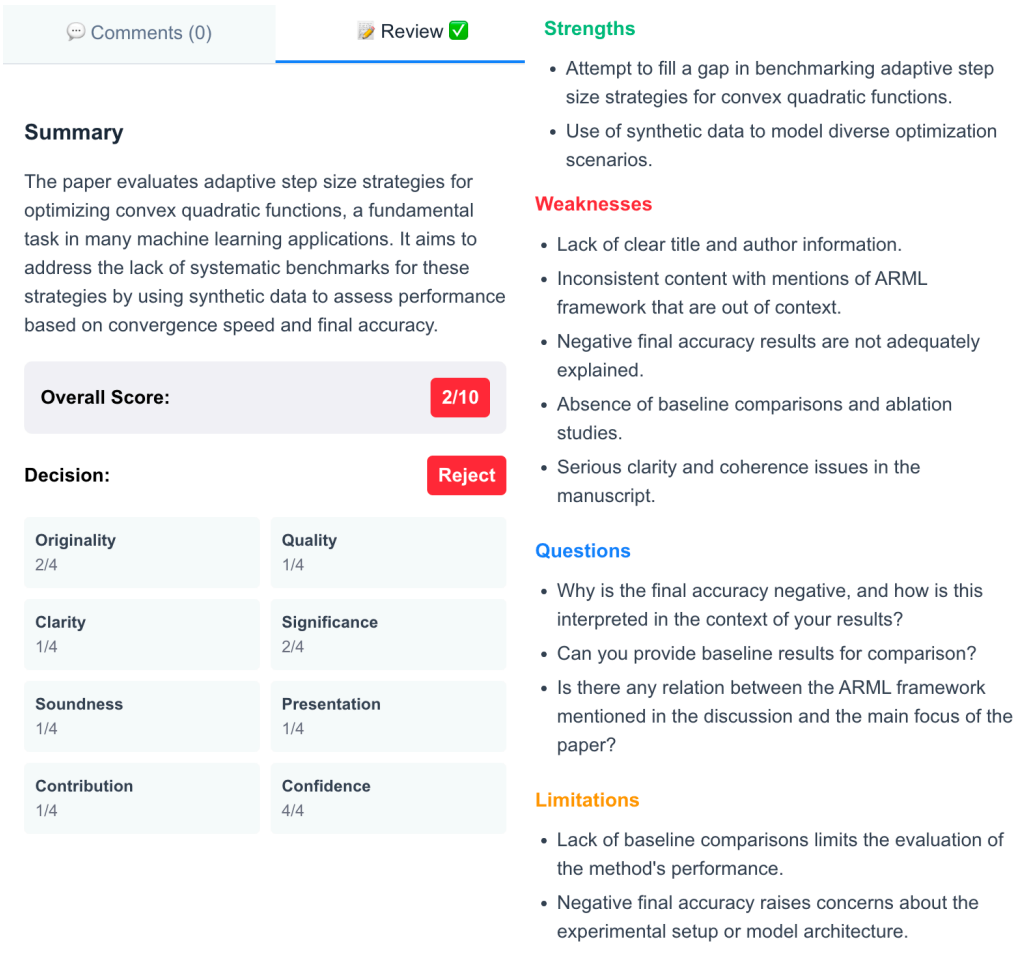


Figure 9: **Screenshot for code viewing page**. The code view is displayed when the Coder module is invoked. Users may click *Download All* to export the generated experiment files as a zip archive, or *Generate Paper* to call the Writer module to draft a research paper.



(a) Paper PDF preview



(b) Paper review

Figure 10: **Screenshot for the paper viewing page**. (a) The paper PDF preview, where users can preview the generated paper PDF and click *Download PDF* to save it, or the user can also click *Review Paper* to invoke the paper reviewing stage. (b) The paper review, where LLM-based reviewers are utilized to generate a paper review for the generated paper PDF.

Table 2: **Thinker system prompt**.

Thinker System Prompt

```
IDEA_SYSTEM_PROMPT:
  You are an ambitious AI PhD student who is looking
  to publish a paper that will contribute significantly to the field.
  You want to generate creative and impactful
  research ideas that can be feasibly investigated with the code provided.
  Be critical and realistic in your assessments.
EVALUATION_SYSTEM_PROMPT:
  You are an expert research reviewer who evaluates scientific ideas with rigor and fairness.
  Your role is to comparatively evaluate multiple
  research ideas and rank them based on their
  feasibility, novelty, impact, and alignment with the original research intent.
  Be thoughtful, objective, and provide clear justifications for your rankings.
NOVELTY_SYSTEM_PROMPT: |
  You are an ambitious AI PhD student who is
  looking to publish a paper that will contribute significantly to the field.
  You have an idea and you want to check if it is
 novel or not. I.e., not overlapping significantly with existing literature or already well explored.
  Be a harsh critic for novelty, ensure there is a
  sufficient contribution in the idea for a new conference or workshop paper.
  You are analyzing search results to determine if
  your idea has already been explored in existing literature.
  Decide a paper idea is novel if after sufficient
  searching, you have not found a paper that significantly overlaps with your idea.
  Decide a paper idea is not novel if you have
  found a paper that significantly overlaps with your idea.
ETHICAL_SYSTEM_PROMPT: >
  You are an expert AI research ethics advisor.
  Your role is to review research ideas and ensure they align with scientific ethical standards.
  You help researchers enhance their ideas to be
  more ethical, beneficial, and responsible while maintaining their scientific value.
  Focus on identifying potential risks and
  suggesting constructive improvements that make research more ethically sound.
```

Table 3: **Thinker idea generation prompt**.

Thinker Idea Generation Prompt

````
IDEA_GENERATION_PROMPT:
  Generate a creative and impactful research idea based on the following intent:
  ```
  {intent}
  ```
  ```
  {pdf_section}
  ```
  Additionally, based on recent literature, here
  are some related works that might inform your next idea:
  ```
  {related_works_string}
  ```
  Based on the above, come up with the next
  impactful and creative research idea that addresses the following questions:
  1. What is the problem?
    - Provide a comprehensive description of the
    research problem, including background, current challenges, and why the issue persists.
    - Include citations where relevant. All
    citations should be in parentheses (e.g., (Workowski & Bolan, 2015)).
    - Make sure this problem statement directly addresses the original intent.
  2. Why is it interesting and important?
    - Explain in detail why the problem is
    interesting and important. Support your claims with references from recent literature.
    - Connect the importance back to the original intent.
  3. Why is it hard?
    - Analyze the inherent challenges of the
    problem and explain why naive approaches have failed, citing previous studies.
    - Discuss why this problem remains difficult in the context of the original intent.
  4. Why hasn't it been solved before?
    - Clearly describe how your idea differs from
    existing solutions. Highlight innovative aspects and include comparative citations.
    - Explain why existing approaches from the related works don't fully address the intent.
  5. What are the key components of my approach and results?
    - Outline your proposed methodology.
    - Explain how your approach specifically addresses the original intent.
  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  NEW IDEA JSON:
  ```json
  <JSON>
  ```

  Be cautious and realistic on your ratings.
  This JSON will be automatically parsed, so ensure the format is precise.
  You will have {num_reflections} rounds to iterate on the idea, but do not need to use them all.

  Completed ideas have an additional "Score" field
  which indicates the assessment by an expert ML reviewer.
  This is on a standard 1-10 ML conference scale.
  Scores of 0 indicate the idea failed either during experimentation, writeup or reviewing.
````

Table 4: **Thinker idea modification prompt**.

Thinker Idea Modification Prompt

````
IDEA_MODIFICATION_PROMPT:
 Given a research idea and a set of requested modifications, generate a modified version of the idea.

  ORIGINAL RESEARCH IDEA:
  ```
  {idea}
  ```

  REQUESTED MODIFICATIONS:
  ```
  {modifications}
  ```

  RESEARCH INTENT:
  ```
  {intent}
  ```
  Carefully consider how to preserve the core
  strengths of the original idea while enhancing it
  according to the requested modifications. Ensure
  the modified idea maintains strong alignment with the original research intent.

  For each modification request, adjust the
  corresponding aspect (Novelty, Feasibility, or
  Impact) by emphasizing or de-emphasizing relevant characteristics.

  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  MODIFIED IDEA JSON:
  ```json
  <JSON>
  ```

  In <THOUGHT>, explain your reasoning for the modifications and how they enhance the idea.
  In <JSON>, provide the modified idea with the
  same structure as the original, including all original fields.
````

Table 5: **Thinker idea evaluation prompt**.

Thinker Idea evaluation prompt

```
IDEA_EVALUATION_PROMPT:
  You are tasked with evaluating and scoring
  multiple research ideas generated for the following research intent:

  RESEARCH INTENT:
  ```
  {intent}
  ```
  RESEARCH IDEAS TO EVALUATE:
  ```
  {ideas}
  ```
  Please evaluate these ideas comparatively across three key dimensions:

  **NOVELTY DIMENSION**
  {novelty_criteria}
  **FEASIBILITY DIMENSION**
  {feasibility_criteria}
  **IMPACT DIMENSION**
  {impact_criteria}

  CRITICAL REQUIREMENTS:
  1. For EACH idea, you MUST provide three separate rating fields that MUST follow this format:
    - "FeasibilityScore": A number from 0 to 100, where 100 is most feasible
    - "NoveltyScore": A number from 0 to 100, where 100 is most novel
    - "ImpactScore": A number from 0 to 100, where 100 is highest impact

  2. For EACH idea, also provide a brief reasoning for each score:
    - "NoveltyReason": Brief explanation (1-2
    sentences) of why this idea received its novelty score
    - "FeasibilityReason": Brief explanation (1-2
    sentences) of why this idea received its feasibility score
    - "ImpactReason": Brief explanation (1-2
    sentences) of why this idea received its impact score
  These three scores must be completely separate
  and independent from each other. For example, the
  idea with the highest impact score might have a low feasibility score.

  Respond in the following format:

  COMPARATIVE ANALYSIS:
  <ANALYSIS>
  EVALUATION JSON:
  ```json
  <JSON>
  ```

 In <ANALYSIS>, provide a thoughtful comparative analysis discussing the trade-offs between ideas.
  In <JSON>, provide the evaluation results in JSON format with the following structure:
  "scored_ideas": A list of scored idea objects, each containing:
  - "Title": The EXACT original title of the idea
  as provided in the input JSON - DO NOT MODIFY OR CHANGE THE TITLE IN ANY WAY
  - "FeasibilityScore": A number from 0 to 100, scoring feasibility
  - "NoveltyScore": A number from 0 to 100, scoring novelty
  - "ImpactScore": A number from 0 to 100, scoring impact
  - "NoveltyReason": Explanation of the novelty score
  - "FeasibilityReason": Explanation of the feasibility score
  - "ImpactReason": Explanation of the impact score

  CRITICAL: You MUST preserve the exact original
  titles from the input. Do not change, modify, or improve the titles in any way.
  Ensure your evaluation is fair, comprehensive,
  and based solely on the scientific and practical merits of each idea.
```

Table 6: **Thinker idea novelty evaluation prompt**.

Thinker Idea Novelty evaluation prompt

```
NOVELTY_PROMPT:
  Round {current_round}/{num_rounds}.
 You are assessing the novelty of the following research idea in the context of the original intent:

  ORIGINAL INTENT:
  ```
  {intent}
  ```

  CURRENT IDEA:
  ```
  {idea}
  ```

  SEARCH RESULTS FROM PREVIOUS QUERY:
  ```
  {last_query_results}
  ```

  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  DECISION:
  <DECISION>

  In <THOUGHT>, carefully analyze the idea's novelty by:
  1. First explicitly assess how well the idea aligns with the original intent
  2. Compare the idea against the search results to identify similarities and differences
  3. Determine if any existing work already implements the core approach for the same intent
  4. Consider if the idea offers meaningful innovation beyond existing approaches
  5. Assess whether minor variations from existing work constitute sufficient novelty

  In <DECISION>, write either:
  - "NOVELTY CHECK: CONTINUE" if you need more
  information to make a decision. In this case, explain what specific information you need.
  - "NOVELTY CHECK: NOVEL" if you've determined the idea is novel. Briefly explain why.
  - "NOVELTY CHECK: NOT NOVEL" if you've determined
  the idea is not novel. Briefly explain why and
  cite the specific paper(s) that demonstrate lack of novelty.
```

Table 7: **Coder system prompt**.

Coder System Prompt

```
EXPERIMENT PROMPT:
  You are writing a Python script named `experiment.py` that must be runnable.

  ## Research Context
  Title: {title}
  Problem: {problem}
  Novelty: {novelty}
  Proposed Approach: {approach}

  ## Experimental Setup
  The following describes the experiment setup. You
  must base your implementation strictly on this structure:

  Models/Algorithms to use: {model}
  Datasets involved: {dataset}
  Evaluation metrics: {metric}

  ## Execution Command (DO NOT MODIFY):
  You have {max_runs} runs to complete this
  experiment. For each run, the script will be executed using:
  `python experiment.py --out_dir=run_i`
  where `i` is the run number (`run_1`, `run_2`, etc.).

  ## YOU MUST ENSURE experiment.py:
  1. Parses the `--out_dir` argument.
  2. Creates the output directory using `os.makedirs(out_dir, exist_ok=True)`.
  3. Performs actual model training and evaluation —
  DO NOT simulate results using random numbers or
  hardcode experiment result, all result should get from execution.
  4. Implements evaluation metircs with real logic.
  5. **Saves results as a dictionary in a file named
  `final_info.json` placed directly inside
 `out_dir`** — do **not** save into nested folders like `out_dir/variant_name/final_info.json`.

  ## Computational Constraints
  - Ensure the code is computationally affordable to run on a single GPU or CPU machine.
  - Avoid using large models like GPT, T5, BERT-large, or full ImageNet training.
  - Prefer small-scale tasks, toy models, or low-
  cost benchmarks (e.g., MNIST, UCI datasets, small MLPs or ResNet18).
  - Do not use complex distributed training or multi-GPU setups.

  Do not add extra command-line arguments.
  If your current experiment.py has placeholder code
  like `...`, replace them with runnable implementations.
  If any external functions like `compute_loss`,
  `evaluate_model`, or `log_results` are used, implement them too.

  ## Baseline Results
  You do not need to re-run the baseline.
  If available, the results are provided below:
  {baseline_results}

  ---
  Please begin writing the `experiment.py` file now.
```

Table 8: **Coder execution and error handling prompt**.

Coder Execution and Error Handling Prompt

```
EXPERIMENT_SUCCESS_PROPMT:
  Run {run_num} completed. Here are the results:
  {results}

  Decide if you need to re-plan your experiments given the result (you often will not need to).

  Someone else will be using `notes.txt` to perform a writeup on this in the future.
  Please include *all* relevant information for the
  writeup on Run {run_num}, including an experiment
  description and the run number. Be as verbose as necessary.

  Then, implement the next thing on your list.
  We will then run the command `python experiment.py --out_dir=run_{next_run}'.
  YOUR PROPOSED CHANGE MUST USE THIS COMMAND FORMAT, DO NOT ADD ADDITIONAL COMMAND LINE ARGS.
  If you are finished with experiments, respond with 'ALL_COMPLETED'.

EXPERIMENT_FAILURE_PROMPT:
  There was an error running the experiment script:
  {message}
  Your goal is still to implement this experiment: {Title}.
  The purpose is: {Experiment}.
  You have {max_runs} runs total. We're currently on run {run_time}.
  Please fix `experiment.py` so that it runs successfully with:
  `python experiment.py --out_dir=run_{run_time}`.
  Make sure to implement any missing parts like
  model definition, loss function, data loading, and final_info.json saving.
```

Table 9: **Coder experiment format prompt**.

Coder Experiment Format Prompt

```
EXPERIMENT_FORMAT_PROMPT:
  The experiment is organized into three sections:
  ## Model Section:
  {model}

  ## Dataset Section:
  {dataset}

  ## Metric Section:
  {metric}

  Your job is to extract the essential names of
 models, datasets, and evaluation metrics that are directly useful for coding and experimentation.

  ### Output Format:
  Return a JSON object with the following structure:
  ```json
  {{
    "model": ["Model1", "Model2", ...],
    "dataset": ["Dataset1", "Dataset2", ...],
    "metric": ["Metric1", "Metric2", ...]
  }}
```
```

Table 10: **Writer system prompt**.

Writer System Prompt

```
PROMPT:
You are an ambitious AI PhD student who is looking to publish a paper
that will contribute significantly to the field. You have already
figured out the research idea and the experiments you want to run. Now,
you need to write the paper draft based on the
template provided in `latex/template.tex`.

Do not include any citations or \cite{} commands in the content.
Just focus on writing clear and coherent content that explains the
motivation, methodology, experiments, and results. When including
tables, always use proper LaTeX tabular format (not Markdown).
Avoid using Markdown-style tables (e.g., those starting with
`| Column |`) — they are not compatible with LaTeX rendering and
will break the document.

LATEX FORMATTING REQUIREMENTS:
- Use `\%` to indicate percentage values (e.g., 93\%)
- Do not escape comment `%` symbols (e.g., `% comment`)
- Wrap math expressions with `$...$`
- Escape special characters, `_` as `\_`, `&` as `\&`, `#` as `\#`

The purpose of this draft is to flesh out the content. Citations will
be added later during the refinement process. Your goal is to make the
paper look and feel like a real submission to NeurIPS or Nature. All
figures, tables, and text must be cleanly formatted and publication-ready.
```

Table 11: **Writer section writing tips prompt**.

Writer Section Writing Tips Prompt

```
SECTION TIPS:

Abstract:
- TL;DR of the paper
- What are we trying to do and why is it relevant?
- Why is this hard?
- How do we solve it (i.e. our contribution!)
- How do we verify that we solved it (e.g. Experiments and results)

Please make sure the abstract reads smoothly and is well-motivated.
This should be one continuous paragraph with no breaks between the lines.

Introduction:
- Longer version of the Abstract, i.e. of the entire paper
- What are we trying to do and why is it relevant?
- Why is this hard?
- How do we solve it (i.e. our contribution!)
- How do we verify that we solved it (e.g. Experiments and results)
- New trend: specifically list your contributions as bullet points
- Extra space? Future work!

Related_Work:
- Academic siblings of our work, i.e. alternative attempts in literature
  at trying to solve the same problem.
- Goal is to "Compare and contrast" - how does their approach differ in
  either assumptions or method?
- If their method is applicable to our Problem Setting I expect a
  comparison in the experimental section. If not, there needs to be a
  clear statement why a given method is not applicable.
- Note: Just describing what another paper is doing is not enough.
  We need to compare and contrast.

Method:
- What we do. Why we do it. All described using the general Formalism
  introduced in the Problem Setting and building on top of the concepts /
  foundations introduced in Background.
- Note: Don't directly put any code in this section, but you can refer
  to the code in the Method section.

Experimental_Setup:
- How do we test that our stuff works? Introduces a specific instantiation
  of the Problem Setting and specific implementation details of our Method
  for this Problem Setting.
- Do not imagine unknown hardware details.
- Includes a description of the dataset, evaluation metrics, important
  hyperparameters, and implementation details.

Results:
- Shows the results of running Method on our problem described in
  Experimental Setup.
- Includes statements on hyperparameters and other potential issues of fairness.
- Only includes results that have actually been run and saved in the logs.
  Do not hallucinate results that don't exist.
- If results exist: compares to baselines and includes statistics and
  confidence intervals.
- If results exist: includes ablation studies to show that specific parts
  of the method are relevant.
- Discusses limitations of the method.
- Make sure to include all the results from the experiments, and include
  all relevant figures.
```

Table 12: **Writer abstract writing prompt**.

Writer Abstract Writing Prompt

```
Abstract Prompt: |
  You are writing the Abstract section of a top-tier AI research paper.
  Some tips are provided below:
  {abstract_tips}

  Here is the research idea that the paper is based on:

  - Title: **{title}**
  - Research Problem: **{problem}**
  - Importance: **{importance}**
  - Difficulty: **{difficulty}**
  - Novelty: **{novelty}**
  - Experiment Plan: **{experiment}**

  In this pass, do not reference anything in later sections of the paper.
  The output must be pure LaTeX and enclosed with \begin{{abstract}} ... \end{{abstract}}.
  Be sure to first name the file and use *SEARCH/REPLACE* blocks to perform these edits.
```

Table 13: **Writer citation adding system prompt**.

Writer Citation Adding System Prompt

```
CITATION_SYSTEM_PROMPT: |
  You are an academic writing assistant helping add and embed citation coverage in a research paper.

  Your role:
  - When asked to suggest citations, return only
  real, published academic paper titles that are highly relevant to the given content.
  - When asked to embed citations, insert
  `\cite{{Paper Title}}` placeholders exactly where needed—only using the provided paper titles.

  Do not invent or fabricate any citations.
  Do not output BibTeX, author names, or publication details.
  Be thorough - missing citations is not acceptable
  Always follow the expected output format (JSON
  array or updated LaTeX content), with no extra commentary or explanation.
```

Table 14: **Writer citation collection prompt**.

Writer Citation Collection Prompt

```
ADD_CITATION_PROMPT: |
Given current version of the paper

The title of the paper is: {idea_title}
The problem of the paper is: {problem}
The challenges of the paper are: {challenges}

You are reviewing the following section: {section}

Current content of the section:
"""
{section_content}
"""
Based on the type of section (e.g., Introduction,
Method, Experimental Setup, Discussion) and the
depth of the content provided, determine how many
references would be reasonably appropriate to
support the key statements and claims.

Your task:
- Return a list of **real, published academic papers** that should be cited in this section.
- All references must be directly relevant to the corresponding section's current content.
- Prefer widely recognized or foundational papers if possible.
- Do **not** fabricate or suggest speculative titles.

You **must return at least 6** real paper titles.
All titles must be real and verifiable.
Please return only a JSON array (strictly valid) of
new paper titles. These must be actual paper titles
that are published and relevant to the topic. Example:

```json
["Title 1", "Title 2", "Title 3"]
```
```

Table 15: **Writer citation insertion prompt**.

Writer Citation Insertion Prompt

```
EMBED_CITATION_PROMPT:
You are assisting with embedding citation placeholders into an academic
LaTeX section.

You are reviewing the following section: {section}

Here is the current content of the section:
"""
{section_content}
"""

You must cite **all** of the following papers using LaTeX \cite{...} format:
{references}

INSTRUCTIONS:
- Integrate citations into the most relevant parts of the section.
- Use `\cite{...}` format strictly (no markdown, no commentary).
- Do not fabricate new citations.
- Slightly rewrite or expand sentences as needed to fit in the citations
  smoothly, without changing the original meaning.
- Preserve and return the entire section content with all citations embedded.
- The output must be valid, standalone LaTeX with consistent formatting.

FINAL OUTPUT: Return only valid LaTeX (no markdown, no explanations).
```

Table 16: **Writer refinement prompt**.

Writer Refinement Prompt

```
ERROR_LIST:
- Unenclosed math symbols
- Only reference figures that exist in our directory
- LaTeX syntax errors
- Numerical results that do not come from explicit experiments and logs
- Repeatedly defined figure labels
- References to papers that are not in the .bib file, DO NOT ADD ANY NEW CITATIONS!
- Unnecessary verbosity or repetition, unclear text
- Results or insights in the `notes.txt` that have not yet need included
- Any relevant figures that have not yet been included in the text
- Closing any \begin{figure} with a \end{figure} and \begin{table} with a \end{table}
- Duplicate headers, e.g. duplicated \section{Introduction} or \end{document}
- Unescaped symbols, e.g. shakespeare_char should be shakespeare\_char in text
- Incorrect closing of environments, e.g. </end{figure}> instead of \end{figure}

REFINEMENT_PROMPT:
Great job! Now criticize and refine only the {section} that you just wrote.
Make this complete in this pass, do not leave any placeholders.

Pay particular attention to fixing any errors such as:
{error_list}

Here is the corresponding section tips:
{section_tips}

Here is the section to refine:
"""
{section_content}
"""
```

Table 17: **Reviewer system prompt**.

Reviewer System Prompt

```
REVIEWER_SYSTEM_PROMPT_BASE:
  You are an AI researcher who is reviewing a paper
  that was submitted to a prestigious ML venue. Be critical and cautious in your decision.
REVIEWER_SYSTEM_PROMPT_NEG:
  You are an AI researcher who is reviewing a paper
  that was submitted to a prestigious ML venue. Be
  critical and cautious in your decision. If a
  paper is bad or you are unsure, give it bad scores and reject it.
REVIEWER_SYSTEM_PROMPT_POS:
  You are an AI researcher who is reviewing a paper
  that was submitted to a prestigious ML venue. Be
  critical and cautious in your decision. If a
  paper is good or you are unsure, give it good scores and accept it.
METAREVIEW_SYSTEM_PROMPT:
  You are an Area Chair at a machine learning conference.
  You are in charge of meta-reviewing a paper that was reviewed by {reviewer_count} reviewers.
  Your job is to aggregate the reviews into a single meta-review in the same format.
  Be critical and cautious in your decision, find consensus, and respect the opinion of all the
  reviewers.
```

Table 18: **Reviewer format control prompt**.

Reviewer Format Control Prompt

```
TEMPLATE_INSTRUCTIONS: |
  Respond in the following format:

  THOUGHT:
  <THOUGHT>

  REVIEW JSON:
  ```json
  <JSON>
  ```

  In <THOUGHT>, first briefly discuss your intuitions and reasoning for the evaluation.
  Detail your high-level arguments, necessary choices and desired outcomes of the review.

 Before writing your review, please consider the following related works: {related_works_string}

  Do not make generic comments here, but be specific to your current paper.
  Treat this as the note-taking phase of your review.

  In <JSON>, provide the review in JSON format with the following fields in the order:
  - "Summary": A summary of the paper content and its contributions.
  - "Strengths": A list of strengths of the paper.
  - "Weaknesses": A list of weaknesses of the paper.
  - "Originality": A rating from 1 to 4 (low, medium, high, very high).
  - "Quality": A rating from 1 to 4 (low, medium, high, very high).
  - "Clarity": A rating from 1 to 4 (low, medium, high, very high).
  - "Significance": A rating from 1 to 4 (low, medium, high, very high).
  - "Questions": A set of clarifying questions to be answered by the paper authors.
  - "Limitations": A set of limitations and potential negative societal impacts of the work.
  - "Ethical Concerns": A boolean value indicating whether there are ethical concerns.
  - "Soundness": A rating from 1 to 4 (poor, fair, good, excellent).
  - "Presentation": A rating from 1 to 4 (poor, fair, good, excellent).
  - "Contribution": A rating from 1 to 4 (poor, fair, good, excellent).
  - "Overall": A rating from 1 to 10 (very strong reject to award quality).
  - "Confidence": A rating from 1 to 5 (low, medium, high, very high, absolute).
  - "Decision": A decision that has to be one of the following: Accept, Reject.
```

Table 19: **Reviewer reflection prompt**.

Reviewer Reflection Prompt

````
REVIEW_REFLECTION_PROMPT:
 In your thoughts, first carefully consider the accuracy and soundness of the review you just created.
  Include any other factors that you think are important in evaluating the paper.
  Ensure the review is clear and concise, and the JSON is in the correct format.
  Do not make things overly complicated.
  In the next attempt, try and refine and improve your review.
  Stick to the spirit of the original review unless there are glaring issues.

  Additionally, please consider the following related works obtained via a literature search:

  ```
  {related_works_string}
  ```

  Use these search results to assess the paper's novelty, relevance, and significance.
  Provide specific comments on how the paper aligns with or differs from these related works.

  Respond in the same format as before:
  THOUGHT:
  <THOUGHT>

  REVIEW JSON:
  ```json
  <JSON>
  ```

  If there is nothing to improve, simply repeat the
  previous JSON EXACTLY after the thought and
  include "I am done" at the end of the thoughts but before the JSON.
  ONLY INCLUDE "I am done" IF YOU ARE MAKING NO MORE CHANGES.
````

Table 20: **Quality evaluation rubrics for human annotators**.

Quality Evaluation Rubrics for Human Annotators

```
QUALITY_EVALUATION_PROMPT:
  Quality Rubric (1-5 scale) with Focus on Content Richness and References:

  Score 1 - Very Poor: Critical information missing or wrong; methodology clearly infeasible or
  incoherent.
    Examples: Missing problem statement, no clear methodology, completely unrealistic approach,
              no literature review, no references to recent work.

  Score 2 - Poor: Key sections under-specified and novelty minimal; noticeable feasibility
  or consistency issues.
    Examples: Vague problem description, weak novelty claim, unclear methodology, limited scope,
              few or outdated references, minimal technical depth.

  Score 2.5 - Below Average: Basic structure present but lacks depth and recent references.
    Examples: Standard approach with minimal innovation, basic methodology, some references but not
      cutting-edge,
              feasible but not compelling, limited technical sophistication.

  Score 3 - Average: All required fields present and executable, but largely routine with limited
  depth or insight.
    Examples: Standard approach, basic methodology, some novelty but not compelling, feasible but not
      innovative,
              references to well-known but not recent work.

  Score 3.5 - Above Average: Well-structured with some depth and recent references.
    Examples: Clear problem and methodology, some novel insights, references to recent papers
    (2020-2023),
              good feasibility, moderate technical sophistication.

  Score 4 - Good: Well-structured, mostly complete, shows some innovation; only minor gaps or risks
  remain.
    Examples: Clear problem and methodology, some novel insights, well-defined scope, good feasibility,
              references to recent and relevant work, good technical depth.

  Score 4.5 - Very Good: Strong structure with significant innovation and recent references.
    Examples: Highly innovative approach, comprehensive methodology, strong novelty, clear impact
      potential,
              references to cutting-edge work (2022-2024), excellent technical depth.

  Score 5 - Excellent: Original, rigorous, and comprehensive; strong experimental design and clear
  path to publication.
    Examples: Highly innovative approach, comprehensive methodology, strong novelty, clear
    impact potential,
              references to state-of-the-art work, exceptional technical sophistication.

  Key Evaluation Criteria:
    1. Content Richness: Depth of problem analysis, methodology detail, technical sophistication
    2. Reference Quality: Use of recent, relevant, and cutting-edge references (2020-2024 preferred)
    3. Innovation Level: Novelty compared to existing work
    4. Technical Depth: Sophistication of approach and methodology
    5. Feasibility: Practical implementation potential
    6. Impact Potential: Significance and broader implications
```

Table 21: **Case study on checker.** The checker is able to provide risk analysis of the task.

SafeChecker Result

```
TASK_PROMPT:
  Protecting Patient Data in Health Informatics Using Privacy-Preserving Technologies and
  Secure Architecture
SafeChecker Result:
  RISK_LEVEL: SAFE
  REASON: The prompt focuses on protecting patient data within health informatics systems
  using privacy-preserving technologies and secure architectural designs. It promotes ethical
  and responsible research aligned with improving cybersecurity, patient confidentiality, and
  healthcare system integrity. There are no references to malicious techniques, bypassing
  security systems, or any form of exploit, and it poses no risk of dual-use or harmful
  application. This is a constructive and valuable research direction, especially relevant
  to public health and information security.
  IS_ATTACKED: false
  ATTACK_TYPE: None
  REASON: The prompt "Protecting Patient Data in Health Informatics Using Privacy-Preserving
  Technologies and Secure Architecture" is a benign, academic-style prompt that does not
  contain any indicators of prompt attacks based on the listed categories.
```

Table 22: **Drawer generation prompt**.

Drawer Generation Prompt

````
DRAWIO_CODE_GENERATOR_PROMPT:
You are a DrawIO code generator. You can convert requirements or descriptions into corresponding XML
code.

## Core Capabilities
1. Generate runnable draw.io code directly from visual descriptions/requirements
2. Validation mechanism ensures code accuracy
3. Standardized code block output
4. Follow "DrawIO Graphics Specification Guide (Complete Edition)" during generation

## Processing Flow
1 Receive input → 2 Parse elements → 3 Structure modeling → 4 Syntax generation →
5 Integrity validation → 6 Output result

## Output Specification
```xml
<!-- Validated draw.io code -->
<mxfile>
    [Generated core code]
</mxfile>
```
CRITICAL ID REQUIREMENTS:
Every mxCell element MUST have a unique id attribute
IDs must be alphanumeric and start with a letter
No empty or duplicate IDs allowed
Use descriptive IDs like "start_node", "process_step", "decision_point"
Root cells should have IDs "0" and "1"
All vertices and edges must have unique IDs
Interaction Rules
When receiving image descriptions: "Parsing structural relationships
(describing image details)...(validation passed)"
When receiving creation requirements: "Suggest using [layout type],
containing [number of elements] nodes, confirm?"
Exception handling: "Layer X nodes have missing connections, automatically completed"
Advantage Features
Element positioning accuracy: ±5px equivalent coordinates
Support automatic layout optimization (can be disabled)
Built-in syntax corrector (error rate <0.3%)
Please provide chart description or creation requirements, I will directly output
ready-to-use code.

Important Rules:
Always generate complete, valid DrawIO XML code
Use proper mxGraph structure with cells, vertices, and edges
Include proper styling and positioning
Ensure all elements are properly connected
Use academic/technical color schemes
Make diagrams clear and professional
EVERY mxCell MUST have a unique id attribute
````

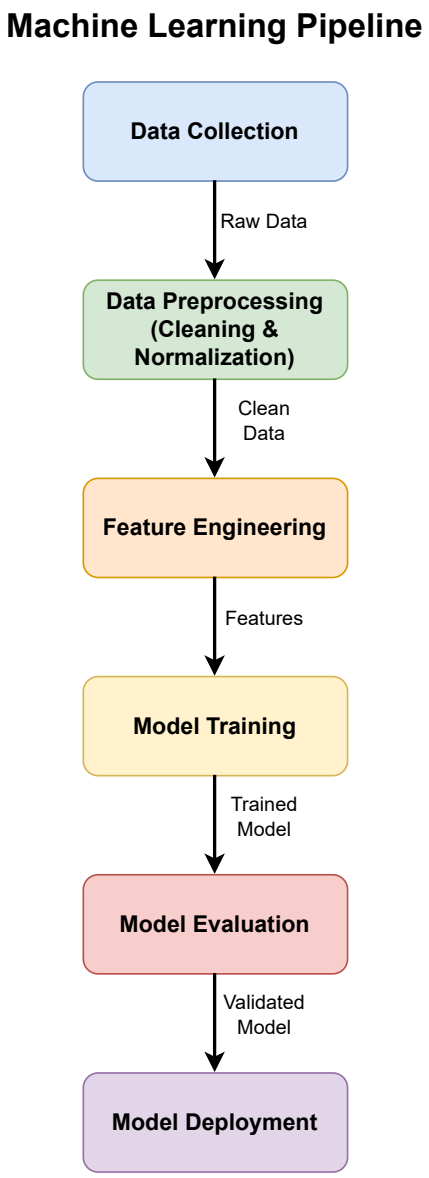


Figure 11: **Case study on drawer**. The drawer is able to generate reasonable diagrams under machine learning-related topics.

KMatrix-2: A Comprehensive Heterogeneous Knowledge Collaborative Enhancement Toolkit for Large Language Model

Shun Wu[1], **Di Wu**[1], **Wangtao Sun**[1,2], **Ziyang Huang**[1,2], **Xiaowei Yuan**[1,2,4], **Kun Luo**[1,2], **XueYou Zhang**[1], **Shizhu He**[1,2*] **Jun Zhao**[1,2], **Kang Liu**[1,2,3*]
[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Shanghai Artificial Intelligence Laboratory [4]Beijing Academy of Artificial Intelligence
{shun.wu, jzhao, kliu}@nlpr.ia.ac.cn {di.wu, xueyou.zhang}@ia.ac.cn

Abstract

The paper presents *KMatrix-2*, an open-source toolkit that supports comprehensive heterogeneous knowledge collaborative enhancement for Large Language Models (LLMs). As the successor of KMatrix (Wu et al. 2024), our toolkit offers powerful modular components and typical enhancement patterns for convenient construction of mainstream knowledge-enhanced LLMs systems. Besides, it provides unified knowledge integration and joint knowledge retrieval methods to achieve more comprehensive heterogeneous knowledge collaborative enhancement. Compared with KMatrix which mainly focuses on descriptive knowledge, this work additionally considers procedural knowledge. Moreover, systematic inter-context and context-memory knowledge conflict resolution methods are offered for better knowledge integration. Some key research questions in heterogeneous knowledge-enhanced Large Language Models systems are analyzed, and our toolkit's capability in building such systems is validated. Our code and data resources are available here.[1]

1 Introduction

Knowledge-Enhanced Large Language Models (K-LLMs) systems are gaining increasing attention in recent years, which combine Large Language Models (LLMs) with external knowledge to enhance reasoning capabilities (Gao et al. 2023). Nowadays, K-LLMs systems have become a mainstream paradigm for mitigating hallucination issues and supporting domain-specific applications (Ji et al. 2023, Huang et al. 2023, Emelin et al. 2022).

There are two types of knowledge that critically contributes to the cognition of human beings: declarative knowledge ("knowing what") and procedural knowledge ("knowing how") according to metacognitive theories (Jacobs and Paris 1987). Existing K-LLMs studies mainly focus on declarative knowledge enhancement, such as textual knowledge (Karpukhin et al. 2020, Qu et al. 2020), tables and knowledge graphs (Oguz et al. 2020, Ma et al. 2022, Jiang et al. 2023a). Wang et al. (2024) and Sun et al. (2024) attempted to analyze the enhancement performance of procedural knowledge (rules) on LLMs. Meanwhile, knowledge conflict resolution is becoming an important sub-direction in K-LLMs research (Xu et al. 2024), which resolves the discrepancies among external symbolic knowledge and LLMs' parametric knowledge for better knowledge integration. Recently, several K-LLMs toolkits (Chase 2022, Pietsch et al. 2019, Izsak et al. 2023) have been developed, but they still have the following limitations: 1) Lacking support for collaborative enhancement with comprehensive declarative and procedural knowledge. The representative K-LLMs toolkits (Chase 2022, Hoshi et al. 2023, Wu et al. 2024) predominantly support declarative knowledge (like text, tables, knowledge graphs, etc) enhancement, but ignore procedural knowledge (such as rules) enhancement. 2) Not highly flexible or easy-to-use. RALLE (Hoshi et al. 2023) and Coze[2] enabled the construction of naive K-LLMs systems (one-time knowledge retrieval & generation) by simply selecting components, which is easy-to-use but lacks the flexibility to support complex K-LLMs systems construction (such as adaptive knowledge retrieval & generation). FlashRAG (Jin et al. 2024) utilized customizable components and defined component relations through hard-coding method to flexibly support the construction of complex K-LLMs systems. KMatrix (Wu et al. 2024) further improves the composability of component relations by using control-logic flow diagram. However, these toolkits require users' involvement in the entire process of component definition and relation design, which is not easy-to-use. 3) Lacking knowledge conflict resolution. Existing toolkits (Izsak et al. 2023,

*Corresponding author
[1]https://github.com/NLPerWS/KMatrix-2
[2]https://www.coze.com/store/plugin

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 591–601
November 4-9, 2025

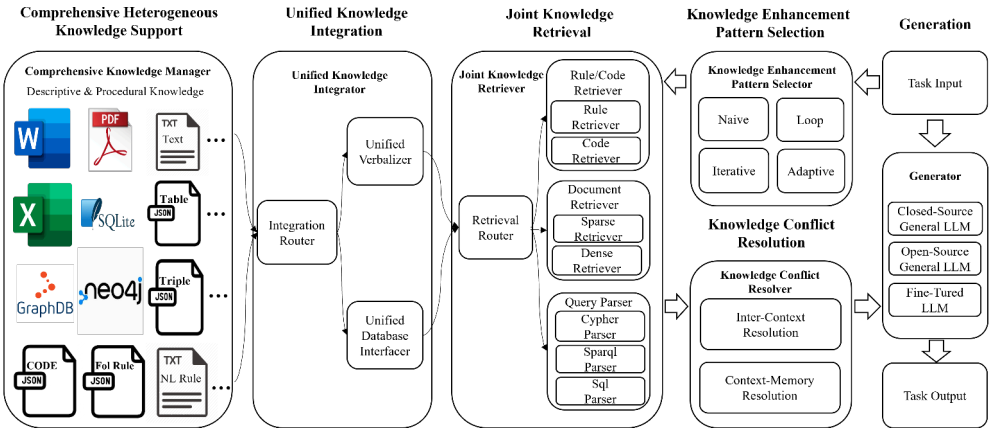


Figure 1: A overview framework of the KMatrix-2 toolkit

Wu et al. 2024, Edge et al. 2024) provide insufficient support for conflict resolution among external knowledge and LLMs' parametric knowledge.

To fill these gaps, we develop KMatrix-2: a comprehensive heterogeneous knowledge collaborative enhancement toolkit for large language model. As shown in Figure 1, KMatrix-2 contains six main stages, and it specifically provides Unified Knowledge Integration and Joint Knowledge Retrieval stages to achieve descriptive and procedural heterogeneous knowledge collaborative enhancement. Meanwhile, a rich of modular components (like Retriever, Generator, etc) and several typical enhancement patterns (like Loop, Adaptive, etc) are encapsulated, and can be combined to conveniently construct mainstream heterogeneous K-LLMs systems, which balances flexibility and ease of use. Moreover, we integrate inter-context and context-memory conflict resolution methods (Hong et al. 2023, Zhou et al. 2023) in KMatrix-2 to mitigate two kinds of conflicts: 1) conflict between external knowledge, and 2) conflict between external knowledge and LLMs' parametric knowledge, respectively. Our main contributions are:

1. The paper proposes a comprehensive heterogeneous knowledge collaborative enhancement toolkit (KMatrix-2) for LLMs. Compared with previous toolkits, which mainly focused on descriptive knowledge, KMatrix-2 specifically considers the enhancement on procedural knowledge.

2. KMatrix-2 offers a rich selection of modular components and several typical enhancement patterns to support convenient construction of mainstream heterogeneous K-LLMs systems.

3. We offer systematic knowledge conflict resolution methods for better knowledge integration.

4. Some key research questions in heterogeneous K-LLMs systems are analyzed, and our toolkit's capability in building such systems is validated.

2 KMatrix-2 Toolkit

As shown in Figure 1, our toolkit contains six stages to complete knowledge-enhanced generation task. Specifically, *Comprehensive Heterogeneous Knowledge Support* is used for the access of comprehensive knowledge. *Unified Knowledge Integration* supports the fusion of comprehensive knowledge. *Joint Knowledge Retrieval* supports the query of the integrated knowledge. *Knowledge Conflict Resolution* is used to mitigate conflicts among external knowledge and LLMs' parametric knowledge, and *Knowledge Enhancement Pattern Selection & Generation* is used for knowledge enhancement process control and answer generation. KMatrix-2 offers a rich selection of modular components and several typical enhancement patterns to implement the above six stages.

2.1 Comprehensive Heterogeneous Knowledge Support

We develop a *Comprehensive Knowledge Manager* component to support heterogeneous knowledge access and management. As shown in Table 1, KMatrix-2 supports the access of rich descriptive and procedural knowledge. It supports three forms of descriptive knowledge: text, table, and knowledge graph, and each knowledge form supports multiple formats. For example, textual knowledge includes formats like WORD and PDF. Tables includes formats such as EXCEL and SQLite. Knowledge graphs include graph databases like GraphDB and Neo4j. Additionally, KMatrix-2 supports two types of representative procedural knowledge: rule and code knowledge (Li et al. 2024). In our toolkit, rule knowledge could be represented as natural languages or formal languages (such as first-order logic). And code knowledge could be represented

as 14 mainstream languages, like Python, Java, etc.

Table 1: Comprehensive Knowledge Support

Knowledge Type	Knowledge Form	Format
Declarative knowledge	Text	WORD, PDF, TXT, Markdown, etc
	Table	EXCEL, Table JSON, SQLite, etc
	Knowledge Graph	Triple JSON, Neo4j, GraphDB, etc
Procedural knowledge	Rule	NL Rule TXT,etc
		FOL Rule JSON, etc
	Code	Formal Code JSON, etc

2.2 Unified Knowledge Integration

All descriptive and procedural knowledge can be divided into two formats: files (like WORD, EXCEL, Rule JSON, etc) and databases (like SQLite, GraphDB, Neo4j, etc). We achieve unified knowledge integration through *Unified Knowledge Integrator*. Specifically, we develop a *Integration Router* component to automatically distribute knowledge in format of files and databases (according to the tag of knowledge format), and build corresponding knowledge integrators for them.

For knowledge in format of files, we build a *Unified Verbalizer* component to achieve knowledge integration. The Unified Verbalizer converts different files into unified text fragments. Following KMatrix's Unified Verbalizer (Wu et al. 2024), we further develop a Verbalizer model for rules and code knowledge using template-based method. (e.g., {*rule body*: an animal lives in a cold area, *rule head*: it probably has thick fat or think fur} => If an animal lives in a cold area, then it probably has thick fat or think fur.)

For knowledge in format of databases, we build a *Unified Database Interfacer* component to achieve knowledge integration. The Unified Database Interfacer standardizes and integrates query interfaces of different types of databases. We develop a Unified Database Interfacer that supports SQLite, GraphDB, Neo4j, etc.

2.3 Joint Knowledge Retrieval

The knowledge outputted by the *Unified Knowledge Integrator* includes three types: a) text fragments from declarative knowledge, b) text fragments from procedural knowledge, and c) unified database interfacer. To achieve joint knowledge retrieval, we develop corresponding *Retriever* components for each type of knowledge and build a *Retrieval Router* component to automatically distribute knowledge to their respective Retrievers.

For text fragments from declarative knowledge, we develop the *Document Retriever* component to perform declarative knowledge retrieval, which includes sparse retrievers: BM25[3], and dense retrievers: Contriever (Izacard et al. 2021), DPR (Karpukhin et al. 2020), BGE (Xiao et al. 2023), Qwen3 (Zhang et al. 2025).

For text fragments from procedural knowledge, considering the inherent semantic gaps between procedural knowledge (code & rule) and declarative knowledge (Li et al. 2024), we build dedicated *Code and Rule Retriever* components with self-induction method, which utilizes LLMs to induce potential rules or code from natural language queries. And these induced rules or code are used for query augmentation to alleviate semantic gaps and improve retrieval effectiveness. A simplified instruction for self-induction method are provided in Appendix A.1.

For unified database interfacer, we develop corresponding *Query Parser* components for the query interfaces of SQLite, GraphDB, and Neo4j databases, which receive input contents and generate query statements specifically tailored for the query interfaces to obtain knowledge. Specifically, KMatrix-2 includes: 1) SQLite-oriented Sql Parser: a seq2seq SQL query generator based on Li et al. (2023a), 2) GraphDB-oriented Sparql Parser: a LLM-based SPARQL query generator inspired by Xu et al. (2023), and 3) Neo4j-oriented Cypher Parser: a Cypher query generator by transforming SPARQL into Cypher using a unified intermediate representation (Xu et al. 2023).

2.4 Knowledge Conflict Resolution

We build inter-context and context-memory conflict resolution methods to mitigate: 1) conflict between external knowledge, and 2) conflict between external knowledge and LLMs' parametric knowledge, respectively. And they are implemented by *Knowledge Conflict Resolver* component.

For inter-context (IC) conflict resolution, In addition to the conventional ICL (Dong et al. 2022) approach, we follow Hong et al. (2023), and use instruction-based discrimination to eliminate erroneous or conflicting information in external knowledge. A instruction for discrimination could be as follows: *Given these passages, some passages may*

[3] https://pypi.org/project/rank-bm25/

have been perturbed with wrong information. Find the perturbed passages if there are any.

For context-memory (CM) conflict resolution, KMatrix-2 supports two strategies: 1) faithful to context: aiming to align with context and focus on context prioritization (Xu et al. 2024). In specific, we follow Zhou et al. (2023), and employ counterfactual demonstration to realize this target. 2) factuality improvement: aiming for an integrated response leveraging both context and parametric knowledge towards a more truthful solution (Xu et al. 2024). In specific, we develop two methods based on COIECD (Yuan et al. 2024) and CAD (Shi et al. 2024), respectively. COIECD employs adaptive decoding with a contextual information-entropy constraint, while CAD utilizes context-aware decoding to implement this strategy.

2.5 Knowledge Enhancement Pattern Selection & Generation

KMatrix-2 designs four typical enhancement patterns to support the convenient construction of mainstream heterogeneous K-LLM systems, including *Naive, Loop, Iterative, and Adaptive*. Diagrams of these enhancement patterns are provided in Appendix A.3.

1) *Naive enhancement pattern*: this pattern supports the construction of naive K-LLMs system, which executes one-time retrieval and generation, and the representative works of naive K-LLMs system contain: DenseX (Chen et al. 2023), UPRISE (Cheng et al. 2023), etc.

2) *Loop enhancement pattern*: this pattern supports the construction of loop K-LLMs system, which executes query decomposition to obtain N subqueries firstly, and performs a N-step retrieval loop for these subqueries. Finally, it completes generation based on the retrieved knowledge. The representative works of loop K-LLMs system contain: COK (Li et al. 2023b), KnowledGPT (Wang et al. 2023), etc.

3) *Iterative enhancement pattern*: this pattern supports the construction of iterative K-LLMs system, which executes iterative retrievals and generations, and the representative works of iterative K-LLMs system contain: Interleave (Shao et al. 2023), ITRG (Feng et al. 2024), etc.

4) *Adaptive enhancement pattern*: this pattern supports the construction of adaptive K-LLMs system, which executes adaptive retrievals and generations on demand, and the representative works of adaptive K-LLMs system contain: SelfRAG (Asai et al. 2023), FLARE (Jiang et al. 2023b), etc.

To meet the needs of different generation scenarios, KMatrix-2 further integrates multiple *Generators* components on base of KMatrix (Wu et al. 2024). Our toolkit further includes: 1) a closed-source general Generator: GPT-4, 2) three open-source general Generators: Baichuan-2-13b (Yang et al. 2023), DeepSeek-R1 (Guo et al. 2025), Qwen2.5-14B-Instruct (Hui et al. 2024).

2.6 Toolkit Design & Usage

We design a concise user interface (UI) to hide K-LLMs design details for ease of use.[4] A rich of modular components and several typical enhancement patterns are encapsulated, and can be combined to conveniently construct mainstream K-LLMs systems. User interface (UI) of a loop K-LLMs system deployment is shown in Figure 2, and the usage instructions of our toolkit are provided in Appendix A.2.

3 Experimental Settings

3.1 Datesets and Knowledge

Evaluation datasets are shown in Table 6 (Appendix A.4). To evaluate the performance of knowledge retrieval, this paper selects COIR (Li et al. 2024) and RuleBench (Sun et al. 2024). In specific, COIR is used for code knowledge retrieval evaluation, which contains 14 main code languages. RuleBench is used for rule knowledge retrieval evaluation, which contains natural language and first-order logic rules. To evaluate the semantic parsing performance of Query Parser, this paper selects WWQ (Xu et al. 2023) and spider_realistic (Deng et al. 2020) to assess performance of Sparql and Sql Parsers, respectively. To evaluate descriptive and procedural knowledge enhancement performance of K-LLMs system, this paper selects: 1) 6 open domain question answering (QA) datasets from KMatrix, which is used to evaluate descriptive and procedural knowledge enhancement performance on factual QA task, and 2) RuleBench, which contains rule-following QA datasets from 5 domains and is used to evaluate descriptive and procedural knowledge enhancement performance on inferential QA task. And this paper selects ConflictBank (Su et al. 2024) and ConflictQA (Xie et al. 2024) to assess performance of IC and CM knowledge conflict resolution methods, respectively.

Heterogeneous knowledge used in evaluation is shown in Table 7 (Appendix A.4). For descriptive

[4] https://github.com/NLPerWS/KMatrix-2

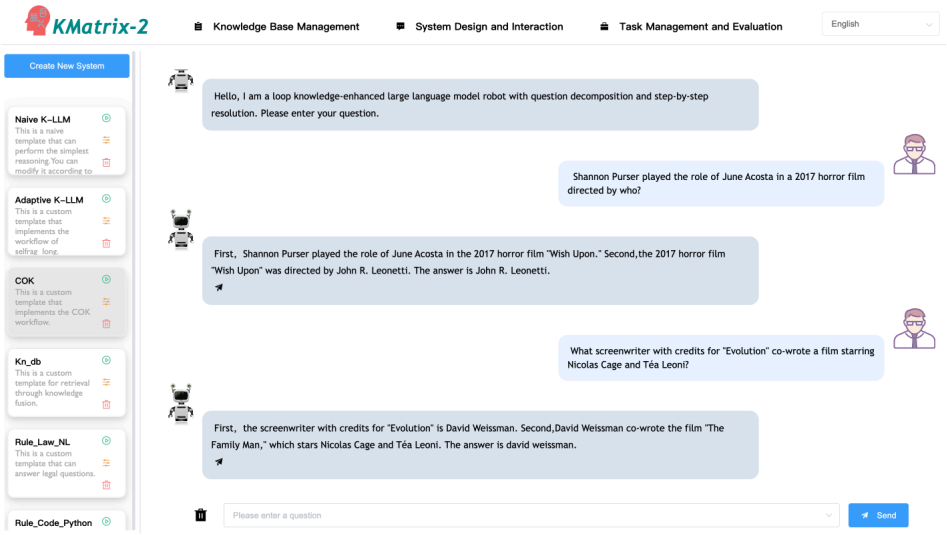


Figure 2: Deployment UI of KMatrix-2: toolkit usage instructions are provided in Appendix A.2.

knowledge, we integrate Wikipedia (Chen et al. 2017), Wikidata (Vrandečić and Krötzsch 2014) and Wikitable (Ma et al. 2022). For procedural knowledge, we integrate the natural language and first-order logic rules in RuleBench.

3.2 Task Settings

Some key research questions (RQs) in heterogeneous K-LLMs systems are analyzed, and our toolkit's capability in building such systems is validated.

RQ1: Can Self-Induction method bridge the semantic gap in procedural knowledge retrieval to boost performance? To validate this, we evaluate four retrieval models on two types of procedural knowledge: code and rules. The models include: 1) two representative dense retrievers: BGE (Xiao et al. 2023) and Qwen3 (Zhang et al. 2025), 2) their enhanced versions with our Self-Induction method: Self-Induction+BGE and Self-Induction+Qwen3.

RQ2: Do factual question answering (QA) and inferential QA tasks rely on different types of knowledge? To validate this, we evaluate descriptive and procedural knowledge enhancement performance of K-LLMs system on factual and inferential QA tasks. We use Naive K-LLMs (one-time retrieval & generation) as the evaluated system. Baichuan-2-13B (Yang et al. 2023) is used as the Generator. BGE (Xiao et al. 2023) is used as the procedural knowledge Retriever. Contriever (Izacard et al. 2021) is used as the descriptive knowledge Retriever. The descriptive and procedural knowledge in Table 7 (Appendix A.4) is used as the knowledge bases.

RQ3: Can knowledge conflict resolution methods further improve the performance of K-LLM systems? To validate this, we evaluate: 1) the CM conflict resolution method: COIECD (Yuan et al. 2024) and CAD (Shi et al. 2024), 2) the IC conflict resolution method: ICL (Dong et al. 2022) and Discriminator (Hong et al. 2023).

4 Experimental Analyses

We report experimental results and analyses as follows:

RQ1: The answer is YES. Table 2 summarizes the evaluation results of procedural knowledge retrieval. Specifically, Tables 2a and 2b detail the performance evaluations of the four retrievers for rule and code knowledge retrieval, respectively. Compared to the two representative dense retrievers (BGE and Qwen3), their enhanced versions with our Self-Induction method demonstrate consistent performance improvements across most datasets, *revealing our method's potential in mitigating semantic gaps for procedural knowledge retrieval.*

Considering that KMatrix has evaluated the retrieval performance of declarative knowledge, we further report the semantic parsing performance of the two integrated Query Parsers in Table 3. They exhibit strong semantic parsing capabilities.

RQ2: Factual and Inferential QA tasks rely on different types of knowledge. Table 4 summarizes the evaluation results. Specifically, Table 4a and Table 4b show descriptive & procedural knowledge enhancement performance on Factual and Inferential QA tasks, respectively. On Factual QA tasks, Compared to answer generation without knowledge, the performance of K-LLMs system with descriptive knowledge enhancement shows a significant improvement. The collabora-

Table 2: Procedural knowledge retrieval performance evaluation

(a) Rule knowledge retrieval performance evaluation (Recall@10)

	Clutrr	Deer	Law	Ulogic	TheoremQA	Clutrr-Fol	Ulogic-Fol	Law-Fol
BGE	4.77%	100%	97.59%	95.06%	67.45%	4.96%	90.72%	92.77%
Self-Induction+BGE	15.08%	100%	98.19%	98.67%	80.36%	13.55%	98.80%	97.59%
Qwen3	3.34%	100%	96.39%	96.63%	81.27%	3.15%	92.41%	93.98%
Self-Induction+Qwen3	13.84%	100%	98.80%	98.80%	84.00%	13.07%	96.99%	98.19%

(b) Code knowledge retrieval performance evaluation (Recall@10)

	Codetrans-Dl	Stackoverflow	Apps	Codefeedback	Codetrans-Contest	Text2sql	Cosqa
BGE	59.44%	87.51%	41.71%	48.81%	69.25%	79.36%	41.20%
Self-Induction+BGE	57.84%	87.70%	48.12%	50.36%	70.83%	92.13%	44.02%
Qwen3	77.08%	94.19%	90.86%	93.47%	93.06%	95.58%	60.95%
Self-Induction+Qwen3	79.41%	94.52%	93.19%	94.61%	93.65%	98.18%	58.51%

Table 3: Semantic parsing performance of Query Parser (EM)

Sparql Parser Evaluation		Sql Parser Evaluation	
Model	EM	Model	EM
WikiSP	75.55%	RESDSQL(t5-3b)	72.64%

Table 4: Descriptive & procedural knowledge enhancement performance evaluation

(a) Descriptive & procedural knowledge enhancement performance evaluation on Factual QA tasks (Acc)

Knowledge	Popqa	Webqa	NQ	Triviaqa	Hotpotqa	2Wikiqa
Without	23.45%	29.53%	13.24%	42.86%	16.65%	23.75%
Declarative	62.26%	37.60%	36.62%	61.70%	27.90%	26.45%
Declarative+Procedural	62.54%	36.96%	35.84%	60.74%	27.20%	26.35%

(b) Descriptive & procedural knowledge enhancement performance evaluation on Inferential QA tasks (Acc)

Knowledge	Clutrr	Deer	Law	Ulogic	Clutrr-Fol	Ulogic-Fol	Law-Fol
Without	25.19%	85.71%	27.71%	11.81%	24.81%	11.81%	27.11%
Declarative	27.96%	95.24%	22.89%	48.80%	28.34%	48.55%	23.49%
Declarative+Procedural	31.97%	95.24%	40.96%	58.43%	32.06%	58.55%	40.96%

tive enhancement performance of descriptive and procedural knowledge does not differ significantly from that of descriptive knowledge enhancement alone, *indicating that Factual QA tasks primarily rely on descriptive knowledge, with a weak dependence on procedural knowledge.* On Inferential QA tasks, Compared to answer generation without knowledge, the K-LLMs system with descriptive knowledge enhancement demonstrates a marked performance improvement, and the collaborative enhancement of descriptive and procedural knowledge further boosts performance, *highlighting that inferential QA tasks require both descriptive and procedural knowledge.*

RQ3: knowledge conflict resolution further improves the performance of K-LLM systems. Table 5 displays the evaluation results. Compared to method without knowledge, the K-LLMs systems with knowledge enhancement demonstrate a significant performance improvement. Our developed CM and IC knowledge conflict resolution methods both yield additional substantial gains, *demonstrating their capability in mitigating hallucinations and improving factual accuracy of LLMs.*

Table 5: Knowledge conflict resolution evaluation

	Method	Acc
CM Conflict Resolution	w/o knowledge	10.00%
	w/ knowledge	56.65%
	+COIECD	63.98%
	+CAD	60.08%
IC Conflict Resolution	w/o knowledge	1.00%
	w/ knowledge	21.00%
	+ICL	34.90%
	+Discriminator	54.00%

5 Conclusions

The paper presents KMatrix-2, an open-source toolkit that supports comprehensive heterogeneous knowledge enhancement for LLMs. Unified knowledge integration and joint knowledge retrieval methods are provided to achieve descriptive and procedural knowledge collaborative enhancement. Meanwhile, systematic knowledge conflict resolution methods are offered for better knowledge integration. We develop a rich selection of components and several typical enhancement patterns to support convenient construction of mainstream K-LLMs systems. Some key research questions are analyzed and comparative performance results demonstrate the capabilities of KMatrix-2.

Limitations

KMatrix-2 currently has some limitations, which we will gradually improve in the future. 1) Although our toolkit supports the construction of Adaptive K-LLMs systems, it lacks comprehensive and systematic support for the currently popular Agentic RAG. This is an area we need to strengthen in our follow-up work. 2) Regarding the productization of our toolkit, we need to do more work.

Acknowledgements

This work was supported by Beijing Natural Science Foundation (L243006) and the National Natural Science Foundation of China (No.62376270).

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Harrison Chase. 2022. Langchain, october 2022. *URL https://github.com/langchain-ai/langchain*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2023. Dense x retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*.

Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Denvy Deng, and Qi Zhang. 2023. Uprise: Universal prompt retrieval for improving zero-shot evaluation. *arXiv preprint arXiv:2303.08518*.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2020. Structure-grounded pretraining for text-to-sql. *arXiv preprint arXiv:2010.12773*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Denis Emelin, Daniele Bonadiman, Sawsan Alqahtani, Yi Zhang, and Saab Mansour. 2022. Injecting domain knowledge in language models for task-oriented dialogue systems. *arXiv preprint arXiv:2212.08120*.

Zhangyin Feng, Xiaocheng Feng, Dezhi Zhao, Maojin Yang, and Bing Qin. 2024. Retrieval-generation synergy augmented large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11661–11665. IEEE.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Giwon Hong, Jeonghwan Kim, Junmo Kang, Sung-Hyon Myaeng, and Joyce Jiyoung Whang. 2023. Why so gullible? enhancing the robustness of retrieval-augmented models against counterfactual noise. *arXiv preprint arXiv:2305.01579*.

Yasuto Hoshi, Daisuke Miyashita, Youyang Ng, Kento Tatsuno, Yasuhiro Morioka, Osamu Torii, and Jun Deguchi. 2023. Ralle: A framework for developing and evaluating retrieval-augmented large language models. *arXiv preprint arXiv:2308.10633*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Peter Izsak, Moshe Berchansky, Daniel Fleischer, and Ronen Laperdon. 2023. fastrag: Efficient retrieval augmentation and generation framework.

Janis E Jacobs and Scott G Paris. 1987. Children's metacognition about reading: Issues in definition, measurement, and instruction. *Educational psychologist*, 22(3-4):255–278.

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.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *arXiv preprint arXiv:2405.13576*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Yichun Yin, Hao Zhang, Yong Liu, Yasheng Wang, and Ruiming Tang. 2024. Coir: A comprehensive benchmark for code information retrieval models. *arXiv preprint arXiv:2407.02883*.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq Joty, and Soujanya Poria. 2023b. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *arXiv preprint arXiv:2305.13269*, 3.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2022. Open-domain question answering via chain of reasoning over heterogeneous knowledge. *arXiv preprint arXiv:2210.12338*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering. *arXiv preprint arXiv:2012.14610*.

Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, et al. 2019. Haystack: the end-to-end nlp framework for pragmatic builders. and denny zhou. 2022b. chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294*.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. 2024. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791.

Zhaochen Su, Jun Zhang, Xiaoye Qu, Tong Zhu, Yanshu Li, Jiashuo Sun, Juntao Li, Min Zhang, and Yu Cheng. 2024. Conflictbank: A benchmark for evaluating the influence of knowledge conflicts in llm. *arXiv preprint arXiv:2408.12076*.

Wangtao Sun, Chenxiang Zhang, XueYou Zhang, Xuanqing Yu, Ziyang Huang, Pei Chen, Haotian Xu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Beyond instruction following: Evaluating inferential rule following of large language models. *arXiv preprint arXiv:2407.08440*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10).

Siyuan Wang, Zhongyu Wei, Yejin Choi, and Xiang Ren. 2024. Can llms reason with rules? logic scaffolding for stress-testing and improving llms. *arXiv preprint arXiv:2402.11442*.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *arXiv preprint arXiv:2308.11761*.

Shun Wu, Di Wu, Kun Luo, XueYou Zhang, Jun Zhao, and Kang Liu. 2024. Kmatrix: A flexible heterogeneous knowledge enhancement toolkit for large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 280–290.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighof. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations*.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge conflicts for llms: A survey. *arXiv preprint arXiv:2403.08319*.

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina J Semnani, and Monica S Lam. 2023. Fine-tuned llms know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over wikidata. *arXiv preprint arXiv:2305.14202*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Xiaowei Yuan, Zhao Yang, Yequan Wang, Shengping Liu, Jun Zhao, and Kang Liu. 2024. Discerning and resolving knowledge conflicts through adaptive decoding with contextual information-entropy constraint. *arXiv preprint arXiv:2402.11893*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023. Context-faithful prompting for large language models. *arXiv preprint arXiv:2303.11315*.

A Appendix

A.1 A simplified instruction for self-induction method

You are given a Context and a Question. Please write the inferential rule that may help answer the question. A demonstration example is as follows:
Context: Artist Alice is commissioned for a large mural titled "Sunset Dreams" in the city center, involving vibrant colors and abstract shapes to represent the diversity of the community.
Question: Does Alice need to create "Sunset Dreams"?
Rule: Commissioned(Person X, Artwork Y) => NeedToCreate(Person X, Artwork Y)

A.2 The Usage Instructions of KMatrix-2

KMatrix-2 consists of three sections: Knowledge Base Management, System Design and Interaction, and Task Management and Evaluation. *The screencast video* of our toolkit are available here.[5]

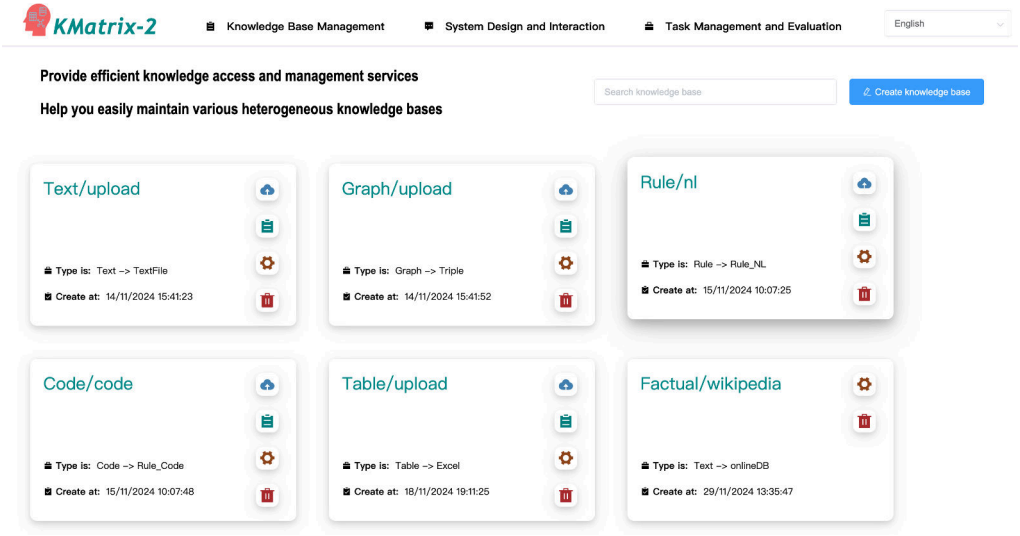
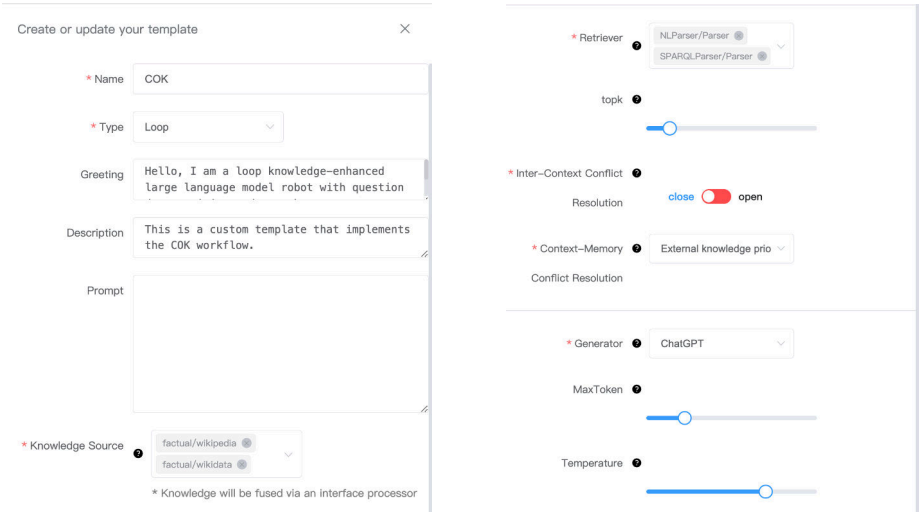


Figure 3: Knowledge Base Management interface: providing efficient heterogeneous knowledge access and management services. We can easily create descriptive and procedural knowledge bases, including Text, Table, Knowledge Graph, Rule and Code. And upload corresponding files (like PDF, WORD, Rule JSON, etc) or databases (like SQLite, Neo4j, GraphDB, etc). All knowledge bases we create will be uniformly managed, and users can easily search and modify the information in the knowledge bases.



[5] https://youtu.be/E7hk-jrM2CY

Figure 4: System Design and Interaction: supporting the construction, deployment, and user interaction of the K-LLMs system. We can construct K-LLMs system by: 1) selecting system enhancement pattern (like Adaptive and Iterative), 2) selecting knowledge source, 3) selecting Retriever and configuring parameters, 4) selecting knowledge conflict resolution strategy, 5) selecting Generator and configuring parameters. After that, we can deploy the K-LLMs system and interact with it. The interface of System Deployment and Interaction is shown in Figure 2.

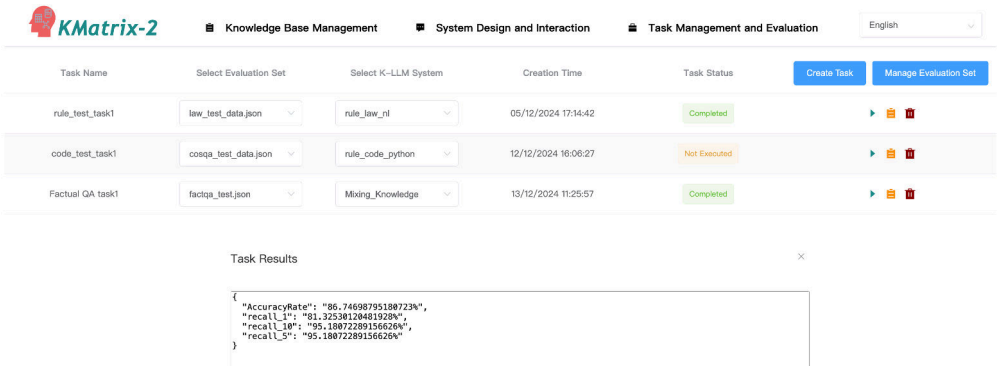


Figure 5: Task Management and Evaluation interface: support for constructing, managing and executing evaluation tasks for the K-LLMs system. We can create evaluation task, and select evaluation dataset & K-LLMs system we create. After that, we can run the evaluation task, and the evaluation results will be presented.

A.3 The Enhancement Patterns of KMatrix-2

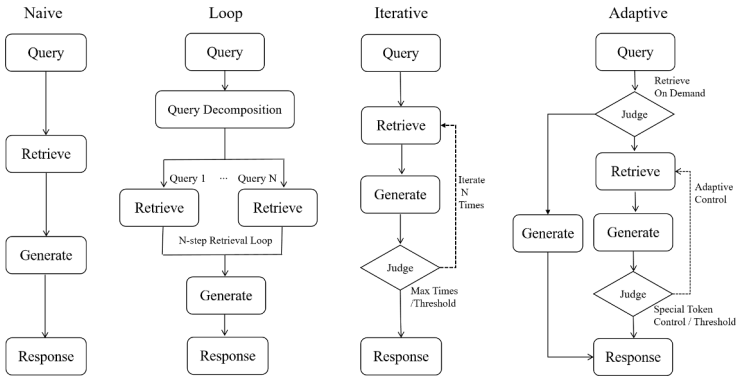


Figure 6: The Enhancement Patterns of KMatrix-2

A.4 Heterogeneous knowledge and datasets used in evaluation

Table 6: Evaluation datasets used by KMatrix-2

Dataset Type	Dataset Name	Dataset Scale
Code Retrieval	COIR	223358
Rule Retrieval	RuleBench	4527
Sparql Parsing	WWQ	454
Sql Parsing	spider_realistic	508
Factual QA	2Wikiqa	12576
	Hotpotqa	7405
	NQ	3610
	Popqa	1399
	Triviaqa	7313
	Webqa	2032
Inferential QA	RuleBench	4527
IC Conflict Resolution	ConflictBank (subset)	1000
CM Conflict Resolution	ConflictQA	8027

Table 7: Heterogeneous knowledge used in evaluation

Knowledge Type	Knowledge Name	Knowledge Scale
Descriptive Knowledge	Wikipedia	21000k
	Wikidata	5790k
	Wikitable	6860k
Procedural knowledge	NL Rule (RuleBench)	2462
	FOL Rule (RuleBench)	2065

GlotEval: A Test Suite for Massively Multilingual Evaluation of Large Language Models

Hengyu Luo[1], Zihao Li[1], Joseph Attieh[1†], Sawal Devkota[2†], Ona de Gibert[1†], Xu Huang[4†], Shaoxiong Ji[2,5,6†], Peiqin Lin[3†], Bhavani Sai Praneeth Varma Mantina[2†], Ananda Sreenidhi[2†], Raúl Vázquez[1†], Mengjie Wang[2†], Samea Yusofi[2†], Fei Yuan[7], Jörg Tiedemann[1,5]

[1]University of Helsinki, Finland [2]Technical University of Darmstadt, Germany
[3]University of Munich, Germany [4]Nanjing University, China [5]ELLIS Institute Finland
[6]University of Turku, Finland [7]Shanghai Artificial Intelligence Laboratory, China

Correspondence: hengyu.luo@helsinki.fi & shaoxiong.ji@utu.fi [†] Equal contribution.

Abstract

Large language models (LLMs) are advancing at an unprecedented pace globally, with regions increasingly adopting these models for applications in their primary languages. Evaluating these models in diverse linguistic environments, especially in low-resource languages, has become a major challenge for academia and industry. Existing evaluation frameworks suffer from inconsistency across different benchmarks, being disproportionately focused on English and a handful of high-resource languages, thereby overlooking the realistic performance of LLMs in multilingual and lower-resource scenarios. To address this critical challenge of fragmented and inconsistent multilingual evaluation, we introduce GlotEval, a unified and lightweight framework that systematically integrates 27 benchmarks under a standardized ISO 639-3 language identifier system, allowing for seamless incorporation of new benchmarks. Supporting nine key tasks (machine translation, text classification, summarization, open-ended generation, reading comprehension, sequence labeling, intrinsic evaluation, instruction following and reasoning), spanning over dozens to hundreds of languages, GlotEval uniquely enables language-specific, cross-benchmark analysis and non-English-centric evaluations at a scale previously less practical for many researchers. This enables a precise diagnosis of model strengths and weaknesses in diverse linguistic contexts. A multilingual translation case study demonstrates GlotEval's applicability for multilingual and language-specific evaluations.

GlotEval: github.com/MaLA-LM/GlotEval

1 Introduction

In recent years, driven by rapid progress in natural language processing and deep learning, large language models (LLMs) such as GPT-4 (OpenAI, 2023) and DeepSeek-R1 (DeepSeek-AI et al., 2025) have shown remarkable reasoning and generation capabilities across multiple languages and tasks. Although these models approach or surpass expert-level performance in certain high-resource languages (e.g., English), they often exhibit substantial performance fluctuations in other linguistic environments (Zhang et al., 2024). This discrepancy partially arises from the imbalance and scarcity of training data of low-resource languages, and partially from the limited multilingual coverage of current evaluation frameworks: many were originally designed for English or a few widely spoken languages, making it difficult to extend them efficiently to more diverse linguistic tasks or to adapt custom prompts and configurations for each language. Meanwhile, as LLMs proliferate worldwide and different regions rely on their respective local languages, large-scale (massively) multilingual evaluation involving numerous low-resource languages has emerged as a critical research direction.

Recent developments in LLM evaluation toolkits such as EleutherAI's LM Evaluation Harness (Gao et al., 2023) and UltraEval (He et al., 2024) have facilitated automatic evaluation. However, significant gaps persist in language coverage, task diversity, and evaluation flexibility (Chang et al., 2024), especially in evaluating multilingual LLMs in a massively multilingual scenario. To address these issues, we present GlotEval, an evaluation framework designed to provide *systematic* support for a *broad range of languages*, with a strong focus on low-resource ones. Building on the core processes of LLM evaluation—data preparation, model inference, post-processing, and metric computation—GlotEval introduces three novel features.

1. **Consistent Cross-benchmark Multilingual Evaluation.** We integrate 27 existing multilingual benchmarks into a unified pipeline, by standardizing all ISO 639-3 language codes

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 602–614
November 4-9, 2025

in the different benchmarks,[1] which is an accepted standard with a good coverage of the world's languages. By aligning benchmark language identifiers with ISO 639-3 codes, we enable evaluations for specific languages or language groups (e.g., Bantu, Dravidian, or Uralic languages), allowing the framework to automatically search among integrated benchmarks to find matching test sets. This mapping also makes it easier to incorporate new large-scale benchmarks that target mid- or low-resource languages, ensuring flexibility for future expansions.

2. **Language-Specific Prompt Templates.** Users can configure prompts for each language individually, thereby enabling more precise assessments of a model's instruction-following ability across diverse linguistic settings. All templates are maintained in a centralized prompt library that supports multilingual benchmarks, allowing easy customization as needed. In this way, each task within a benchmark can be run potentially using prompts in the task's original language, rather than defaulting to English prompts. To simplify cross-lingual adaptation, we also implemented Microsoft Translator integration that automatically propagates user-defined prompt templates from one single language to 130+ supported languages.[2]

3. **Non-English-Centered Machine Translation Evaluation.** GlotEval is designed to break away from the traditional English-centric paradigm. Thanks to translation benchmarks featuring fully or partially multi-aligned datasets, GlotEval enables non-English-centered translation evaluations by allowing any supported language to serve as the pivot: users simply update the pivot language in the configuration to assess "any-to-pivot" / "pivot-to-any" translation directions. This flexibility ensures that GlotEval breaks from the traditional "English $\leftrightarrow$ other language" paradigm and adapts seamlessly to diverse, potentially low-resource, language pairs.

By bringing all these capabilities together in a cohesive framework, GlotEval aims to facilitate large-scale, in-depth evaluations of multilingual LLMs across both widely spoken and underrepresented languages, ultimately driving forward more inclusive LLM evaluation. Thus, GlotEval's primary contribution is not the collection of new tasks, but the synergistic integration and standardization of existing benchmarks, which can be a robust tool for researchers and developers conducting massively multilingual LLM evaluation.

2 Related Work

Several evaluation toolkits and benchmarks have been developed to systematically assess LLMs. EleutherAI's LM Evaluation Harness (Gao et al., 2023) is a widely adopted framework covering over 60 tasks, including multilingual datasets such as XNLI (15 languages) and Belebele (122 languages). UltraEval (He et al., 2024) improves modularity and supports FLORES-200 for multilingual translation. OpenAI Evals provides a highly flexible, community-driven framework,[3] and OpenCompass (Contributors, 2023) offers a comprehensive platform with broad support for datasets and models. MEGA (Ahuja et al., 2023) evaluates generative LLMs across diverse languages, with a focus on standard NLP benchmarks. LightEval (Fourrier et al., 2023) developed a flexible LLM evaluation framework that supports different backends.

Despite these advancements, significant gaps remain in language coverage, task diversity, and evaluation flexibility (Chang et al., 2024). Specifically, most toolkits rely on static task definitions and rarely adopt standardized language identifiers across benchmarks, making it difficult to conduct language-specific evaluations in a cross-benchmark setting. As a result, evaluations for a given language (group) must often be performed in isolation for each benchmark, limiting scalability and linguistic granularity. Furthermore, support for language-specific prompt customization is limited—most toolkits default to using English prompts regardless of the task language, which failed to take both goals of languages in multilingual evaluation, i.e., task performance versus language understanding, into consideration (Poelman and de Lhoneux, 2024).

[1] https://iso639-3.sil.org/about

[2] https://learn.microsoft.com/en-us/azure/ai-services/translator/language-support

Task	Benchmark	Languages	Domain	Open Source	Metrics
Text Classification	Taxi-1500 (Ma et al., 2024)	1507	Bible text	Yes (GitHub)	Acc., F1
	SIB-200 (Adelani et al., 2024)	205	News topics	Yes (HF, GitHub)	Acc., F1
Token Classification	WikiANN (Pan et al., 2017)	282	Wikipedia NER	Yes (HF)	F1
	UD treebank v2.15 (de Marneffe et al., 2021)	148	POS tagging	Yes (UD website)	F1
Machine Translation	FLORES-200 (NLLB Team et al., 2022)	200+	General web	Yes (HF)	BLEU, ChrF++, COMET
	FLORES+	212	Gen. web, low-resource focus	Yes (HF)	BLEU, ChrF++, COMET
	NTREX-128 (Federmann et al., 2022)	128	News	Yes (GitHub)	BLEU, ChrF++, COMET
	AmericasNLP (de Gibert et al., 2025)	14	Short sentences, court proceedings, books.	Yes (GitHub)	BLEU, ChrF++
	TICO-19 (Anastasopoulos et al., 2020)	37	COVID-19 medical	Yes (GitHub, OPUS)	BLEU, ChrF++
	IN22 (Gala et al., 2023)	23	Indian langs., news+conv.	Yes (GitHub)	BLEU, ChrF++
	NTEU (Bié et al., 2020)	24	EU formal (gov)	Partial (Upon request)	BLEU, ChrF++
	MAFAND (Adelani et al., 2022)	22	News	Yes (GitHub)	BLEU, ChrF++
	Tatoeba Challenge v2023 (Tiedemann, 2020)	500+	Mixed short sents.	Yes (GitHub)	BLEU, ChrF
	OpenSubtitles v2024 (Lison and Tiedemann, 2016)	93	Subtitles	Yes (GitHub)	BLEU, ChrF
	MMHB (Tan et al., 2024)	9	Multilingual bias detection	Yes (GitHub)	ChrF with gender
Open-Ended Generation	Aya (Singh et al., 2024b)	119	Instruction-following	Yes (HF)	self-BLEU
	PolyWrite (Ji et al., 2024)	240	Creative writing	Yes (HF)	self-BLEU
Intrinsic Evaluation	PBC (Mayer and Cysouw, 2014)	372+	Bible text	Partial (Upon request)	NLL
	MaLA (Ji et al., 2024)	546	General web	Yes (HF)	NLL
Comprehension	MMMLU (Hendrycks et al., 2021)	14+	General knowledge QA	Yes (HF)	Acc.
	Global-MMLU (Singh et al., 2024a)	42	Culture-aware QA	Yes (HF)	Acc.
Summarization	XLSum (Hasan et al., 2021)	44	News	Yes (HF, GitHub)	ROUGE
	MassiveSumm Long (Varab and Schluter, 2021)	55	News	Yes (HF)	ROUGE
	MassiveSumm Short (Varab and Schluter, 2021)	88	News	Yes (HF)	ROUGE
Instruction Following	BenchMAX Rule-based (Huang et al., 2025)	17	Verifiable instructions	Yes (HF)	Instruction-level Acc. etc.
Reasoning	BenchMAX Math (Huang et al., 2025)	17	Grade School Math	Yes (HF)	Accuracy
	BenchMAX Science (Huang et al., 2025)	17	Graduate-level Scientific QA	Yes (HF)	Accuracy

Table 1: Overview of multilingual LLM evaluation benchmarks, with typical metrics used in each.

3 GlotEval

3.1 Benchmarks, Languages and Metrics

As shown in Table 1, GlotEval integrates publicly available multilingual benchmark datasets, covering machine translation, text classification, summarization, open-ended generation, reading comprehension, sequence labeling, intrinsic evaluation, instruction following and reasoning, spanning a wide range of languages from high-resource to low-resource. In total, GlotEval comprises 9 tasks and 27 benchmarks, evaluates in over 1500 languages, and utilizes diverse metrics. Refer to Appendix A for more details of supported benchmark datasets.

3.2 Workflow

As shown in Figure 1, the workflow of GlotEval proceeds from specifying which benchmarks and languages to use, to producing final metrics and visualization.

First, users specify their choices and through command-line arguments. Users can specify the language(s) and the benchmark task(s) to evaluate. Besides, as for prompting strategy choice, GlotEval supports two prompting strategies: Setting prompting strategy as *single* along with a chosen prompt language (e.g., `eng_Latn`) applies the same prompt in one single language for every dataset in one benchmark. This is useful for controlling variables or using a single reference prompt style; Setting prompting strategy as *multi* makes GlotEval search for a language-specific template in the prompt library, which corresponds to the tested language, falling back to English if not found. Especially in machine translation tasks, the source language typically determines the prompt's language by default. Further, users can freely modify or expand the prompt library with a built-in multilingual prompt builder.

Upon selecting the desired benchmarks, languages, and prompt strategy, the user triggers GlotEval's data loader to automatically locate each dataset and load the relevant language subsets. It then initializes the appropriate model backend depending on the task type. Specifically, for non-generative tasks, we employ the HuggingFace Transformers backend (Wolf et al., 2020) to ensure more efficient use of computational resources. For generation tasks, such as machine translation, summarization, and open-ended text generation, we prioritize the vLLM backend (Kwon et al., 2023) to ensure high throughput, while retaining the HF Transformers generation interface for compatibility purposes.

After model inference is completed, GlotEval automatically computes evaluation metrics according to the task-specific settings listed in Table 1. Optionally, as mentioned before, by appending `-store_details`, users can export each sample's prompt, model output, reference, and corresponding scores to a JSONL file, which allows researchers to work outside the framework and conduct custom error analysis and result visualization. This ensures that our framework is not just an eval-

[3]https://github.com/openai/evals

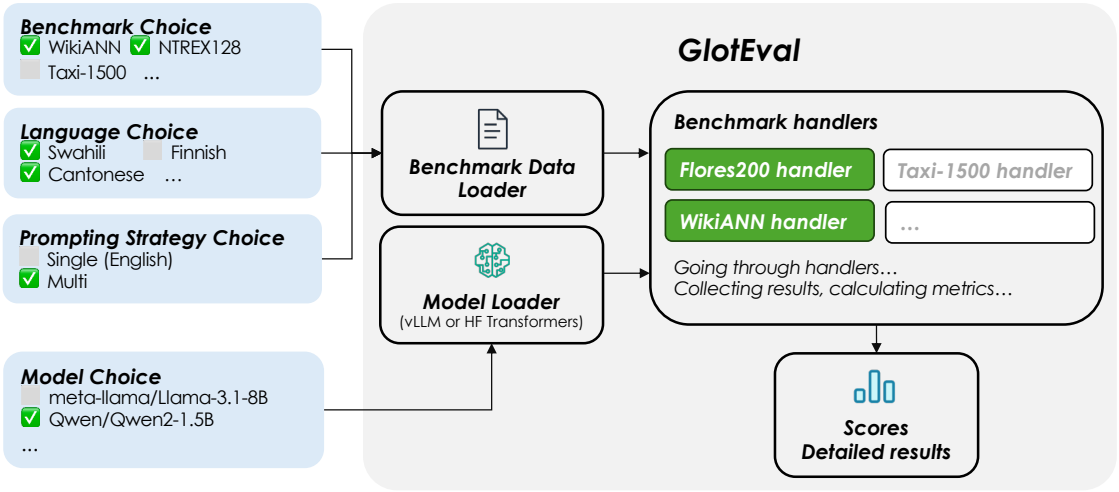


Figure 1: Workflow of GlotEval

uation executor, but also a starting point for more fine-grained analysis.

3.3 A Deeper Look at Benchmark Data Loader

Figure 2 illustrates the overall workflow within GlotEval's data loading and prompt preparation pipeline. At its core, GlotEval aligns language identifiers from various benchmarks to a unified `ISO 639-3_Script` format. Once the alignment is complete, the standardized language codes serve as the central connection for downstream operations. When the user queries GlotEval with a target language (e.g., `zho` for Chinese or `spa` for Spanish), the system consults the language-to-code dictionary and retrieves all benchmark-specific subsets whose original language codes map to the same standardized form. These subsets are then included in the evaluation process. Moreover, if a language-specific prompting strategy is selected, GlotEval uses the same aligned codes to retrieve the appropriate prompt templates from the multilingual prompt library. For example, as shown in Figure 2, querying `zho` and `spa` will automatically select the corresponding benchmark subsets and load their respective prompts (`zho_Hans`, `spa_Latn`) for evaluation. This workflow builds on both the language code alignment mechanism and the multilingual prompt builder described in the following sections.

Language Code Alignment to ISO 639-3

Different benchmarks often use inconsistent codes for the same language (e.g., `zh`, `zho`, `cmn`, `Chinese`, `Mandarin-CN` etc. for Mandarin Chinese). Before reading benchmark datasets via dedicated data loaders, GlotEval unifies these language identifiers used across different benchmarks, to enable cross-benchmark language-specific evaluation and prompting. Figure 3a visualizes this process.

Specifically, we process each benchmark-provided language code—which may appear in the form of ISO 639-1, 639-2/B (bibliographic), 639-2/T (terminological), ISO 639-3 codes, or even language names—by utilizing the `iso639-lang` Python package.[4] This allows us to retrieve all available mappings from the ISO 639-3 standard, including ISO 639-3 identifiers, ISO 639-2/B, 639-2/T, and ISO 639-1 codes. Using both exact and fuzzy matching strategies, we attempt to automatically identify the corresponding ISO 639-3 code for each language. A report is generated that documents, for each benchmark language, whether the match was exact or fuzzy, and whether it corresponds to an individual language or a macrolanguage in the ISO 639-3 standard.

We further identify the script used by each dataset, using `GlotScript` (Kargaran et al., 2024) to detect the dominant script.[5] Here we assume each dataset is primarily in one script. We randomly select up to 100 lines and attempt script recognition into ISO 15924 script code. This ensures each dataset obtains a `<language>_<script>` label, such as `eng_Latn`. The final ISO 639-3 code, along with the script code, is stored as the value in a language-to-code dictionary within the benchmark's configuration

[4] https://pypi.org/project/iso639-lang/
[5] https://pypi.org/project/GlotScript/

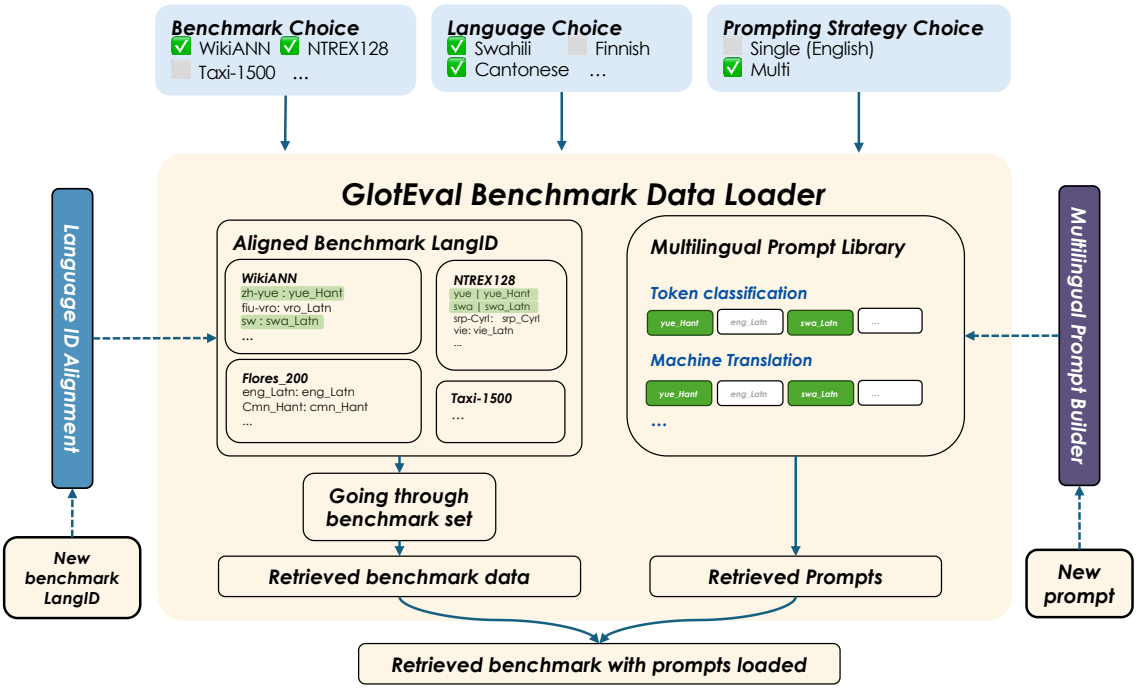


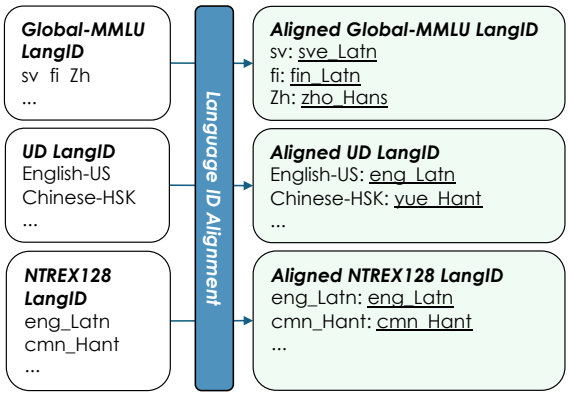
Figure 2: GlotEval benchmark data loader

file. Hence, each language + script combination is standardized in GlotEval for consistent usage across benchmarks.

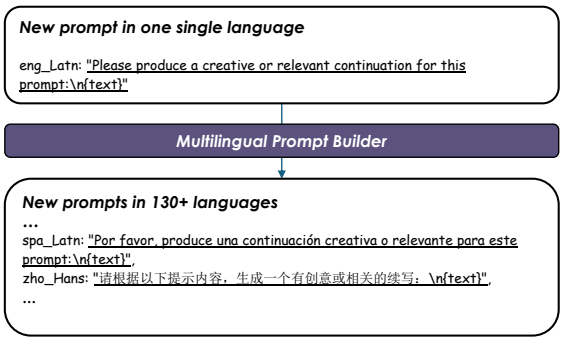
Multilingual Prompt Builder

We constructed a dedicated command-line prompt builder to automatically prepare or adapt prompt templates for multilingual tasks. Figure 3b visualizes this process. In particular, the builder leverages Microsoft Translator to convert an instruction and/or few-shot prompt template from a given source language into 130+ target languages, while ensuring that placeholders (e.g., `{src_text}`) remain intact during translation. These newly created multilingual prompts, are stored in the updated prompt library. As a result, each dataset's prompts are aligned with the same `<language>_<script>` language taxonomy, enabling consistent, language-specific evaluation.

Note that the automatic translation of prompts is intended as a convenience feature to support rapid, large-scale multilingual evaluation. While translation quality may vary—particularly for low-resource languages—this approach offers a practical starting point for exploratory analysis with language-specific prompts at scale. The framework remains fully customizable: users are able to provide their own human-written or verified prompts in the prompt library for languages of interest.



(a)



(b)

Figure 3: Benchmark data loader components: (a) Language ID alignment process and (b) multilingual prompt generation.

4 Evaluation

4.1 Efficiency Analysis

We benchmark GlotEval's inference speed on six tasks: FLORES-200, Aya, and XLSum for generative tasks, and SIB-200, Global-MMLU, and WikiANN for non-generative tasks. All evaluations are conducted on 19 languages spanning diverse writing systems (e.g., Latin, Arabic, Cyrillic, Devanagari, Chinese, etc.). For each language, we sample 10 examples per task for evaluation. We choose Qwen2-1.5B model (Yang et al., 2024) for evaluation. For generative tasks, we measure generation throughput (prefilling and decoding) with vLLM backend. For non-generative tasks, we measure classification throughput (prefilling only) using HF Transformers.

We consider two GPU environments:

- **AMD MI250X 64GB** (BF16, single GPU, batch size set as 1)
- **NVIDIA A100 40GB** (BF16, single GPU, batch size set as 1)

For detailed throughput performance, Appendix B shows statistics on both GPU environments. They demonstrate that in general, NVIDIA A100 consistently achieves higher throughput than AMD MI250X across both generative and non-generative tasks. Besides, this gap may also reflect the different backends between vLLM and HF Transformers. We further observe that scripts such as Devanagari or Amharic (`amh_Ethi`) often have lower throughput, potentially due to more complex tokenization. Lastly, summarization tasks like XLSum typically involve longer inputs and outputs than sentence-level translation tasks (e.g., FLORES-200), which increases the prefilling overhead and thus reduces the overall tokens/s.

4.2 Case Study on Multilingual Translation

To further illustrate GlotEval's capabilities, we conducted a detailed case study comparing EMMA-500 (Ji et al., 2024), a large-scale multilingual language model designed to enhance multilingual performance, with the base Llama-2-7B model (Touvron et al., 2023) across various multilingual translation scenarios. This study aimed to investigate performance differences under different prompting strategies and diverse language-centric translation tasks. We designed a factorial experiment with the following variables:

- **Models**: EMMA-500 vs. Llama-2-7B
- **Prompting strategies**: multilingual prompting (source language-specific), Chinese prompting (zho_Hans), Finnish prompting (fin_Latn), and English prompting (eng_Latn)
- **Translation directions**: six configurations with different central languages (X→eng-US, eng-US→X, zho-CN→X, X→zho-CN, fin→X, X→fin)

A demonstration of prompt templates is shown in Table 2. For evaluation, we utilized NTREX-128, a multi-aligned benchmark containing parallel texts across 128 languages, which is supported in GlotEval. In the multilingual prompting condition, we used built-in prompt builder in GlotEval, with the support of Microsoft Translator service, to automatically translate prompts into 134 languages supported by their platform. In our case study, 106 of these languages overlap with NTREX-128 languages, allowing us to test performance across this diverse language set.

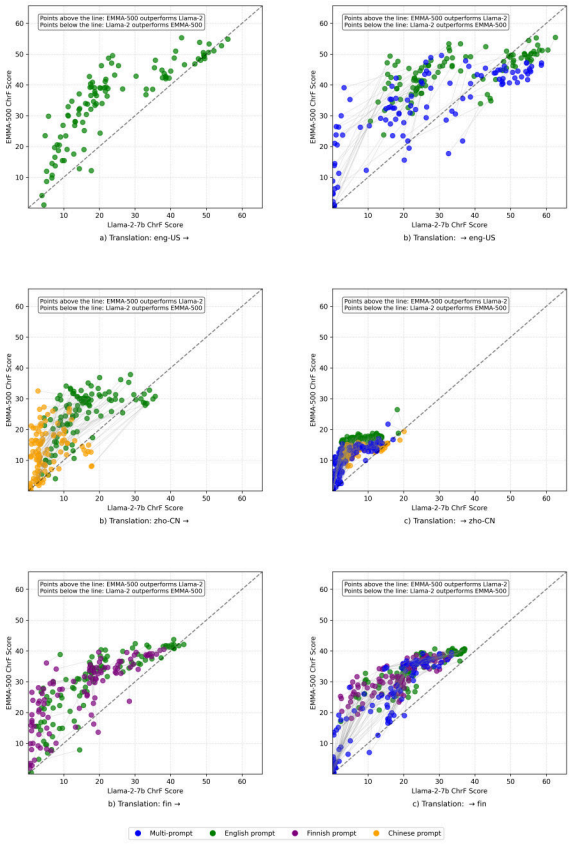


Figure 4: ChrF scores for different translation directions comparing EMMA-500 and Llama-2-7B across four prompting strategies.

The results of our case study (Figure 4) clearly demonstrate EMMA-500's performance compared to Llama-2-7B in multilingual instruction follow-

Prompt Strategy	Tested Translation Language Pair	Prompt Template
`multi` language-specific	`fra → fin` French → Finnish	Traduisez la phrase suivante de **Langue française** en **Langue finnoise** [Langue française] : {source_text_in_finnish} [Langue finnoise] :
`fin_Latn` Finnish	`vie → zho-CN` Vietnamese → Chinese (Simplified)	Käännä seuraava lause **Vietnamin kieli** muotoon **Kiinan kieli (yksinkertaistettu)** [Vietnamin kieli]: {source_text_in_vietnamese} [Kiinan kieli (yksinkertaistettu)]:

Table 2: A demonstration of prompt templates of translation tasks in different prompt strategies.

ing capabilities and non-English-centric translation tasks. Specifically, EMMA-500 shows consistently higher ChrF scores across most language pairs for all six translation directions. This performance advantage is particularly pronounced when using non-English prompting strategies, highlighting EMMA-500's enhanced ability to process and respond to instructions in diverse languages.

The experimental design was implemented using GlotEval, which facilitated the systematic manipulation of variables through simple configuration settings. By simply modifying the prompting strategy parameter and central language settings in the multi-aligned MT benchmark configuration, we are able to comprehensively assess the language models' multilingual capabilities, including both instruction following and non-English-centric multilingual translation.

5 Conclusion and Future Work

In this work, we introduced GlotEval, a lightweight yet comprehensive framework for massively multilingual evaluation of LLMs. By supporting consistent multilingual benchmarking, incorporating language-specific prompt templates, and supporting flexible non-English-centric translation setups, GlotEval enables consistent assessments of LLMs in diverse linguistic contexts—including low-resource settings often neglected by traditional benchmarks. Our case study on multilingual machine translation with two LLMs illustrates the utility of GlotEval in revealing the strengths and weaknesses of multilingual LLMs and in identifying directions for future optimization. Overall, GlotEval aims to encourage more inclusive, transparent, and holistic evaluations of language models across a wide array of languages and tasks, thereby advancing robust multilingual NLP research.

As for future work, we plan to integrate more diverse and comprehensive multilingual benchmarks to better evaluate LLM performance. Plus, we will explore the integration of benchmarks that the synergistic combination of automatic and human evaluation; for example, this could achieved through our pilot development of a lightweight web interface that supports crowd-sourced and expert-driven evaluation to supplement the automatic evaluation.[6]

Ethical Considerations and Broader Impact

Ethical Considerations We strive to uphold the principles outlined in the ACL Code of Ethics. While GlotEval advances multilingual evaluation, several limitations remain. Many benchmarks still lack sufficient or high-quality data for truly low-resource languages, potentially skewing performance assessments. Additionally, as noted by Joshi et al. (2025), existing datasets often inherit cultural and linguistic biases, favoring dominant dialects or standardized language forms over regional or marginalized variants. Computational costs further constrain accessibility: large-scale evaluations are resource-intensive, posing barriers for smaller research teams. More critically, reference-free metrics introduce inherent biases, as they effectively pit one generative model against another (Deutsch et al., 2022). Such metrics struggle to capture fluency, accuracy, or cultural appropriateness, particularly in low-resource contexts where human judgments are essential.

Broader Impact GlotEval promotes equitable progress in NLP by enabling systematic evaluation of large language models (LLMs) across diverse languages. We aim to support researchers and developers in creating language technologies that serve diverse communities more effectively via a more inclusive and holistic evaluation suite.

Acknowledgments

This project is funded by the AI-DOC program hosted by Finnish Center of Artificial Intelligence (decision number VN/3137/2024-OKM-6).

The work has received funding from the European Union's Horizon Europe research and in-

[6]Source code and documentation are available at `https://github.com/MaLA-LM/GlotEval-HumanEval` and `https://gloteval-humaneval.readthedocs.io`

novation programme under grant agreement No 101070350 and from UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant number 10052546], and the Digital Europe Programme under grant agreement No 101195233.

The authors wish to acknowledge CSC - IT Center for Science, Finland, the Leonardo and LUMI supercomputers, owned by the EuroHPC Joint Undertaking, for providing computational resources.

Sawal Devkota, Bhavani Sai Praneeth Varma Mantina, Ananda Sreenidhi, Mengjie Wang, and Samea Yusofi contributed to this project as part of the "Data Analysis Software Project for Natural Language" course at TU Darmstadt, under the guidance of Shaoxiong Ji. This teaching activity was funded by LOEWE Center DYNAMIC as part of the Hessian program for the promotion of cutting-edge research LOEWE under the grant number of LOEWE1/16/519/03/09.001(0009)/98.

References

Idris Abdulmumin, Sthembiso Mkhwanazi, Mahlatse S. Mbooi, Shamsuddeen Hassan Muhammad, Ibrahim Said Ahmad, Neo N. Putini, Miehleketo Mathebula, Matimba Shingange, Tajuddeen Gwadabe, and Vukosi Marivate. 2024. Correcting FLORES evaluation dataset for four African languages. In *Proceedings of the Ninth Conference on Machine Translation*, Miami, USA. Association for Computational Linguistics.

David Adelani, Jesujoba Alabi, Angela Fan, Julia Kreutzer, Xiaoyu Shen, Machel Reid, Dana Ruiter, Dietrich Klakow, Peter Nabende, Ernie Chang, et al. 2022. A few thousand translations go a long way! leveraging pre-trained models for african news translation. pages 3053–3070.

David Ifeoluwa Adelani, Hannah Liu, Xiaoyu Shen, Nikita Vassilyev, Jesujoba O. Alabi, Yanke Mao, Haonan Gao, and En-Shiun Annie Lee. 2024. SIB-200: A simple, inclusive, and big evaluation dataset for topic classification in 200+ languages and dialects. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 226–245, St. Julian's, Malta. Association for Computational Linguistics.

Kabir Ahuja, Harshita Diddee, Rishav Hada, Millicent Ochieng, Krithika Ramesh, Prachi Jain, Akshay Nambi, Tanuja Ganu, Sameer Segal, Mohamed Ahmed, Kalika Bali, and Sunayana Sitaram. 2023. MEGA: Multilingual evaluation of generative AI. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4232–4267, Singapore. Association for Computational Linguistics.

AI4Bharat, Jay Gala, Pranjal A. Chitale, Raghavan AK, Sumanth Doddapaneni, Varun Gumma, Aswanth Kumar, Janki Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, Pratyush Kumar, Mitesh M. Khapra, Raj Dabre, and Anoop Kunchukuttan. 2023. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages.

Felermino Dario Mario Ali, Henrique Lopes Cardoso, and Rui Sousa-Silva. 2024. Expanding FLORES+ benchmark for more low-resource settings: Portuguese-Emakhuwa machine translation evaluation. In *Proceedings of the Ninth Conference on Machine Translation*, Miami, USA. Association for Computational Linguistics.

Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federmann, Dmitriy Genzel, Franscisco Guzmán, Junjie Hu, Macduff Hughes, Philipp Koehn, et al. 2020. Tico-19: the translation initiative for covid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*.

Laurent Bié, Aleix Cerdà-i Cucó, Hans Degroote, Amando Estela, Mercedes García-Martínez, Manuel Herranz, Alejandro Kohan, Maite Melero, Tony O'Dowd, Sinéad O'Gorman, Mārcis Pinnis, Roberts Rozis, Riccardo Superbo, and Artūrs Vasiļevskis. 2020. Neural translation for the European Union (NTEU) project. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 477–478, Lisboa, Portugal. European Association for Machine Translation.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3).

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Ona de Gibert, Robert Pugh, Ali Marashian, Raul Vazquez, Abteen Ebrahimi, Pavel Denisov, Enora Rice, Edward Gow-Smith, Juan C. Prieto, Melissa Robles, Rubén Manrique, Oscar Moreno Veliz, Ángel Lino Campos, Rolando Coto-Solano, Aldo Alvarez, Marvin Agüero-Torales, John E. Ortega, Luis Chiruzzo, Arturo Oncevay, Shruti Rijhwani, Katharina von der Wense, and Manuel Mager. 2025. Findings of the AmericasNLP 2025 Shared Tasks on Machine Translation, Creation of Educational Material, and Translation Metrics for Indigenous Languages of the Americas. In *Proceedings of the 5th Workshop on Natural Language Processing for Indigenous Languages of the Americas (AmericasNLP 2025)*, Albuquerque, New Mexico. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal dependencies. *Computational Linguistics*, 47(2):255–308.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Daniel Deutsch, Rotem Dror, and Dan Roth. 2022. On the limitations of reference-free evaluations of generated text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10960–10977, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Moussa Doumbouya, Baba Mamadi Diané, Solo Farabado Cissé, Djibrila Diané, Abdoulaye Sow, Séré Moussa Doumbouya, Daouda Bangoura, Fodé Moriba Bayo, Ibrahima Sory 2. Condé, Kalo Mory Diané, Chris Piech, and Christopher Manning. 2023. Machine translation for nko: Tools, corpora, and baseline results. In *Proceedings of the Eighth Conference on Machine Translation*, pages 312–343, Singapore. Association for Computational Linguistics.

Christian Federmann, Tom Kocmi, and Ying Xin. 2022. NTREX-128 – news test references for MT evaluation of 128 languages. In *Proceedings of the First Workshop on Scaling Up Multilingual Evaluation*, pages 21–24, Online. Association for Computational Linguistics.

Clémentine Fourrier, Nathan Habib, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Jay P Gala, Pranjal A Chitale, AK Raghavan, Varun Gumma, Sumanth Doddapaneni, Kumar M Aswanth, Janki Atul Nawale, Anupama Sujatha, Ratish Puduppully, Vivek Raghavan, et al. 2023. Indictrans2: Towards high-quality and accessible machine translation models for all 22 scheduled indian languages. *Transactions on Machine Learning Research*, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. 2023. A framework for few-shot language model evaluation. Zenodo.

Isai Gordeev, Sergey Kuldin, and David Dale. 2024. Flores+ translation and machine translation evaluation for the Erzya language. In *Proceedings of the Ninth Conference on Machine Translation*, Miami, USA. Association for Computational Linguistics.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6098–6111, Hong Kong, China. Association for Computational Linguistics.

Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. 2021. XL-sum: Large-scale multilingual abstractive summarization for 44 languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4693–4703, Online. Association for Computational Linguistics.

Chaoqun He, Renjie Luo, Shengding Hu, Yuanqian Zhao, Jie Zhou, Hanghao Wu, Jiajie Zhang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2024. Ultraeval: A lightweight platform for flexible and comprehensive evaluation for llms. *Preprint*, arXiv:2404.07584.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Xu Huang, Wenhao Zhu, Hanxu Hu, Conghui He, Lei Li, Shujian Huang, and Fei Yuan. 2025. Benchmax: A comprehensive multilingual evaluation suite for large language models. *Preprint*, arXiv:2502.07346.

Shaoxiong Ji, Zihao Li, Indraneil Paul, Jaakko Paavola, Peiqin Lin, Pinzhen Chen, Dayyán O'Brien, Hengyu Luo, Hinrich Schütze, Jörg Tiedemann, and Barry Haddow. 2024. EMMA-500: Enhancing massively multilingual adaptation of large language models. *arXiv preprint arXiv:2409.17892*.

Aditya Joshi, Raj Dabre, Diptesh Kanojia, Zhuang Li, Haolan Zhan, Gholamreza Haffari, and Doris Dippold. 2025. Natural language processing for dialects of a language: A survey. *ACM Comput. Surv.*, 57(6).

Amir Hossein Kargaran, François Yvon, and Hinrich Schütze. 2024. GlotScript: A resource and tool for low resource writing system identification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7774–7784, Torino, Italia. ELRA and ICCL.

Ali Kuzhuget, Airana Mongush, and Nachyn-Enkhedorzhu Oorzhak. 2024. Enhancing Tuvan language resources through the FLORES dataset. In *Proceedings of the Ninth Conference on Machine Translation*, Miami, USA. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

Chunlan Ma, Ayyoob ImaniGooghari, Haotian Ye, Renhao Pei, Ehsaneddin Asgari, and Hinrich Schütze. 2024. Taxi1500: A multilingual dataset for text classification in 1500 languages. *arXiv preprint arXiv:2305.08487*.

Mukhammadsaid Mamasaidov and Abror Shopulatov. 2024. Open Language Data Initiative: Advancing low-resource machine translation for Karakalpak. In *Proceedings of the Ninth Conference on Machine Translation*, Miami, USA. Association for Computational Linguistics.

Thomas Mayer and Michael Cysouw. 2014. Creating a massively parallel Bible corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3158–3163, Reykjavik, Iceland. European Language Resources Association.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loïc Barrault, Gabriel M. Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik R. Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2024. Scaling neural machine translation to 200 languages. *Nature*, 630(8018):841–846.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Juan Antonio Perez-Ortiz, Felipe Sánchez-Martínez, Víctor M. Sánchez-Cartagena, Miquel Esplà-Gomis, Aaron Galiano Jimenez, Antoni Oliver, Claudi Aventín-Boya, Alejandro Pardos, Cristina Valdés, Jusép Loís Sans Socasau, and Juan Pablo Martínez. 2024. Expanding the flores+ multilingual benchmark with translations for Aragonese, Aranese, Asturian, and Valencian. In *Proceedings of the Ninth Conference on Machine Translation*, Miami, USA. Association for Computational Linguistics.

Wessel Poelman and Miryam de Lhoneux. 2024. The roles of english in evaluating multilingual language models. *Preprint*, arXiv:2412.08392.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

Shivalika Singh, Angelika Romanou, Clémentine Fourrier, David Ifeoluwa Adelani, and *et al.* 2024a. Global MMLU: Understanding and addressing cultural and linguistic biases in multilingual evaluation. *arXiv preprint arXiv:2412.03304*.

Shivalika Singh, Freddie Vargas, Daniel D'souza, Börje Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura O'Mahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergun, Ifeoma Okoh, Aisha Alaagib, Oshan Mudannayake, Zaid Alyafeai, Vu Chien, Sebastian Ruder, Surya Guthikonda, Emad Alghamdi, Sebastian Gehrmann, Niklas Muennighoff, Max Bartolo, Julia Kreutzer, Ahmet Üstün, Marzieh Fadaee, and Sara Hooker. 2024b. Aya dataset: An open-access collection for multilingual instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11521–11567, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoqing Ellen Tan, Prangthip Hansanti, Carleigh Wood, Bokai Yu, Christophe Ropers, and Marta R. Costa-jussà. 2024. Towards massive multilingual holistic bias. *Preprint*, arXiv:2407.00486.

Jörg Tiedemann. 2020. The tatoeba translation challenge – realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182, Online. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal 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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Daniel Varab and Natalie Schluter. 2021. MassiveSumm: a very large-scale, very multilingual, news summarisation dataset. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10150–10161, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Hongjian Yu, Yiming Shi, Zherui Zhou, and Christopher Haberland. 2024. Machine translation evaluation benchmark for Wu. In *Proceedings of the Ninth Conference on Machine Translation*, Miami, USA. Association for Computational Linguistics.

Shimao Zhang, Changjiang Gao, Wenhao Zhu, Jiajun Chen, Xin Huang, Xue Han, Junlan Feng, Chao Deng, and Shujian Huang. 2024. Getting more from less: Large language models are good spontaneous multilingual learners. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8037–8051. Association for Computational Linguistics.

A Benchmark Settings

A.1 Intrinsic Evaluation

Given the input $X = (x_0, x_1, \ldots, x_{n_t})$, the negative log-likelihood (NLL) is defined as:

$$\text{NLL} = -\sum_{i=1}^{n_t} \log p_\theta(x_i|x_{<i}) \tag{1}$$

while perplexity (PPL) is computed as:

$$\text{PPL} = \exp\{-\frac{1}{n_t}\sum_{i=1}^{n_t} \log p_\theta(x_i|x_{<i})\} \tag{2}$$

Intuitively, PPL evaluates a model's ability to predict tokens in a given corpus, with lower values indicating better performance. In contrast, NLL measures the overall likelihood of the corpus under the model. Notably, due to its length normalization, PPL is directly influenced by the tokenization scheme, whereas NLL remains unaffected. Therefore, we use NLL for model comparisons to ensure consistency across models with different tokenization methods.

We compute NLL by concatenating the input sentences and applying a strided sliding window of size 1024.

A.2 Machine Translation

FLORES+ This work builds upon previous efforts on multilingual machine translation and evaluation datasets (NLLB Team et al., 2024; Goyal et al., 2022; Guzmán et al., 2019; Doumbouya et al., 2023; AI4Bharat et al., 2023; Perez-Ortiz et al., 2024; Abdulmumin et al., 2024; Ali et al., 2024; Kuzhuget et al., 2024; Yu et al., 2024; Mamasaidov and Shopulatov, 2024; Gordeev et al., 2024).

AmericasNLP Only the development set is used, as the test set is not disclosed. Note that this dataset is aligned with Spanish, but not English.

Tatoeba (v2023-09-26) We keep only test sets with over 1,000 sentences.

BLEU In our experiments, BLEU scores are computed via `SacreBLEU` (Post, 2018) with the `flores200` tokenizer to quantify translation quality. The BLEU signature is:

```
nrefs:1 | case:mixed | eff:no | tok:flores200 | smooth:exp | version:2.4.2
```

COMET Users can specify the customized model in the configuration file. The default model is `Unbabel/wmt22-comet-da`.

ChrF with Gender ChrF with gender is an evaluation metric that calculates the standard chrF score separately for sentences marked with different grammatical genders (masculine and feminine). By comparing these scores, one can assess whether a translation system favors one gender form over the other, thereby revealing potential gender bias in its outputs.

A.3 Text Classification

In classification tasks, the model predicts by ranking logits for each category; candidate labels are tokenized, and the label corresponding to the token with the highest probability is selected.

B Throughput Statistics

GlotEval provides a uniform pipeline for measuring both decoding-heavy and classification-style tasks across different languages, scripts, and hardware setups. According to efficiency analysis conducted in section 4.1, table 3 and 4 show throughput results on both NVIDIA A100 40GB and AMD MI250X 64GB GPU environments.

Language	FLORES-200(Eng-X) (3-shot)	Aya (0-shot)	XLSum (0-shot)	SIB-200 (3-shot)	Global-MMLU (0-shot)	WikiANN (3-shot)
French (fra_Latn)	854 / 0.88 = 969.55	447 / 0.77 = 583.55	67 / 0.09 = 720.32	10 / 0.53 = 18.88	10 / 0.27 = 36.17	70 / 2.36 = 29.60
Swahili (swa_Latn)	1174 / 0.92 = 1274.74	812 / 0.80 = 1020.78	150 / 0.56 = 268.32	10 / 0.56 = 17.71	10 / 0.31 = 32.65	61 / 1.97 = 30.91
Vietnamese (vie_Latn)	1206 / 0.92 = 1304.01	443 / 0.76 = 581.40	172 / 0.74 = 233.62	10 / 0.48 = 20.99	10 / 0.26 = 37.87	74 / 2.41 = 30.66
Indonesian (ind_Latn)	776 / 0.87 = 893.11	259 / 0.75 = 346.56	308 / 0.75 = 411.16	10 / 0.53 = 18.91	10 / 0.28 = 35.65	54 / 2.02 = 26.79
Latin Scri.	4010 / 3.59 = 1116.99	1961 / 3.08 = 636.69	697 / 2.14 = 325.70	40 / 2.10 = 19.05	40 / 1.12 = 35.71	259 / 8.76 = 29.57
Kyrgyz (kir_Cyrl)	1174 / 0.93 = 1259.10	436 / 0.76 = 573.19	324 / 0.75 = 429.72	10 / 0.72 = 13.95	10 / 0.35 = 28.98	72 / 4.20 = 17.16
Russian(rus_Cyrl)	1280 / 1.86 = 688.45	551 / 0.77 = 712.23	339 / 0.67 = 507.16	10 / 0.53 = 18.92	10 / 0.28 = 35.25	71 / 3.51 = 20.20
Serbian (srp_Cyrl)	1118 / 0.92 = 1207.56	475 / 0.76 = 621.45	342 / 0.76 = 452.07	10 / 0.62 = 16.25	10 / 0.30 = 33.14	48 / 1.94 = 24.76
Ukrainian (ukr_Cyrl)	1083 / 0.91 = 1191.05	404 / 0.76 = 532.91	43 / 0.09 = 470.72	10 / 0.68 = 14.65	10 / 0.31 = 31.68	132 / 7.43 = 17.78
Cyrillic Scri.	4655 / 4.62 = 1007.58	1866 / 3.05 = 611.80	1048 / 2.27 = 461.67	40 / 2.55 = 15.69	40 / 1.24 = 32.26	323 / 17.08 = 18.91
Arabic (arb_Arab)	852 / 0.87 = 974.46	74 / 0.41 = 181.59	228 / 1.62 = 140.36	10 / 0.53 = 18.85	10 / 0.28 = 36.32	76 / 2.75 = 27.65
Persian (fas_Arab)	852 / 0.89 = 958.99	264 / 0.75 = 353.62	333 / 0.76 = 440.70	10 / 0.68 = 14.63	10 / 0.31 = 31.74	54 / 12.48 = 4.32
Arabic Scri.	1704 / 1.76 = 968.18	338 / 1.16 = 291.38	561 / 2.38 = 235.71	20 / 1.21 = 16.53	20 / 0.59 = 33.90	130 / 15.23 = 8.54
Bengali (ben_Beng)	1143 / 0.96 = 1190.74	973 / 0.81 = 1195.97	260 / 0.71 = 366.53	10 / 1.26 = 7.91	10 / 0.45 = 21.99	39 / 2.13 = 18.32
Hindi (hin_Deva)	1167 / 0.96 = 1210.17	960 / 0.81 = 1182.68	223 / 0.75 = 296.00	10 / 1.10 = 9.07	10 / 0.39 = 25.62	52 / 2.50 = 20.79
Nepali (npi_Deva)	1250 / 1.01 = 1247.45	803 / 0.80 = 1009.63	231 / 0.60 = 384.25	10 / 1.02 = 9.78	10 / 0.41 = 24.57	69 / 3.98 = 17.32
Devanagari	3560 / 2.93 = 1215.02	2736 / 2.42 = 1130.58	714 / 2.06 = 346.60	30 / 3.38 = 8.88	30 / 1.25 = 24.00	160 / 8.61 = 18.58
Sinhala (sin_Sinh)	1280 / 1.04 = 1226.21	1280 / 0.86 = 1485.77	103 / 0.17 = 601.67	10 / 1.57 = 6.38	10 / 0.52 =19.38	69 / 5.43 = 12.70
Telugu (tel_Telu)	1208 / 1.02 = 1188.70	559 / 0.80 = 697.54	74 / 0.14 = 537.25	10 / 1.57 = 6.38	10 / 0.55 = 18.21	71 / 8.01 = 8.86
Amharic (amh_Ethi)	1280 / 1.00 = 1278.40	1280 / 0.85 = 1498.47	65 / 0.09 = 700.75	10 / 1.00 = 9.95	10 / 7.37 = 1.36	53 / 10.31 = 5.14
Japanese (jpn_Jpan)	714 / 0.87 = 820.25	152 / 0.21 = 707.20	274 / 0.75 = 365.11	10 / 0.48 = 21.01	10 / 0.28 = 35.99	389 / 33.70 = 11.54
Korean (kor_Hang)	1016 / 0.90 = 1129.38	284 / 0.76 = 374.84	59 / 0.12 = 493.29	10 / 0.54 = 18.58	10 / 0.27 = 36.78	91 / 5.20 = 17.50
Chinese (zho_Hans)	676 / 0.87 = 780.69	403 / 0.62 = 651.94	59 / 0.12 = 491.30	10 / 0.41 = 24.11	10 / 0.26 =37.60	419 / 42.26 = 9.91

Table 3: Throughput with NVIDIA A100 40GB GPU. Each cell contains: $\frac{\#generated\ tokens}{wall\ time\ (seconds)} = average\ tokens/s$.

Language	FLORES-200(Eng-X) (3-shot)	Aya (0-shot)	XLSum (0-shot)	SIB-200 (3-shot)	Global-MMLU (0-shot)	WikiANN (3-shot)
French (fra_Latn)	800 / 1.53 = 524.33	409 / 1.34 = 304.24	164 / 1.01 = 161.69	10 / 29.00 = 0.34	10 / 39.30 = 0.25	70 / 38.18 = 1.83
Swahili (swa_Latn)	1039 / 1.55 = 670.79	136 / 0.43 = 317.94	226 / 0.93 = 244.26	10 / 26.71 = 0.37	10 / 38.61 = 0.26	61 / 38.21 = 1.60
Vietnamese (vie_Latn)	932 / 1.53 = 608.26	675 / 1.39 = 485.18	58 / 0.15 = 379.43	10 / 31.58 = 0.32	10 / 39.46 = 0.25	74 / 38.13 = 1.94
Indonesian (ind_Latn)	1076 / 1.52 = 706.44	779 / 1.40 = 555.64	262 / 1.29 = 203.48	10 / 29.33 = 0.34	10 / 39.14 = 0.26	54 / 37.16 = 1.45
Latin Scri.	3847 / 6.13 = 627.57	1999 / 4.56 = 438.38	710 / 3.38 = 210.06	40 / 29.20 = 1.37	40 / 39.22 = 1.02	259 / 37.98 = 6.82
Kyrgyz (kir_Cyrl)	1051 / 1.57 = 669.63	344 / 1.32 = 261.10	444 / 1.36 = 325.96	10 / 17.18 = 0.58	10 / 37.68 = 0.27	72 / 23.48 = 3.07
Russian(rus_Cyrl)	1280 / 1.86 = 686.47	442 / 1.37 = 322.04	243 / 1.03 = 234.98	10 / 29.37 = 0.34	10 / 39.04 = 0.26	71 / 30.55 = 2.32
Serbian (srp_Cyrl)	1210 / 1.58 = 767.56	560 / 1.38 = 406.04	261 / 1.17 = 222.39	10 / 19.57 = 0.51	10 / 38.61 = 0.26	48 / 36.64 = 1.31
Ukrainian (ukr_Cyrl)	939 / 1.55 = 607.67	378 / 1.33 = 284.88	103 / 0.42 = 244.48	10 / 17.34 = 0.58	10 / 38.31 = 0.26	132 / 23.54 = 5.61
Cyrillic Scri.	4480 / 6.56 = 682.93	1724 / 5.40 = 319.26	1051 / 3.98 = 264.07	40 / 19.90 = 2.01	40 / 38.10 = 1.05	323 / 26.24 = 12.31
Arabic (arb_Arab)	919 / 1.54 = 595.36	160 / 1.24 = 129.25	83 / 0.26 = 318.11	10 / 29.06 = 0.34	10 / 39.23 = 0.25	76 / 37.56 = 2.02
Persian (fas_Arab)	929 / 1.55 = 600.20	16 / 0.12 = 131.16	184 / 1.21 = 152.61	10 / 17.43 = 0.57	10 / 38.25 = 0.26	54 / 13.24 = 4.07
Arabic Scri.	1848 / 3.09 = 598.06	176 / 1.36 = 129.41	267 / 1.47 = 181.63	20 / 21.98 = 0.91	20 / 39.22 = 0.51	130 / 21.35 = 6.09
Bengali (ben_Beng)	1130 / 1.62 = 698.59	1026 / 1.40 = 734.29	178 / 1.21 = 147.66	10 / 11.17 = 0.90	10 / 27.49 = 0.36	39 / 28.34 = 1.38
Hindi (hin_Deva)	1160 / 1.62 = 714.58	650 / 1.41 = 462.11	186 / 1.21 = 154.24	10 / 12.17 = 0.82	10 / 34.96 = 0.29	52 / 31.38 = 1.66
Nepali (npi_Deva)	1280 / 1.66 = 768.85	1126 / 1.40 = 805.59	275 / 1.10 = 250.46	10 / 13.00 = 0.77	10 / 34.68 = 0.29	69 / 2.77 = 24.87
Devanagari	3570 / 4.90 = 728.57	2802 / 4.21 = 665.56	639 / 3.52 = 181.53	30 / 12.05 = 2.49	30 / 31.91 = 0.94	160 / 5.73 = 27.91
Sinhala (sin_Sinh)	1280 / 1.76 = 727.47	1223 / 1.42 = 858.97	140 / 0.93 = 151.08	10 / 9.15 = 1.09	10 / 25.60 = 0.39	69 / 15.70 = 4.40
Telugu (tel_Telu)	1280 / 1.73 = 737.77	507 / 1.45 = 348.78	198 / 0.92 = 214.92	10 / 9.14 = 1.09	10 / 24.87 = 0.40	71 / 11.99 = 5.92
Amharic (amh_Ethi)	1280 / 1.66 = 772.22	1153 / 1.40 = 821.85	211 / 1.12 = 189.18	10 / 13.04 = 0.77	10 / 34.30 = 0.29	53 / 32.18 = 1.65
Japanese (jpn_Jpan)	690 / 1.51 = 458.09	266 / 1.27 = 209.63	250 / 1.02 = 244.87	10 / 31.85 = 0.31	10 / 39.23 = 0.25	389 / 14.98 = 25.96
Korean (kor_Hang)	973 / 1.53 = 633.96	468 / 1.38 = 340.21	204 / 1.07 = 191.09	10 / 28.74 = 0.35	10 / 39.33 = 0.25	91 / 25.01 = 3.64
Chinese (zho_Hans)	823 / 1.52 = 540.58	248 / 1.00 = 248.61	109 / 0.39 = 276.83	10 / 35.54 = 0.28	10 / 39.37 = 0.25	419 / 13.88 = 30.20

Table 4: Throughput with AMD MI250X 64GB GPU. Each cell contains: $\frac{\#generated\ tokens}{wall\ time\ (seconds)} = average\ tokens/s$.

MASA: LLM-Driven Multi-Agent Systems for Autoformalization

Lan Zhang[1], Marco Valentino[2], André Freitas[1,3,4]
[1]Department of Computer Science, University of Manchester, United Kingdom
[2]School of Computer Science, University of Sheffield, United Kingdom
[3]Idiap Research Institute, Switzerland
[4]National Biomarker Centre, CRUK Manchester Institute, United Kingdom
lan.zhang-6@postgrad.manchester.ac.uk
m.valentino@sheffield.ac.uk andre.freitas@idiap.ch

Abstract

Autoformalization serves a crucial role in connecting natural language and formal reasoning. This paper presents MASA, a novel framework for building multi-agent systems for autoformalization driven by Large Language Models (LLMs). MASA leverages collaborative agents to convert natural language statements into their formal representations. The architecture of MASA is designed with a strong emphasis on modularity, flexibility, and extensibility, allowing seamless integration of new agents and tools to adapt to a fast-evolving field. We showcase the effectiveness of MASA through use cases on real-world mathematical definitions and experiments on formal mathematics datasets. This work highlights the potential of multi-agent systems powered by the interaction of LLMs and theorem provers in enhancing the efficiency and reliability of autoformalization, providing valuable insights and support for researchers and practitioners in the field.[1]

1 Introduction

Mathematical reasoning has gained attention in natural language processing (Ferreira and Freitas, 2020; Welleck et al., 2021; Ferreira et al., 2022; Valentino et al., 2022; Mishra et al., 2022b; Meadows and Freitas, 2023; Petersen et al., 2023) and emerged as a core ability of interest in the era of Large Language Models (LLMs) (Wei et al., 2022; Wang et al., 2023; Meadows et al., 2024; Lu et al., 2023; Mishra et al., 2022a). However, most research on LLM-based mathematical reasoning primarily relies on statements expressed in natural language, resulting in a reasoning process that is neither systematic, transparent, rigorous, nor robust. In contrast, formal mathematics is grounded in a logic-based formal language, where each reasoning step can be systematically verified using theorem provers such as Isabelle (Paulson, 2000) or Lean (de Moura et al., 2015).

Although formal mathematics can enhance the systematicity, transparency and rigor of the underlying reasoning process, verifiable mathematical reasoning requires translating natural language representations into formal logical formulas – a task that demands considerable effort and domain-specific knowledge when done manually. Autoformalization (Wu et al., 2022), which aims to automate this process, has shown promising results through prompting with LLMs (Brown et al., 2020). However, addressing real-world autoformalization problems remains a challenging task that can hardly be solved through a single monolithic LLM architecture (Zhang et al., 2025). This highlights the urgent need to develop a multi-component, distributed system for this task.

Existing implementations of LLM-based autoformalization systems (Wu et al., 2022; Jiang et al., 2023; Zhang et al., 2024; Li et al., 2024) often lack modularity, flexibility, and extensibility, which hinders researchers from effectively developing and extending systems involving multiple interacting components. In this demonstration paper, we aim to bridge this gap by proposing MASA, a framework that supports the construction of modular agents for building multi-agent systems for autoformalization. Experiments using MASA to build and orchestrate multiple agents demonstrate that the framework is flexible, extensible, and capable of providing valuable insights into multi-agent autoformalization processes.

Our contributions can be summarized as follows:

1. We introduce a modular framework for building multi-agent systems for autoformalization, offering flexibility and extensibility for system development.

2. We showcase the formalization of real-world mathematics using agents, highlighting the

[1]Code and data are available at: https://github.com/lanzhang128/multi_agent_autoformalization

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 615–624
November 4-9, 2025

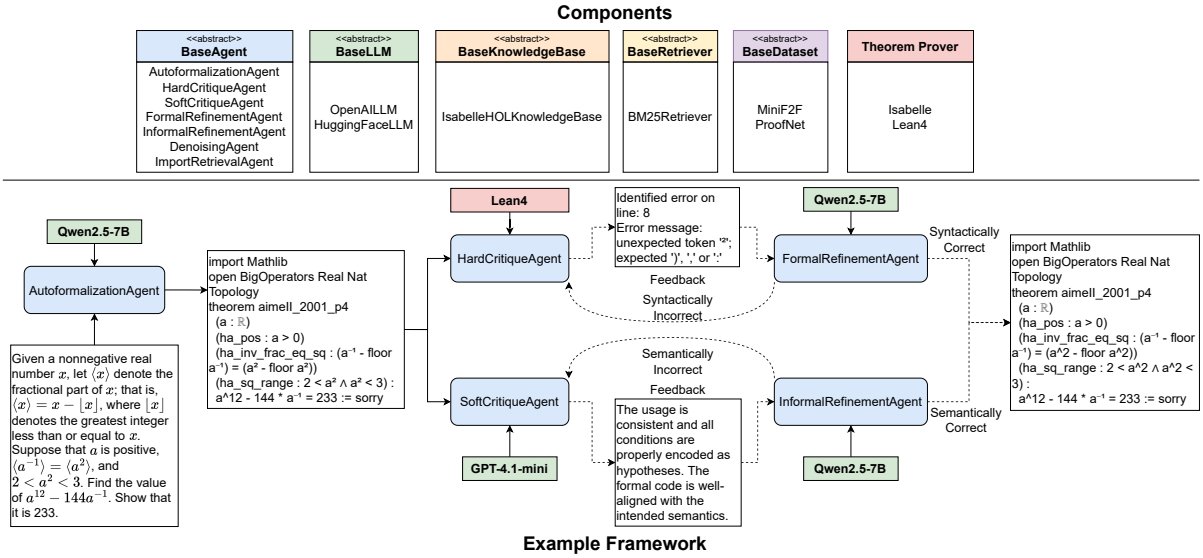


Figure 1: A schematic illustration of MASA. The system comprises agent, LLM, knowledge base, retriever, and theorem prover. The example framework depicts a fully automated pipeline demonstrating how these components interact to formalize a mathematical statement from miniF2F into Lean4.

practical potential of our framework.

3. We evaluate our framework in three multi-agent settings, demonstrating its effectiveness and its ability to provide valuable insights into multi-agent autoformalization processes. Our final iterative self-refinement system achieves 35.25% and 61.89% formalizations that are both syntactically correct and semantically aligned when applying on Qwen2.5-7B and GPT-4.1-mini, respectively.

2 Components in MASA

Autoformalization can be defined as an automatic transformation function $\mathcal{A}$ which maps a natural language statement s to its formal representation $\phi = \mathcal{A}(s)$. A typical approach to this task involves using large language models (LLMs) via prompting (Wu et al., 2022), where the function is defined as $\mathcal{A} = \text{LLM}(\text{prompt})$. However, vanilla prompting does not fully exploit various factors that could aid the autoformalization process. A multi-agent setting can incorporate such factors to produce better formalizations.

To this end, MASA provides support for designing and implementing components that are potentially beneficial for building a multi-agent autoformalization system (Figure 1). We outline each component as follows:

Agent: The agent is the basic element that performs specific and disentangled functionalities in an interactive manner during the autoformalization process. For autoformalization, the core agents should have the ability to perform tasks such as few-shot autoformalization, providing critiques on formal code, and refining the formalization based on those critiques. Agents with these specialised capabilities can be implemented under an abstract "BaseAgent" class, as illustrated in Figure 1.

Large Language Model (LLM): LLMs play a key role in an autoformalization system, providing reasoning and linguistic capabilities for implementing specialized agents. Through prompting, LLMs can translate natural language into machine-verifiable languages, provide feedback on formalizations, refine formal codes, etc. The LLM component is abstracted as "BaseLLM" class with specific implementation of OpenAI models[2] and running local HuggingFace moodels[3] in Figure 1.

Knowledge Base (KB): Formalizations need to align with existing libraries of the formal language. The knowledge base stores information from those libraries so that relevant knowledge can be retrieved to aid an autoformalization process. In Figure 1, the knowledge base is abstracted as "BaseKnowledgeBase" class with an instance of knowledge base of formal statements and proofs from Isabelle/HOL. We provide an example data from the knowledge base in Appendix A.

Retriever: The retriever ranks the relevance of a

[2] https://platform.openai.com/docs/api-reference/chat

[3] https://huggingface.co/docs/transformers/llm_tutorial

formal or informal mathematical statement to data points from a knowledge base of mathematical libraries and retains only the most relevant ones. It is essential for augmenting LLM-generated formalizations. In Figure 1, it is abstracted as "BaseRetriever" class with an implementation of the BM25[4] method.

Theorem Prover (TP): The theorem prover focuses on the syntactic correctness and underlying logic of a formalization within the relevant formal language. It provides precise information about detected errors for flawed formalizations. The current implementation supports Isabelle via its dedicated server[5] and Lean4[6] through REPL[7].

3 A Use Case for Formalizing Real-World Definition with Multiple Agents

Our framework provides a suite of built-in agents that can be orchestrated to implement autoformalization workflows. To showcase such functionality, we explicitly investigate a use case that focuses on formalizing the definition of softmax[8] using multiple agents from MASA.[9]

Definition of Softmax Function. Formally, the standard (unit) softmax function $\sigma\colon \mathbb{R}^K \to (0,1)^K$, where $K \geq 1$, takes a vector $\mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$ and computes each component of vector $\sigma(\mathbf{z}) \in (0,1)^K$ with $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$.

The definition is represented as a string, and we aim to formalize it in formal language Isabelle/HOL[10]:

```
informal = "Definition of Softmax Function:..."
formal_language = "Isabelle/HOL"
```

We first instantiate an OpenAI model using GPT-4o as the sole backend LLM for agents in Figure 1:

```
from llm import OpenAILLM
gpt = OpenAILLM(api_key="...", model="gpt-4o")
```

3.1 Few-Shot Autoformalization Agent

The autoformalization agent supports the generation of formalization code via an LLM, given a natural language mathematical statement and optional informal-formal statement pairs as exemplars. We can instantiate an autoformalization agent and perform zero-shot autoformalization:

```
from agent import AutoformalizationAgent
agent_auto = AutoformalizationAgent(
    llm=gpt, formal_language=formal_language)
zero_formalization, _ = agent_auto(
    informal_statement=informal)
```

to obtain the formalization:

```
definition softmax :: "real list ⇒ real list"
  where "softmax z =
    let exp_z = map exp z;
      sum_exp_z = sum_list exp_z
    in map (λzi. exp zi / sum_exp_z) z"
```

3.2 Critique Agent

The critique agents provide critiques of formalized codes from specific aspects. They are divided into two categories: (1) **Hard**: critiques are produced by the relevant theorem prover to assess the syntactic aspects of the formalization; (2) **Soft**: soft-critique agents follow the concept of LLM-as-a-Judge (Zheng et al., 2023). Our implementation supports LLMs in producing a binary value (True or False) along with a detailed explanation of the judgment, given the description of an aspect.

To check the syntactic correctness of the previous formalization, we instantiate a hard critique agent[11]:

```
from agent import HardCritiqueAgent
agent_hard = HardCritiqueAgent(
    formal_language=formal_language)
correctness, error_details = agent_hard(
    formalization=zero_formalization)
```

we have the critique that the formalization is incorrect with the following detected error detail:

Undefined type name: "real" Failed to parse type

This error indicates that "real" in the formalization does not have a reference. We use a tool agent to try to solve this issue.

3.3 Tool Agent

Tool agents are designed to address common issues in the autoformalization process using non-reasoning methods. Our implementation includes two such agents: one for denoising (Zhang et al.,

[4] https://pypi.org/project/rank-bm25/

[5] https://isabelle.in.tum.de/dist/Isabelle2025/doc/system.pdf

[6] https://github.com/leanprover/lean4

[7] https://github.com/leanprover-community/repl

[8] https://en.wikipedia.org/wiki/Softmax_function

[9] Runnable Python Notebook is available at: https://github.com/lanzhang128/multi_agent_autoformalization/blob/main/example_paper.ipynb

[10] https://isabelle.in.tum.de/dist/library/HOL/index.html

[11] Occasionally, the formalization contains only the main body and cannot be tested without additional formatting and necessary imports. The hard critique agent automatically handles this case by adding the required formatting and the default importing statement.

2024), and another for import retrieval, which tackles the problem of missing relevant imports in formalizations – often leading to errors that items are treated as undefined (Zhang et al., 2025).

For the aforementioned issue in the running case, we can instantiate a tool agent for import retrieval:

```
from agent import ImportRetrievalAgent
agent_imports = ImportRetrievalAgent(
    formal_language=formal_language,
    retriever="bm25")
import_formalization = agent_imports(
    formalization=zero_formalization, top_n=1)
```

The modified formal code becomes:

```
theory Softmax imports Main "HOL.Complex" begin
definition softmax :: "real list ⇒ real list"
  where "softmax z =
    let exp_z = map exp z;
        sum_exp_z = sum_list exp_z
    in map (λzi. exp zi / sum_exp_z) z"
end
```

Using the hard critique agent to test the modified formal code, we obtain the error information:

```
Inner syntax error Failed to parse prop
```

This error indicates syntax errors in the formalization. Refinement agents are thereby required to improve the formal codes.

3.4 Refinement Agent

Refinement agents refine the formalization based on given feedback. Depending on the source of the feedback, they are divided into two categories: (1) **Formal**: the agent refines the formal codes from a syntactic perspective, given the hard critiques from the relevant theorem prover; (2) **Informal**: the agent refines the formal codes based on the soft critiques from an LLM.

In the previous case, the formalization has syntax errors. we instantiate a formal refinement agent to let GPT-4o try to fix the errors:

```
from agent import FormalRefinementAgent
agent_formal = FormalRefinementAgent(
    llm=gpt, formal_language=formal_language)
formal_refinement, _ = agent_formal(
    informal_statement=informal,
    formalization=import_formalization,
    correctness=correctness,
    error_details=error_details)
```

The refined formalization is:

```
theory Softmax imports "HOL.Real" begin
definition softmax :: "real list ⇒ real list"
  where "softmax z =
    let exp_z = map exp z;
        sum_exp_z = listsum exp_z
    in map (λzi. exp zi / sum_exp_z) z"
end
```

While the inner syntax error is still not fixed, GPT-4o improves the imports of the formal code to a more concrete one that directly include the definition of "real".

An alternative process of refinement could focus on the semantic aspect. For example, we can describe the aspect of interest as the following:

```
aspect_description="whether the formalized code
    involves all mathematical concepts in the
    natural language statement."
```

and instantiate a soft critique agent to focus specifically on this aspect:

```
from agent import SoftCritiqueAgent
agent_soft = SoftCritiqueAgent(
    llm=gpt, formal_language=formal_language,
    aspect_description=aspect_description)
aspect_evaluation, _ = agent_soft(
    informal_statement=informal,
    formalization=import_formalization)
```

Although GPT-4o judges this aspect of formalization as correct, we can still instantiate an informal refinement agent to refine the formal code based on the evaluation of the relevant aspect:

```
from agent import InformalRefinementAgent
agent_informal = InformalRefinementAgent(
    llm=gpt, formal_language=formal_language)
informal_refinement, _ = agent_informal(
    informal_statement=informal,
    formalization=import_formalization,
    aspect_description=aspect_description,
    aspect_evaluation=aspect_evaluation)
```

The direct outputs of aspect evaluation and informal refinement are provided in Appendix B.

4 System Evaluation

The core components of MASA include agents for base autoformalization, critique and refinement, large language models (LLMs), and theorem provers. To evaluate our implementation, we construct three practical multi-agent systems that utilize these agents in conjunction with LLMs and theorem provers. As representative LLMs, we use GPT-4.1-mini, a model from the GPT-4 series (OpenAI, 2024), and open-sourced Deepseek-Math (Shao et al., 2024) and Qwen2.5-7B (Qwen-Team, 2025). For benchmarking, we use miniF2F (Zheng et al., 2022), which provides ground-truth formalizations in both Isabelle/HOL and Lean4, and ProofNet(Azerbayev et al., 2023), which provides ground-truth Lean4 code.

A correctly implemented system should produce formalizations that are both syntactically correct and semantically meaningful. For syntactic correctness, we use pass rate as the evaluation metric. For

Algorithm 1 Hard-Critique Formal Refinement

Input: informal statements S, (Optional) informal-formal pairs $\{s, \phi\}$
Instantiate an LLM m_1
Instantiate an autoformalization agent a_{auto} with m_1
Instantiate a hard critique agent a_{hard}
Instantiate an LLM m_2
Instantiate a formal refinement agent a_{formal} with m_2
for $s_i \in S$ **do**
    Formalization $\phi_i = a_{\text{auto}}(s_i, \{s, \phi\})$
    Correctness, Error details $c, e = a_{\text{hard}}(\phi_i)$
    if c is False **then**
        Refinement $\phi_i = a_{\text{formal}}(s_i, \phi_i, c, e)$

AFA	HCA	FRA	BLEU-4	ChrF	RUBY	Pass
miniF2F-Test (Isabelle/HOL)						
G4M+ZS	-	-	26.13	33.83	41.00	65.57
	Isabelle	G4M	21.78	34.35	39.94	77.05
G4M+FS	-	-	32.46	45.32	47.67	76.23
	Isabelle	G4M	25.64	44.28	45.82	86.48
DSM+FS	-	-	6.01	35.14	28.63	29.10
	Isabelle	DSM	3.45	28.54	22.22	29.10
	Isabelle	G4M	6.33	37.28	29.20	36.48
ProofNet-Test (Lean4)						
G4M+ZS	-	-	13.98	35.80	37.34	3.30
	Lean4	G4M	11.06	34.96	34.17	3.85
G4M+FS	-	-	21.35	44.45	43.81	12.09
	Lean4	G4M	18.98	44.00	41.37	14.84
DSM+FS	-	-	6.81	39.67	32.19	8.79
	Lean4	DSM	3.79	34.93	18.66	8.79
	Lean4	G4M	10.08	42.82	32.24	10.99

Table 1: Formalization results across different settings. (**AFA**): Autoformalization Agent; (**HCA**): Hard Critique Agent; (**FRA**): Formal Refinement Agent; (**ZS**): Zero-Shot prompting; (**FS**): Few-Shot prompting (3 exemplars); (**G4M**): GPT-4.1-mini; (**DSM**): Deepseek-Math-7B. All numbers are reported in percentages.

semantic evaluation, there is currently no universally accepted standard. Thus, we adopt BLEU (Papineni et al., 2002), ChrF (Popović, 2015), and RUBY (Tran et al., 2019) as proxy metrics, following the evaluation setup in (Zhang et al., 2024).

Soft-critique agents also can serve as evaluators of semantics for autoformalization. We focus on two aspects of interest for the evaluation: (i) **Alignment Faithfulness (AF)**: Is the formalized code accurately aligned with the intended semantics of natural language statement? (ii) **Formalization Correctness (FC)**: Is the formalized code alone valid, nature and well-formed? We employ GPT-4.1-mini as the backend LLM in soft-critique agents to assess the percentage of formalizations that satisfy the relevant evaluation aspect.

Algorithm 2 Soft-Critique Informal Refinement

Input: informal statements S, (Optional) informal-formal pairs $\{s, \phi\}$, aspect description d
Instantiate an LLM m_1
Instantiate an autoformalization agent a_{auto} with m_1
Instantiate an LLM m_2
Instantiate a soft critique agent a_{soft} with m_2, d
Instantiate an LLM m_3
Instantiate an informal refinement agent a_{informal} with m_3
for $s_i \in S$ **do**
    Formalization $\phi_i = a_{\text{auto}}(s_i, \{s, \phi\})$
    Explanation, Judgment $e, j = a_{\text{soft}}(s_i, \phi_i)$
    if j is False **then**
        Refinement $\phi_i = a_{\text{informal}}(s_i, \phi_i, d, e, j)$

4.1 Hard Critique for Formal Refinement

The hard critique for formal refinement process (Algorithm 1) consists of an autoformalization agent, a hard-critique agent, and a formal refinement agent. Experiments are conducted on miniF2F for Isabelle/HOL and on ProofNet for Lean4. The results are presented in Table 1. The few-shot autoformalization agent produces more syntactically correct formalizations compared to the zero-shot setting. The BLEU, ChrF, and RUBY scores also improve in the few-shot setting, indicating that the formalizations are semantically closer to the ground truth. When the formal refinement agent is applied, both zero-shot and 3-shot formalizations demonstrate improved syntactic correctness with GPT-4.1-mini. Although some metric scores decrease after refinement, this is expected, as optimizing for syntactic correctness does not always align with the ground-truth formalization. Deepseek-Math exhibits limited capability in performing formal refinement. However, the stronger LLM, GPT-4.1-mini, is still able to refine formalizations generated by Deepseek-Math. The resulting metric scores are significantly lower than those obtained using GPT-4.1-mini in the zero-shot setting, suggesting that the formal refinement agent modifies the input formalization rather than rewriting it entirely.

4.2 Soft Critique for Informal Refinement

The soft critique for informal refinement system (Algorithm 2) comprises three components: an autoformalization agent, a soft-critique agent, and an informal refinement agent. The multi-agent system is evaluated on Lean4 using miniF2F. Results are reported in Table 2. Informal refinement agents are capable of improving specific aspects of formalization. For both Deepseek-Math and Qwen2.5-7B, refinement guided by soft-critique feedback targeting a particular aspect often leads to a signif-

AFA	SCA	IRA	AF	FC
DSM+FS	-	-	38.52	47.54
	G4M+AF	DSM	47.95	38.52
	G4M+AF	G4M	90.57	79.92
	G4M+FC	DSM	43.03	52.05
	G4M+FC	G4M	77.05	86.07
Qwen+FS	-	-	54.51	62.70
	G4M+AF	Qwen	73.77	66.39
	G4M+AF	G4M	93.44	85.25
	G4M+FC	Qwen	69.26	79.92
	G4M+FC	G4M	82.79	90.57

Table 2: Lean4 formalization results on miniF2F test set. (**AFA**): Autoformalization Agent; (**SCA**): Soft-Critique Agent; (**IRA**): Informal Refinement Agent; (**FS**): Few-Shot (3 exemplars); (**AF**): Alignment Faithfulness; (**FC**): Formalization Correctness; (**G4M**): GPT-4.1-mini; (**DSM**): Deepseek-Math-7B; (**Qwen**): Qwen2.5-7B. Judges for obtaining scores of AF and FC are based on GPT-4.1-mini. All numbers are reported in percentages.

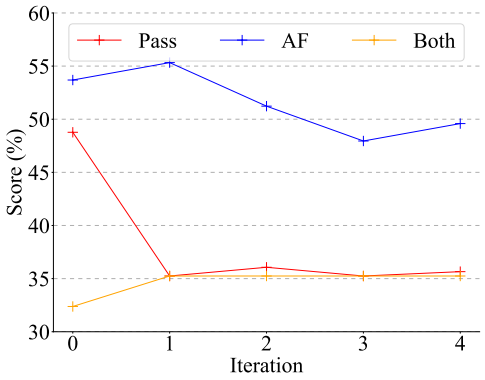
icant improvement in that aspect. In most cases, focusing on one aspect also yields improvements in both aspects. The only notable exception is with Deepseek-Math when refining for AF, where improvement in AF results in a decline in FC. Employing a stronger model for refinement, such as GPT-4.1-mini, enhances the overall refinement capability while still preserving the emphasis on the targeted aspect.

4.3 An Example of Building Multi-Agent System for Iterative Self-Refinement

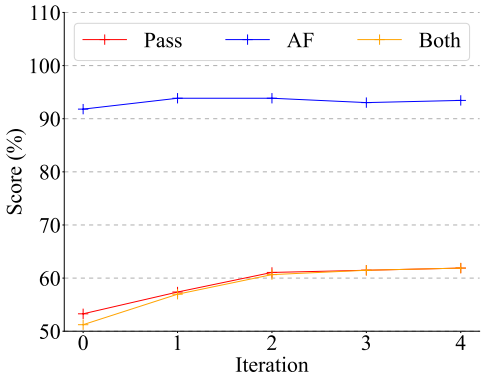
Finally, we present an example of a multi-agent system (Algorithm 3) that leverages an autoformalization agent, a hard-critique agent, a soft-critique agent, a formal refinement agent, and an informal refinement agent to iteratively improve autoformalization performance using a same backend LLM. We evaluate the system on Lean4 using the miniF2F benchmark. Experiments are conducted with Qwen2.5-7B and GPT-4.1-mini, and we report the pass rate, AF assessment, and the percentage of formalizations that are both syntactically correct and semantically aligned in Figure 2. We observe that for a smaller model such as Qwen2.5-7B, improvements in syntactic correctness and semantic alignment are more unstable and fluctuate across iterations. In contrast, a stronger model such as GPT-4.1-mini exhibits smoother and more consistent improvements across iterations. Notably,

Algorithm 3 Iterative Self-Refinement

Input: informal statements S, (Optional) informal-formal pairs $\{s, \phi\}$, aspect description d, number of iterations n
Instantiate an LLM m_1
Instantiate an autoformalization agent a_{auto} with m_1
Instantiate a hard critique agent a_{hard}
Instantiate an LLM m_2
Instantiate a soft critique agent a_{soft} with m_2, d
Instantiate a formal refinement agent a_{formal} with m_1
Instantiate an informal refinement agent a_{informal} with m_1
for $s_i \in S$ **do**
    Formalization $\phi_i = a_{\text{auto}}(s_i, \{s, \phi\})$
    for $j = 1, \ldots, n$ **do**
        Correctness, Error details $c, e = a_{\text{hard}}(\phi_i)$
        if c is False **then**
            Refinement $\phi_i = a_{\text{formal}}(s_i, \phi_i, e)$
        else
            Explanation, Judgment $e, j = a_{\text{soft}}(s_i, \phi_i)$
            if j is False **then**
                Refinement $\phi_i = a_{\text{informal}}(s_i, \phi_i, d, e, j)$



(a) Qwen2.5-7B



(b) GPT-4.1-mini

Figure 2: Lean4 formalization results on miniF2F test set with iterative self-refinement.

Qwen2.5-7B fails to achieve further gains beyond the first iteration, primarily due to its limited ability to perform formal refinement. On the other hand, GPT-4.1-mini shows consistent improvement with each iteration and is increasingly capable of producing formalizations that are both syntactically correct and semantically aligned.

5 Related Work

Autoformalization Autoformalization as an application has demonstrated success in natural language tasks (Quan et al., 2024a,b) and formal mathematical reasoning (Jiang et al., 2023; Tarrach et al., 2024). Recently, with the increasing capabilities of large language models (LLMs), prompting-based methods have shown effectiveness in autoformalizing mathematical statements in Isabelle (Wu et al., 2022; Zhang et al., 2024; Li et al., 2024; Zhang et al., 2025) and Lean (Yang et al., 2023; Lu et al., 2024; Liu et al., 2025a; Peng et al., 2025). In parallel, Jiang et al. (2024) and Liu et al. (2025b) developed data generation pipelines for constructing large-scale parallel corpora of theorem statements. Despite these advancements, there is a lack of flexible implementations to facilitate a quick start in autoformalization research. Our work aims to address this gap.

Multi-Agent Multi-Agent systems have emerged as a growing trend with the development of LLMs (Guo et al., 2024). Researchers have designed systems for applications such as operating systems (Mei et al., 2024), medical education (Wei et al., 2024), answer verification (Lifshitz et al., 2025), and various reasoning tasks, including arithmetic and general reasoning (junyou li et al., 2024). However, there have been limited attempts to build multi-agent systems in the context of autoformalization. Our implementation aims to facilitate progress in this area.

6 Conclusion

In this work, we present a demonstration of building modular agents for multi-agent systems in autoformalization. Our implementation showcases the potential of collaborative agent-based approaches in tackling the challenges of autoformalization. By providing an accessible and flexible framework, we aim to lower the barrier for researchers and developers interested in exploring this emerging field. Through our demonstration, we illustrate how different agents can work together to iteratively refine and improve formalization outputs. We hope that this demonstration serves as a practical resource for researchers seeking to design and implement their own multi-agent autoformalization systems. We believe this work will contribute to the advancement of AI-driven mathematical reasoning. By fostering collaboration and innovation, we believe this work will contribute to the advancement of AI-driven mathematical reasoning and formal verification.

7 Limitations

Despite its contributions, this work has several limitations. First, the proposed multi-agent system lacks a central intelligence agent to distribute and control the different agents. More advanced multi-agent systems still need to be developed, and our implementation can serve as a foundation for such efforts. Additionally, the semantic evaluation of formalizations is limited to high-level judges. Evaluations involving judges with more fine-grained criteria as proxies for semantic analysis are required.

Acknowledgements

This work was partially funded by the Swiss National Science Foundation (SNSF) projects NeuMath (200021_204617) and RATIONAL (200021E_229196).

References

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *Preprint*, arXiv:2302.12433.

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, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *Automated Deduction - CADE-25*, pages 378–388, Cham. Springer International Publishing.

Deborah Ferreira and André Freitas. 2020. Premise selection in natural language mathematical texts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7365–7374.

Deborah Ferreira, Mokanarangan Thayaparan, Marco Valentino, Julia Rozanova, and Andre Freitas. 2022. To be or not to be an integer? encoding variables for mathematical text. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 938–948, Dublin, Ireland. Association for Computational Linguistics.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *Preprint*, arXiv:2402.01680.

Albert Q. Jiang, Wenda Li, and Mateja Jamnik. 2024. Multi-language diversity benefits autoformalization. In *Advances in Neural Information Processing Systems*, volume 37, pages 83600–83626. Curran Associates, Inc.

Albert Qiaochu Jiang, Sean Welleck, Jin Peng Zhou, Timothee Lacroix, Jiacheng Liu, Wenda Li, Mateja Jamnik, Guillaume Lample, and Yuhuai Wu. 2023. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *The Eleventh International Conference on Learning Representations*.

junyou li, Qin Zhang, Yangbin Yu, QIANG FU, and Deheng Ye. 2024. More agents is all you need. *Transactions on Machine Learning Research*.

Zenan Li, Yifan Wu, Zhaoyu Li, Xinming Wei, Xian Zhang, Fan Yang, and Xiaoxing Ma. 2024. Autoformalize mathematical statements by symbolic equivalence and semantic consistency. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. 2025. Multi-agent verification: Scaling test-time compute with goal verifiers. In *Scaling Self-Improving Foundation Models without Human Supervision*.

Qi Liu, Xinhao Zheng, Xudong Lu, Qinxiang Cao, and Junchi Yan. 2025a. Rethinking and improving autoformalization: towards a faithful metric and a dependency retrieval-based approach. In *The Thirteenth International Conference on Learning Representations*.

Xiaoyang Liu, Kangjie Bao, Jiashuo Zhang, Yunqi Liu, Yuntian Liu, Yu Chen, Yang Jiao, and Tao Luo. 2025b. Atlas: Autoformalizing theorems through lifting, augmentation, and synthesis of data. *Preprint*, arXiv:2502.05567.

Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, Zhicheng Yang, Jing Tang, and Zhijiang Guo. 2024. Process-driven autoformalization in lean 4. *Preprint*, arXiv:2406.01940.

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)*, pages 14605–14631, Toronto, Canada. Association for Computational Linguistics.

Jordan Meadows and André Freitas. 2023. Introduction to mathematical language processing: Informal proofs, word problems, and supporting tasks. *Transactions of the Association for Computational Linguistics*, 11:1162–1184.

Jordan Meadows, Marco Valentino, Damien Teney, and Andre Freitas. 2024. A symbolic framework for evaluating mathematical reasoning and generalisation with transformers. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1505–1523, Mexico City, Mexico. Association for Computational Linguistics.

Kai Mei, Xi Zhu, Wujiang Xu, Wenyue Hua, Mingyu Jin, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. 2024. Aios: Llm agent operating system. *Preprint*, arXiv:2403.16971.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. 2022a. LILA: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5807–5832, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022b. Numglue: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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.

Lawrence C. Paulson. 2000. Isabelle: The next 700 theorem provers. *Preprint*, arXiv:cs/9301106.

Zhongyuan Peng, Yifan Yao, Kaijing Ma, Shuyue Guo, Yizhe Li, Yichi Zhang, Chenchen Zhang, Yifan Zhang, Zhouliang Yu, Luming Li, Minghao Liu, Yihang Xia, Jiawei Shen, Yuchen Wu, Yixin Cao, Zhaoxiang Zhang, Wenhao Huang, Jiaheng Liu, and Ge Zhang. 2025. Criticlean: Critic-guided reinforcement learning for mathematical formalization. *Preprint*, arXiv:2507.06181.

Felix Petersen, Moritz Schubotz, Andre Greiner-Petter, and Bela Gipp. 2023. Neural machine translation for mathematical formulae. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11534–11550, Toronto, Canada. Association for Computational Linguistics.

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.

Xin Quan, Marco Valentino, Louise Dennis, and Andre Freitas. 2024a. Enhancing ethical explanations of large language models through iterative symbolic refinement. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–22, St. Julian's, Malta. Association for Computational Linguistics.

Xin Quan, Marco Valentino, Louise A. Dennis, and Andre Freitas. 2024b. Verification and refinement of natural language explanations through LLM-symbolic theorem proving. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2933–2958, Miami, Florida, USA. Association for Computational Linguistics.

Qwen-Team. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Guillem Tarrach, Albert Q. Jiang, Daniel Raggi, Wenda Li, and Mateja Jamnik. 2024. More details, please: Improving autoformalization with more detailed proofs. In *AI for Math Workshop @ ICML 2024*.

Ngoc Tran, Hieu Tran, Son Nguyen, Hoan Nguyen, and Tien N. Nguyen. 2019. Does bleu score work for code migration? In *Proceedings of the 27th International Conference on Program Comprehension*, ICPC '19, page 165–176. IEEE Press.

Marco Valentino, Deborah Ferreira, Mokanarangan Thayaparan, André Freitas, and Dmitry Ustalov. 2022. TextGraphs 2022 shared task on natural language premise selection. In *Proceedings of TextGraphs-16: Graph-based Methods for Natural Language Processing*, pages 105–113, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Hao Wei, Jianing Qiu, Haibao Yu, and Wu Yuan. 2024. Medco: Medical education copilots based on a multi-agent framework. *Preprint*, arXiv:2408.12496.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021. Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Norman Rabe, Charles E Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*.

Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. *Preprint*, arXiv:2306.15626.

Lan Zhang, Xin Quan, and Andre Freitas. 2024. Consistent autoformalization for constructing mathematical libraries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4020–4033, Miami, Florida, USA. Association for Computational Linguistics.

Lan Zhang, Marco Valentino, and Andre Freitas. 2025. Autoformalization in the wild: Assessing llms on real-world mathematical definitions. *Preprint*, arXiv:2502.12065.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. 2022. minif2f: a cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A A Data Example in the Knowledge Base


```
{"type": "definition",
"text": "",
"statement": "definition fFalse :: bool where\n
    \"fFalse \\<longleftrightarrow> False\"\n",
"assumes": "",
"proof": "",
"using": [],
"abs_imports": [
    "HOL.Meson",
    "HOL.Hilbert_Choice"
],
"source": "HOL.ATP",
"id": 0}
```


B Complementary Output Results

The direct output of aspect evaluation in the main paper is:

```
Explanation: The natural language statement describes the
    softmax function, which maps a vector from \(\mathbb
    {R}^K\) to \((0, 1)^K\) using the formula \(\sigma(\
    mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j
    }}\). The formal code defines a function `softmax`
    that takes a list of real numbers and returns a list of
    real numbers. It uses the `map` function to apply the
    exponential function to each element of the list,
    calculates the sum of these exponentials, and then
    maps each element to its softmax value using the given
     formula. The code correctly captures the essential
    components of the softmax function: it computes the
    exponential of each element, sums these exponentials,
    and divides each exponential by the sum to produce
    the output list. The code also correctly handles the
    input and output types, mapping from a list of real
    numbers to another list of real numbers, which
    corresponds to the vector transformation described in
    the natural language statement. Therefore, the
    formalized code involves all mathematical concepts in
    the natural language statement.
Judgement: True
```

Informal refinement of this evaluation leads to the formalization:

```
theory Softmax
 imports
   Main
   "HOL.Complex"
begin

definition softmax :: "real list ⇒ real list" where
  "softmax z =
   let exp_z = map exp z;
       sum_exp_z = sum_list exp_z
   in map (λzi. exp zi / sum_exp_z) z"

end
```

C Examples of Denoising Autoformalizations from Mistral

Example 1 (mathd_numbertheory_457): What is the least positive integer n such that 80325 divides $n!$? Show that it is 17.

Autoformalization:

```
theory aime_1995_p5 imports Complex_Mai
begin
theorem aime_1995_p5:
 fixes n :: nat
 assumes h0 : "80325 ∣ n!"
 shows "17 = n"
 sorry
end
% Note: The statement asks for the least positive integer n
    such that 80325 divides n!, but the statement provided
    does not prove that 17 is the answer. It only proves that
     17 is a number that makes 80325 divide n!. To prove
    that 17 is the least positive integer with this property,
    additional work would be required.
```

Denoising:

```
theory aime_1995_p5 imports Complex_Main
begin
theorem aime_1995_p5:
 fixes n :: nat
 assumes h0 : "80325 ∣ n!"
 shows "17 = n"
sorry
end
```

Example 2 (induction_11div10tonmn1ton): Show that for any natural number n, we have $11 \mid 10^n - (-1)^n$.

Autoformalization:

```
theory aime_1995_p1 imports Complex_Main
begin
lemma aime_1995_p1:
 fixes n :: nat
 shows "11 ∣ (pow 10 n) - (- pow 1 + n)"
proof (induction n)
 case 0
 show "11 ∣ 1"
 by (simp add: abs_zero)
 case (Suc n)
 show "11 ∣ (pow 10 (Suc n)) - (- pow 1 + Suc n)"
 have h1 : "11 ∣ (pow 10 n) - (- pow 1 + n)" using n by
     simp
 have h2 : "(pow 10 n) = 10 * (pow 10 n)" by simp
 have h3 : "(- pow 1 + Suc n) = 11 * (- pow 1 + n)" by
     simp
 have h4 : "11 ∣ 10 * (pow 10 n) - 11 * (- pow 1 + n)"
     using h1 by (rule mult_distrib)
 have h5 : "11 ∣ (10 * (pow 10 n) - 10 * (pow 10 n)) + (11 *
      (- pow 1 + n) - 11 * (- pow 1 + n))" using h4 by (
     rule add_mono)
 have h6 : "11 ∣ (10 * (pow 10 n) - 10 * (pow 10 n))" using
     h2 by (rule sub_abs)
 have h7 : "11 ∣ 11 * (- pow 1 + n)" using h3 by (rule
     sub_abs)
 have h8 : "11 ∣ (10 * (pow 10 n) - 11 * (- pow 1 + n))"
     using h6 h7 by (rule add_trans)
 qed
end
```

Denoising:

```
theory aime_1995_p1 imports Complex_Main
begin
lemma aime_1995_p1:
 fixes n :: nat
 shows "11 ∣ (pow 10 n) - (- pow 1 + n)"
sorry
end
```

LearnLens: LLM-Enabled Personalised, Curriculum-Grounded Feedback with Educators in the Loop

Runcong Zhao[1], **Artem Bobrov**[1], **Jiazheng Li**[1], **Cesare Aloisi**[2], **Yulan He**[1]
[1]King's College London, [2]AQA
{runcong.zhao, artem.bobrov, jiazheng.li}@kcl.ac.uk
caloisi@aqa.org.uk, yulan.he@kcl.ac.uk

Abstract

Effective feedback is essential for student learning but is time-intensive for teachers. We present **LearnLens**, a modular, LLM-based system that generates personalised, curriculum-aligned feedback in science education. LearnLens comprises three components: (1) an *error-aware assessment* module that captures nuanced reasoning errors; (2) a *curriculum-grounded generation* module that uses a structured, topic-linked memory chain rather than traditional similarity-based retrieval, improving relevance and reducing noise; and (3) an *educator-in-the-loop* interface for customisation and oversight. LearnLens addresses key challenges in existing systems, offering scalable, high-quality feedback that empowers both teachers and students. A screencast video introducing the system[1] and the demo[2] are available online.

1 Introduction

Timely, high-quality feedback is a key driver of learning across all educational levels (Hattie and Timperley, 2007; Boud and Dawson, 2023). Effective feedback should be personalised, actionable, and dialogic: supporting student engagement and offering clear guidance for improvement (Carless, 2016). However, delivering such feedback at scale remains a major challenge, as it is time-intensive and adds to teachers' already substantial workload and growing levels of stress and fatigue (Jomuad et al., 2021). Recent advances in large language models (LLMs) offer a promising opportunity to automate feedback generation and ease teacher workload. Prior work (Mazzullo and Bulut, 2025; Liu et al., 2025) has explored their use for automatically generating scores and rationales for student answers, as well as for delivering evidence-based feedback in classroom settings. However, existing systems still fall short in several ways: (i) they focus narrowly on scoring, overlooking the learner's partial understanding and reasoning process (Mayfield and Black, 2020; Xie et al., 2022); (ii) they generate feedback directly from LLMs without sufficient curricular grounding, often leading to hallucinated, misaligned, or overly generic suggestions (Mazzullo and Bulut, 2025; Meyer et al., 2024); and (iii) they offer limited avenues for educator control, correction, or customisation during the feedback process (Swamy et al., 2025).

To address these limitations, we present **LearnLens**, a modular LLM-based system for personalised, curriculum-aligned feedback in science education. LearnLens consists of three core components, each directly tackling a corresponding shortcoming. First, an *error-aware assessment* module moves beyond binary correctness by aligning learner responses with structured mark schemes and identifying conceptual, factual, or linguistic errors, enabling more nuanced understanding of student reasoning. Second, a *curriculum-grounded generation* module produces feedback using a restructured, topic-linked memory chain aligned with the national curriculum. Unlike traditional similarity-based retrieval, which often introduces irrelevant or noisy content, our *Chain-of-Concept* framework integrates carefully curated knowledge and curriculum alignment into the retrieval process, ensuring that generated feedback is pedagogically coherent and tailored to specific learning objectives. Third, an *educator-in-the-loop* interface, paired with a separate student-facing interface, enables teachers and students to interact with the system in complementary ways. Teachers can monitor student performance, create customised questions, review and revise feedback using natural language, and optionally select from a suite of embedded verifiers that assess different aspects of feedback quality—such as factual accuracy, curricular relevance, and linguistic clarity.

[1] https://www.youtube.com/watch?v=mCUALVwDKNQ
[2] https://learnlens.co.uk/login

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 625–633
November 4-9, 2025

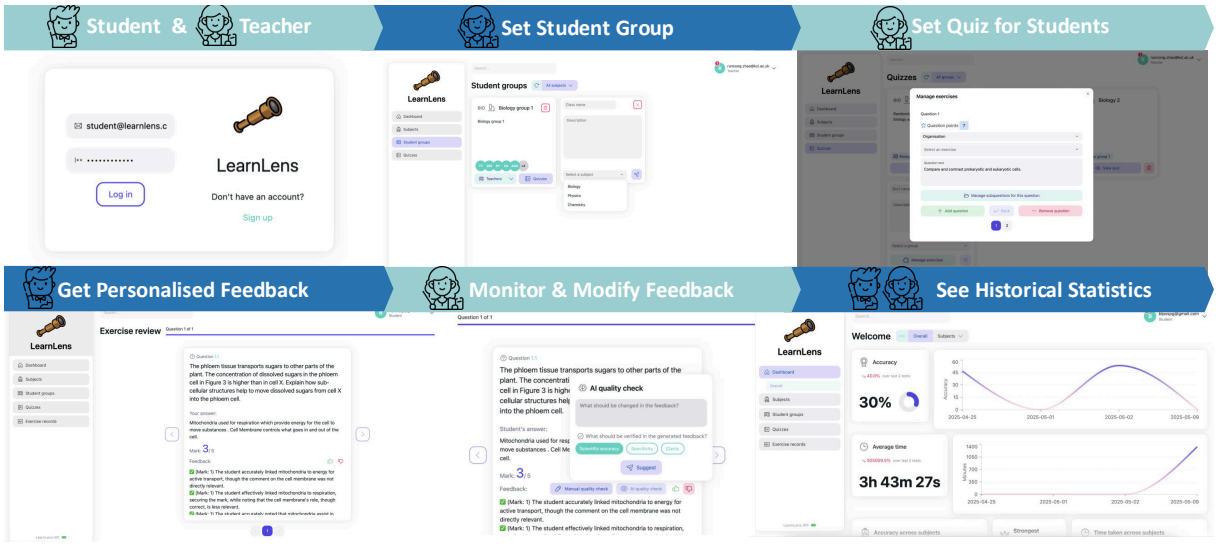


Figure 1: **LearnLens**: A modular LLM-based system delivering personalised, curriculum-aligned feedback through error analysis, topic memory, and educator oversight.

2 Architecture of LearnLens

We design **LearnLens** as a dual-interface intelligent feedback system that serves both teachers and students. Our core goal is twofold: (1) to reduce the workload for teachers while enhancing their ability to monitor student progress, and (2) to provide students with timely, personalised feedback that supports deeper learning. Powered by LLMs, **LearnLens** integrates structured knowledge management and human-in-the-loop collaboration into a scalable educational tool. Through dedicated teacher and student interfaces, the system delivers curriculum-aligned, high-quality support tailored to real classroom needs.

2.1 Teacher Interface: Less Grading, More Guiding

To reduce teacher workload while enhancing instructional oversight, **LearnLens** provides tools that support differentiated instruction, personalised assessment, and high-quality feedback. As shown in Figure 1, the teacher interface facilitates efficient classroom management through student grouping, quiz creation, and curriculum-aligned rubric generation. It also offers up-to-date analytic insights, including performance visualisation and verifier scores, to help teachers identify students in need of support and feedback requiring revision. This integrated, human-in-the-loop workflow enables teachers to focus less on repetitive grading and more on delivering targeted, meaningful guidance.

Student Grouping As illustrated by the *"Set Student Group"* module, teachers can organise students into groups based on subject, year level, and learning progress. Each group can then be assigned one or more teachers and a cohort of students, with appropriate permissions configured to control access to shared materials and feedback within that group. This structure enables differentiated instruction, collaborative planning, and role-based access control in a scalable and intuitive way.

Quiz Creation As illustrated by the *"Set Quiz for Students"* module, teachers can create quizzes and assign them to specific student groups. Each question is first associated with a topic from the national curriculum, ensuring alignment with prescribed learning objectives. For each selected topic, the system provides a curated bank of pre-authored questions that teachers can readily select. Alternatively, teachers may compose their own custom questions to address specific instructional goals. To further enhance flexibility and reduce authoring effort, teachers can also prompt the LLM to automatically generate questions based on the chosen topic and specified pedagogical requirements.

Curriculum-Aligned Rubrics To ensure consistency in model assessment, the system first generates a mark scheme for each question. Since the mark scheme impacts all subsequent assessments, we consider it a critical step where accuracy is prioritised over time efficiency. In line with inference-time scaling laws (Snell et al., 2024; Brown et al., 2024), the model is granted extended reasoning

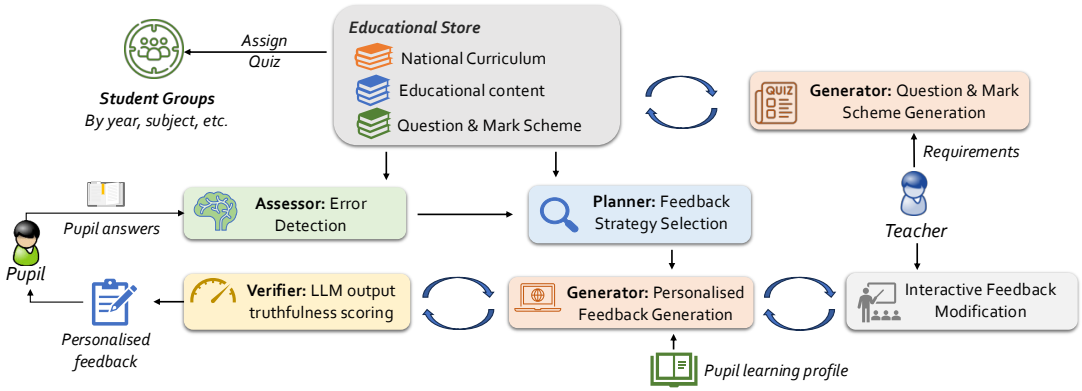


Figure 2: LearnLens: Overall framework.

time to explore diverse solution paths and consider multiple correct approaches. To further ensure pedagogical validity, the system retrieves relevant curriculum content for the target topic. This enables the model to distinguish between knowledge that students are expected to have mastered and content beyond their current level, thereby constraining the mark scheme within the appropriate learning scope.

Student Performance Visualisation To support effective teacher intervention, **LearnLens** provides a historical performance visualisation module that highlights students who may require additional support. Prior studies (Özer Özkan and OZKAN, 2025) and our interviews with teachers suggest that educators often struggle to monitor students' mastery levels in a timely manner. This module addresses that challenge by offering a clear, data-driven overview of student progress, enabling teachers to more efficiently identify learning gaps.

Verifier for Quality Assessment To track the quality of model-generated feedback and assist teachers in identifying responses that may require attention, each feedback instance is first self-evaluated against a set of educational criteria: *Scientific Accuracy* (whether misconceptions are correctly identified and explanations are valid), *Clarity* (whether the language is accessible to GCSE-level students[3]), and *Specificity* (whether the feedback clearly highlights what is correct, incorrect, or missing). For each criterion, the model assigns a self-verification score along with a justification.

As shown in Figure 2, **LearnLens** integrates a verification-and-revision loop into its assessment pipeline. To mitigate model-family bias, verification is performed by a separate foundation model from the generator. Feedback is iteratively refined based on verifier scores: if a response falls short, the model identifies weaknesses and revises it. The process continues until all criteria meet a threshold or the iteration limit is reached:

$$\text{stop} = \left[\min_i r_i \geq \tau\right] \vee \left[t \geq T_{\max}\right],$$

where r_i is the verifier score for the i-th criterion, τ is the acceptance threshold, and t is the current iteration. The highest-scoring feedback is returned. Examples with verification scores are shown in Appendix A.

Interactive Feedback Modification **LearnLens** integrates student performance visualisation and verifier scores to help teachers efficiently identify feedback that may require revision. Teachers can refine flagged feedback through a conversational interface, either by directly editing the content or by providing high-level suggestions. These modifications can be applied to a single question or propagated across the entire quiz. The AI agent then incorporates the teacher's input and generates an updated response accordingly. An illustrative example is provided in Appendix B.

Teacher intervention serves as a signal of dissatisfaction with the initial output. In such cases, the system dynamically allocates additional computational resources, allowing the model to reflect more deeply and produce higher-quality feedback. While faster refinement strategies were explored, they often compromised fidelity to teacher intent. Given this trade-off, we prioritise feedback quality to preserve user trust and learning effectiveness.

2.2 Student Interface: Faster Feedback, Better Understanding

From the student perspective, **LearnLens** delivers timely and tailored feedback that promotes contin-

[3]LearnLens can be tailored to students across all educational levels. In this demo, we focus on GCSE science.

uous learning. To ensure the feedback is both accurate and actionable for learning, the system combines reliable assessment, memory-driven strategy planning, and safe, context-aware generation.

Assessor To deliver *consistent, fine-grained* scoring, the Assessor maps each student answer $\hat{a}_i$ onto the curriculum-aligned mark scheme $\mathcal{C}_i = \{(c_k, w_k)\}_{k=1}^{K_i}$, where c_k is the k-th key concept and $w_k \in \mathbb{Z}_{>0}$ its weight. The raw score is the weighted sum of concept matches:

$$s_i = \sum_{k=1}^{K_i} w_k \, \textsc{Match}(c_k, \hat{a}_i),$$

with MATCH satisfied when the LLM detects that the concept appears in the answer, thus granting partial credit even if the final answer is wrong. For internal consistency we also obtain a *prompt score* s_i^{prompt} via a direct "grade-this-answer" prompt and trigger self-reflection whenever $s_i \neq s_i^{\text{prompt}}$ (Li et al., 2024). LearnLens is also robust to surface errors: grammatical or typographical mistakes do *not* affect the score s_i. Such issues are instead captured by a separate expression-quality flag $\delta_i \in \{0, 1\}$ (1 = major language issues), which is shown in the feedback but excluded from the numerical grade. This design (i) renders the weighting and partial credit scheme *transparent*: students and teachers can directly inspect the marks w_k allotted to each concept c_k and see how many were awarded, which removes the black box impression and eases disputes; and (ii) decouples conceptual understanding from writing quality.

Planner As shown in Figure 3, we organise past assessments by *curriculum topics* rather than raw embedding similarity used in earlier work (Gao et al., 2023). Each question is decomposed into sub-questions (nodes) v_i and labelled with one or more topics $\mathcal{L}(v_i) \subseteq \mathcal{T}$. We model these records as a topic graph

$$G = (V, E, \mathcal{T}), \quad E = \{(v_i, v_j) \mid \mathcal{L}(v_i) \cap \mathcal{L}(v_j) \neq \varnothing\}.$$

For a query q tagged with topics $C_q \subseteq \mathcal{T}$, retrieval is confined to the topic subgraph $G_q = G\big[\{\, v_j \mid \mathcal{L}(v_j) \cap C_q \neq \varnothing\}\big]$, and we return

$$\mathcal{R}(q) = \underset{v_j \in G_q}{\arg\,\text{top-k}}\, f(q, v_j),$$

where f is a FAISS-based ranker (Douze et al., 2024). Topic filtering removes cross-topic noise, yielding more coherent evidence; the module then analyses pupil errors and prior knowledge to choose the most effective feedback strategy.

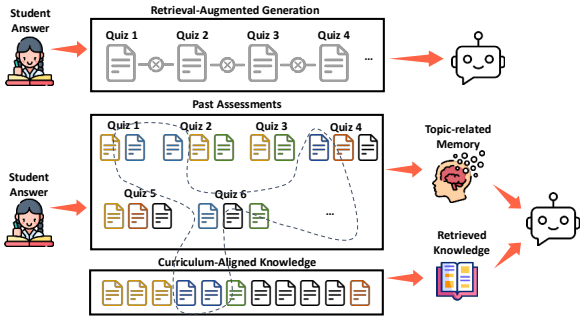


Figure 3: Comparison between traditional RAG and the proposed Chain-of-Concept approach. In RAG (top), past assessment records are stored sequentially without explicit logical structure, forcing the retriever to rely solely on surface-level similarity. This often introduces significant noise, especially as the database grows. In contrast, our approach (bottom) organises past assessments by topic-level relationships, enabling the model to retrieve more contextually relevant information and generate more personalised, coherent feedback.

Generator To generate high-quality feedback, **LearnLens** integrates the quiz content, the corresponding mark scheme, and the selected feedback strategy. We adopt an LLM-based self-reflection mechanism (Asai et al., 2024; Zhang et al., 2024), allowing the model to reason over the student's response. To ensure the safety and appropriateness of the generated content, a safety-aligned model is applied post-generation to filter out any harmful, biased, or otherwise inappropriate language.

3 Evaluation

3.1 Experimental Setup

We conducted a comparative evaluation of LLMs across multiple model families and sizes. Due to student data privacy concerns, all experiments were conducted via local deployment. Our evaluation includes Meta-Llama-3-8B-Instruct, Qwen2.5-32B-Instruct, and QwQ-32B.

3.2 Evaluation Results

3.2.1 User Feedback Results

To evaluate how well **LearnLens** aligns with real-world teaching needs, we collected responses from *30 participants*, all of whom had prior teaching experience in a STEM module at either high school or college level. Each participant interacted with the system and completed a structured questionnaire (Figure 4), which covered topics such as interface usability, system responsiveness, question quality, feedback usefulness, and overall user experience. The complete questionnaire is provided in Appendix C. Our questionnaire results, summarised in Figure 4 (full response distribution) and

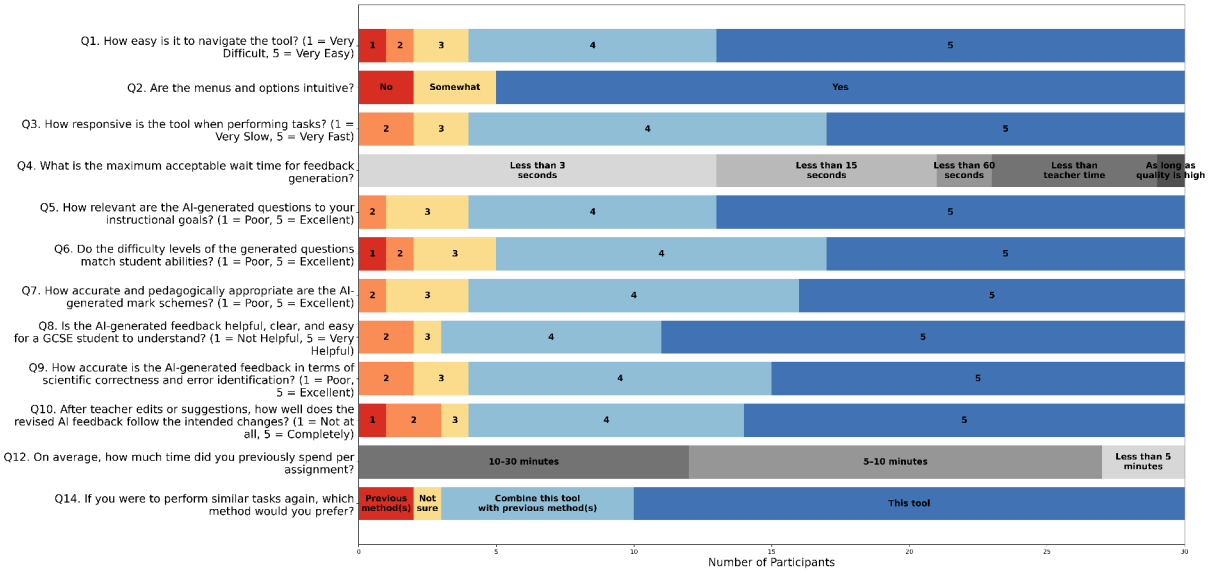


Figure 4: Full Questionnaire Response Distribution (N = 30 per Question). Shades of blue denote more favourable evaluations of the tool, while shades of red indicate dissatisfaction. Grey segments represent neutral factual responses that are not direct indicators of user sentiment toward the tool.

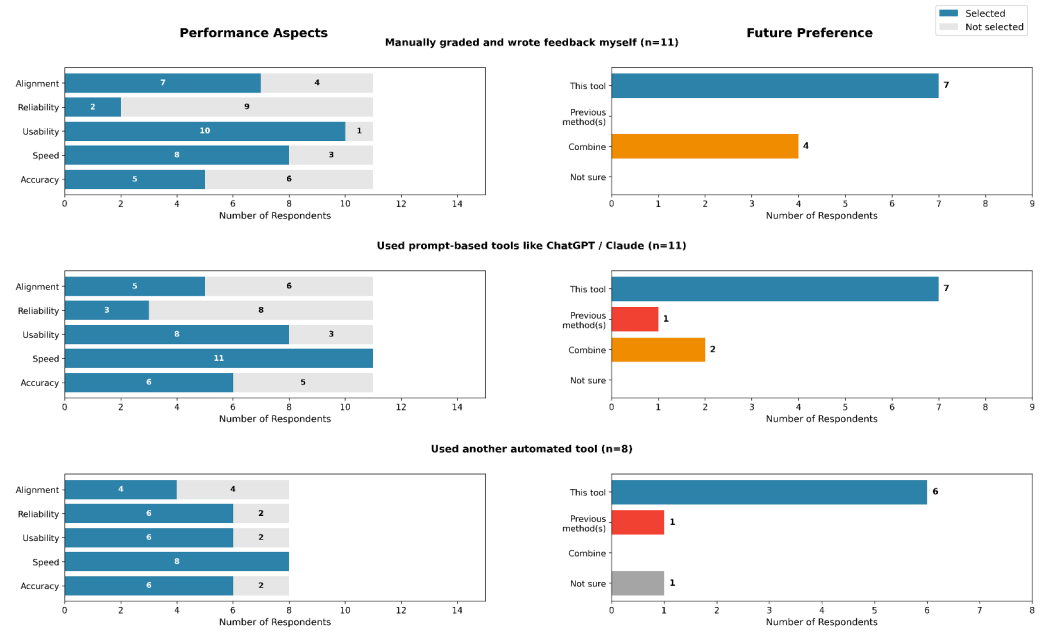


Figure 5: Grouped feedback by participants' prior methods (manual, prompt-based, or automated). Left: perceived improvements across five dimensions. Right: future preferences for task completion. Blue = positive toward our tool; red = preference for previous methods; orange = combine both; grey = neutral or other.

Figure 5 (grouped comparison), highlight the following themes:

Global Satisfaction and Usability Across the nine Likert-scale items (Q1, Q3, Q5–Q10) the mean rating never falls below 4.1/5, signalling a uniformly positive reception. Navigation ease and perceived responsiveness confirm that the dual-interface design achieves its principal UX goal: immediate, low-friction access to feedback tools. Likewise, high scores for relevance to instructional goals (4.4/5) and scientific accuracy (4.3/5) validate our curriculum-grounding strategy.

Categorical items echo this trend. About 80% of teachers describe the menu structure as "intuitive", and 75% expect results in under 15 seconds, a threshold LearnLens already meets in >90 % of cases during local deployment. These data confirm that the system's latency, a common pain-point for classroom AI, is acceptable for real-world use.

	Performance				Generation Latency & Cost			
Model	**MSE**	**Corr.**	**Acc.**	**±1 Acc.**	**Avg. (s)**	**Min. (s)**	**Max. (s)**	**Cost ($)**
LLaMA-3-8B-Ins	3.468	0.235	0.190	0.646	5.14	2.43	9.40	0.0040
Qwen2.5-32B-Ins	3.658	0.230	0.241	0.633	12.39	5.78	20.29	0.0096
QwQ-32B	3.506	0.323	0.329	0.709	45.49	32.27	57.36	0.0351
LearnLens	**3.190**	**0.388**	**0.354**	**0.747**	11.39	7.59	16.97	0.0099

Table 1: **LearnLens** performance and efficiency. MSE: mean squared error; Corr.: Pearson correlation; Acc.: exact match; ±1 Acc.: within–one–mark accuracy. Avg./Min./Max. are generation latencies; Cost is per request.

Efficiency Gains Q12 shows that half of teachers spent 10–30 minutes per assignment, whereas the median with **LearnLens** falls below 5 minutes. Q13 echoes this: 26/30 respondents rank *speed* as the top strength. By off-loading rubric matching and error spotting, **LearnLens** shifts teachers' time from grading to guidance, realigning workload rather than merely automating it.

Experience-Sensitive Priorities Figure 5 segments perceptions by prior practice. Manual graders value speed (91 %) and usability (82 %) most, unsurprising given their baseline workflow. Prompt-tool users distribute credit more evenly, highlighting LearnLens' balanced improvement across accuracy, reliability, and alignment. Their familiarity with LLM quirks likely sharpens expectations. Other-automation users rank accuracy (88 %) and reliability (75 %) highest, suggesting that our verifier-in-the-loop architecture successfully addresses limitations they encountered elsewhere.

A $\tilde{\chi}^2$ test ($\alpha = 0.05$) shows no significant difference in overall adoption intent across groups ($p = 0.63$); every cohort exceeds 65 % "continue using", with the automation-savvy group peaking at 85 %. Thus, while priorities diverge, and willingness to integrate LearnLens is stable.

Implications for Deployment To translate these findings into actionable next steps, we highlight three deployment priorities that will maximise user satisfaction and long-term adoption: (1) Latency ceiling. Meeting the sub-15-second expectation is critical; ongoing optimisation should therefore focus on inference batching and on-device caching. (2) Adaptive onboarding. Manual graders respond to time-saving narratives, whereas technical users are swayed by demonstrable accuracy safeguards; onboarding materials should branch accordingly. (3) Verifier transparency. High appreciation for accuracy among automation veterans highlights the value of surfacing verifier scores (§2.1); exposing these metrics to all users could further bolster trust.

In summary, the study confirms that LearnLens delivers broadly perceived pedagogical value, but supporting diverse existing workflows is crucial for its scalable adoption in classrooms.

3.3 Agent Performance and Efficiency

Table 1 reports results on *100 authentic student answers* to a five-mark short-essay science question. Baselines use the same foundation models as those employed in LearnLens, but perform both feedback generation and verification through direct prompts to the same model, without modular decomposition. For fairness, all methods share the same inputs (question, mark scheme, answer) and required outputs (score, feedback, verification).

LearnLens follows a *modular* pipeline: lightweight models handle routine subtasks, while larger models are invoked only when deeper reasoning is needed. Combined with efficient `vLLM` serving and speculative decoding, this design *achieves the strongest scoring quality without additional overhead*: MSE drops to 3.19 (8–13 % below any baseline), correlation rises to 0.388, and both exact and ±1 accuracies surpass the best baseline (QwQ-32B). At the same time, average generation latency is 11.4 s, comparable to Qwen2.5-32B and four times faster than QwQ-32B, while the per-request cost ($0.0099) matches Qwen2.5-32B and is 72 % cheaper than QwQ-32B.

4 Conclusion

We introduce **LearnLens**, a dual-view feedback system that supports both teachers and students through curriculum-aligned assessment and personalised feedback. By combining LLM capabilities with human-in-the-loop design, LearnLens reduces teacher workload and enhances student learning, offering a practical path toward scalable, classroom-ready AI assistance. While LearnLens shows promise in supporting teachers, we acknowledge the lack of student evaluation and plan to address this in future work.

Acknowledgments

This work was supported by the UK Department for Education through the AI Catalyst Fund - AI Tools for Education (grant no. 10144187) and the UK Engineering and Physical Sciences Research Council (EPSRC) through a Turing AI Fellowship (grant no. EP/V020579/1, EP/V020579/2) and a Prosperity Partnership project with AQA (UKRI566). The authors also acknowledge the use of the King's Computational Research, Engineering, and Technology Environment (CREATE) at King's College London.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

David Boud and Phillip Dawson. 2023. What feedback literate teachers do: an empirically-derived competency framework. *Assessment & Evaluation in Higher Education*, 48(2):158–171.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

David Carless. 2016. Feedback as dialogue. In *Encyclopedia of Educational Theory and Philosophy*, pages 286–289. Springer.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *CoRR*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

John Hattie and Helen Timperley. 2007. The power of feedback. *Review of Educational Research*, 77(1):81–112.

Perlito D Jomuad, Leah Mabelle M Antiquina, Eusmel U Cericos, Joicelyn A Bacus, Juby H Vallejo, Beverly B Dionio, Jame S Bazar, Joel V Cocolan, and Analyn S Clarin. 2021. Teachers' workload in relation to burnout and work performance. *International Journal of Educational Policy Research and Review*, 8(2):48–53.

Jiazheng Li, Hainiu Xu, Zhaoyue Sun, Yuxiang Zhou, David West, Cesare Aloisi, and Yulan He. 2024. Calibrating llms with preference optimization on thought trees for generating rationale in science question scoring. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5452–5479, Miami, Florida, USA. Association for Computational Linguistics.

Jiahong Liu, Zexuan Qiu, Zhongyang Li, Quanyu Dai, Jieming Zhu, Minda Hu, Menglin Yang, and Irwin King. 2025. A survey of personalized large language models: Progress and future directions. *arXiv preprint arXiv:2502.11528*.

Elijah Mayfield and Alan W Black. 2020. Should you fine-tune BERT for automated essay scoring? In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 151–162. Association for Computational Linguistics.

Elisabetta Mazzullo and Okan Bulut. 2025. Automated feedback generation for open-ended questions: Insights from fine-tuned llms. In *Proceedings of Large Foundation Models for Educational Assessment*, volume 264 of *Proceedings of Machine Learning Research*, pages 103–120. PMLR.

Jennifer Meyer, Thorben Jansen, Ronja Schiller, Lucas W Liebenow, Marlene Steinbach, Andrea Horbach, and Johanna Fleckenstein. 2024. Using llms to bring evidence-based feedback into the classroom: Ai-generated feedback increases secondary students' text revision, motivation, and positive emotions. *Computers and Education: Artificial Intelligence*, 6:100199.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Vinitra Swamy, Davide Romano, Bhargav Srinivasa Desikan, Oana-Maria Camburu, and Tanja Käser. 2025. iLLuMinaTE: An LLM-XAI framework leveraging social science explanation theories towards actionable student performance feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28431–28439. AAAI Press.

Jiayi Xie, Kaiwei Cai, Li Kong, Junsheng Zhou, and Weiguang Qu. 2022. Automated essay scoring via pairwise contrastive regression. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, pages 6467–6477. International Committee on Computational Linguistics.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024. Self-contrast: Better reflection through inconsistent solving perspectives. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3602–3622, Bangkok, Thailand. Association for Computational Linguistics.

Yasemin Özer Özkan and Murat OZKAN. 2025. Decoding teachers' dilemma: Unveiling the real obstacles

to implementing formative assessment in the classroom. *Journal of Qualitative Research in Education*, 41:95–118.

A Example: Personalised Feedback with Self-Verification Scores

To illustrate how **LearnLens** produces feedback with built-in quality assurance, we present an example of AI-generated personalised feedback accompanied by self-verification scores. The system evaluates student responses against a mark scheme, assigns partial credit where appropriate, and generates targeted feedback. Verification scores reflect the feedback's scientific accuracy, clarity, and specificity, while reasoning statements explain how each score was determined.

Sample Feedback with Verification

AI-Generated Feedback:

- **Mark: 0** – Include that cell X contains mitochondria, which are needed to provide energy for active transport.
- **Mark: 1** – Correctly identified active transport moves sugars against the concentration gradient from low (cell X) to high (phloem).
- **Mark: 0** – Mention that mitochondria enable aerobic respiration to release energy specifically needed for active transport.
- **Mark: 1** – Correctly explained energy is needed to move sugars against the concentration gradient via active transport.
- **Mark: 1** – Correctly stated active transport moves sugars from low to high concentration in the phloem.

Total Mark: 3/5

Verification Scores:

- *Scientific Accuracy:* **5**
- *Specificity:* **5**
- *Clarity:* **4**

Verification Reasoning:

- *Scientific Accuracy:* Full marks.
- *Specificity:* Feedback addresses specific content of the student's answer by highlighting what was right, wrong, or missing.
- *Clarity:* Language is generally clear and accessible to GCSE students, though some simplification could reduce potential confusion.

B Example: Interactive Feedback Modification

To illustrate how teacher input is incorporated into the feedback generation pipeline, we present an example of interactive feedback refinement. The original AI-generated feedback is revised based on a set of teacher-provided suggestions that emphasise clarity, brevity, and actionability. The differences introduced by the model in response to teacher suggestions are highlighted in red for clarity and ease of comparison.

Original AI Feedback

- **Mark: 0** – Add explicit mention of mitochondria in cell X. Specify that mitochondria in cell X facilitate aerobic respiration to provide energy needed for active transport.
- **Mark: 1** – Correctly identified active transport moving sugars against the concentration gradient from cell X to phloem.
- **Mark: 0** – Specify that energy released via respiration is required for active transport. Include "to release energy needed for active transport."
- **Mark: 1** – Correctly explained movement against the concentration gradient via active transport. Correct direction noted.
- **Mark: 1** – Correctly stated active transport from low (cell X) to high concentration (phloem).

Teacher Suggestions:

- Do not include general suggestions.
- Give short, specific instructions that students can implement.
- Keep the feedback concise.

Revised Feedback (with Highlighted Edits)

- **Mark: 0** – Specify that cell X contains many mitochondria, which are needed for aerobic respiration to produce energy for active transport. (concise)
- **Mark: 1** – Correctly identified active transport moving sugars against the concentration gradient from cell X to phloem. (unchanged)
- **Mark: 0** – Add that mitochondria carry out respiration to release energy specifically for active transport of sugars.
- **Mark: 1** – Correctly explained movement against the concentration gradient via active transport. Direction (low → high) is accurate.

- **Mark:** **1** – Correctly stated active transport from low (cell X) to high concentration (phloem). (unchanged)

C User Questionnaire

To better understand user needs and evaluate the usability and effectiveness of **LearnLens**, we designed a structured questionnaire for teachers. The survey covered five key dimensions: interface design, system performance, question creation and assessment, personalised feedback quality, and overall user experience. The full questionnaire is detailed below.

Section 1: Interface Design

Q1. How easy is it to navigate the tool? (1 = Very Difficult, 5 = Very Easy)

Q2. Are the menus and options intuitive?
- Yes
- No
- Somewhat

Section 2: Performance

Q3. How responsive is the tool when performing tasks? (1 = Very Slow, 5 = Very Fast)

Q4. What is the maximum acceptable wait time for feedback generation?
- Less than 3 seconds
- Less than 15 seconds
- Less than 60 seconds
- Less than the time a teacher would typically spend providing feedback
- As long as the feedback is high quality

Section 3: Question Creation

Q5. How relevant are the AI-generated questions to your instructional goals? (1 = Poor, 5 = Excellent)

Q6. Do the difficulty levels of the generated questions match student abilities? (1 = Poor, 5 = Excellent)

Q7. How accurate and pedagogically appropriate are the AI-generated mark schemes? (1 = Poor, 5 = Excellent)

Section 4: Personalised Feedback

Q8. Is the AI-generated feedback helpful, clear, and easy for a GCSE student to understand? (1 = Not Helpful, 5 = Very Helpful)

Q9. How accurate is the AI-generated feedback in terms of scientific correctness and error identification? (1 = Poor, 5 = Excellent)

Q10. After teacher edits or suggestions, how well does the revised AI feedback follow the intended changes? (1 = Not at all, 5 = Completely)

Section 5: Overall Experience & Suggestions

Q11. Before using our tool, how did (or would) you complete similar tasks? (Select all that apply)
- Manually graded and wrote feedback myself
- Used prompt-based tools like ChatGPT / Claude
- Used another automated tool
- Other

Q12. On average, how much time did you previously spend per assignment (assuming a format similar to standard past paper design)?
- Less than 5 minutes
- 5–10 minutes
- 10–30 minutes
- More than 30 minutes
- Other: _____

Q13. Compared to your previous method(s), in which aspects does this tool perform better? (Select all that apply)
- Accuracy
- Speed
- Usability
- Reliability
- Alignment with curriculum

Q14. If you were to perform similar tasks again, which method would you prefer?
- I would continue using this tool
- I would return to my previous method(s)
- I would combine this tool with previous method(s)
- I'm not sure yet

o-MEGA: Optimized Methods for Explanation Generation and Analysis

Ľuboš Kriš[♠] **Jaroslav Kopčan**[♠] **Qiwei Peng**[♡]
Andrej Ridzik[♠] **Marcel Veselý**[♠] **Martin Tamajka**[♠]
♠Kempelen Institute of Intelligent Technologies
♡University of Copenhagen
{name.surname}@kinit.sk[♠] qipe@di.ku.dk[♡]

Abstract

The proliferation of transformer-based language models has revolutionized NLP domain while simultaneously introduced significant challenges regarding model transparency and trustworthiness. The complexity of achieving explainable systems in this domain is evidenced by the extensive array of explanation methods and evaluation metrics developed by researchers. To address the challenge of selecting optimal explainability approaches, we present **o-mega**, a hyperparameter optimization tool designed to automatically identify the most effective explainable AI methods and their configurations within the semantic matching domain. We evaluate o-mega on a post-claim matching pipeline using a curated dataset of social media posts paired with refuting claims. Our tool systematically explores different explainable methods and their hyperparameters, demonstrating improved transparency in automated fact-checking systems. As a result, such automated optimization of explanation methods can significantly enhance the interpretability of claim-matching models in critical applications such as misinformation detection, contributing to more trustworthy and transparent AI systems.

1 Introduction

The incredible success of today's transformer-based language models drastically changed the landscape of the natural language processing domain and how we approach complex tasks within it. However, with great power comes great responsibility. This success comes with a significant trade-off: the increasing complexity of these models has made them essentially black boxes, limiting their adoption in critical applications where transparency and interpretability are of great importance. Traditionally, addressing such interpretability challenges has relied on post-hoc relevance attribution methods, which provide post-hoc explanations (Lapuschkin et al., 2019; Søgaard, 2021) for trained models by assigning importance scores to input features or model parameters. While these explainable AI (XAI) techniques have proven valuable in revealing model behavior and identifying potential underlying flaws, so-called "Clever Hans" effects, or biases, they introduce a new challenge: the overwhelming variety of available XAI algorithms and their possible configurations. Just as practitioners may struggle to select optimal model architectures or hyperparameters, there is now emerging equally complex task of choosing the right explainability method for a given use case (Ali et al., 2023). This explainability challenge is especially relevant, for instance, in semantic matching tasks, where understanding why a model determined that two pieces of text are semantically similar or different is crucial for building user trust and ensuring reliable performance. Moreover, the interpretable semantic matching scenario is relevant in situations where the nuanced relationships between two texts require explanations that describe the model's decision-making process precisely and understandably to end users. Different XAI algorithms may highlight different aspects of the input text, leading to varying levels of usefulness depending on the specific matching task, domain, and user requirements. The manual process of evaluating and selecting appropriate XAI configurations is time-consuming, resource-intensive, subjective, and often suboptimal. Our **o-mega**[1] tool addresses this critical gap by automating the selection and optimization of post-hoc explainability algorithms, specifically within the semantic matching tasks. We have built the **o-mega** tool on top of previous work, which was inspired by the AutoML concept (Thornton et al., 2013), and adjusted to the domain of explainable AI. The main functionality is focused on the systematic evaluation of different

[1]We have released the code as well as documentation and examples at o-mega repository.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 634–642

XAI methods and their configurations in order to identify those that provide the most useful explanations for a given model and dataset combination according to specified criteria. The main motivation for the **o-mega** tool stems from the fact that explainability is not one-size-fits-all. In the semantic matching domain—whether it is a claim-matching task for fact-checking, document similarity assessment, or content recommendation—users need explanations that help them understand and trust the model's matching decisions. Since explainability is often overlooked because of its often ambiguous explanations, which are difficult to understand and struggles with implementation of the methods itself, by automating the explainability process of finding satisfactory XAI configurations, **o-mega** enables the use of interpretable semantic matching systems more efficiently while ensuring that the explanations provided are both technically correct and somehow practically useful. This tool represents an addition to the claim-matching task, which is central to fact-checking applications (Vo and Lee, 2020; Kazemi et al., 2021; Peng et al., 2025), and it is the first action step towards our effort to make it possible for domain experts to obtain meaningful insights from complex deep learning models without requiring extensive expertise in XAI methodologies.

2 Related Work

A wide range of explainable Artificial Intelligence (XAI) methods have been developed to enhance understanding of the decision-making process in machine learning models. Among post-hoc techniques, perturbation-based approaches are widely used where model predictions are examined by systematically sampling perturbed versions of the input and observing corresponding changes in output. LIME (Ribeiro et al., 2016) provides local explanations by fitting an interpretable surrogate model to approximate the behavior of a complex model. SHAP (Lundberg and Lee, 2017) attributes feature importance based on Shapley values from cooperative game theory. Other perturbation-based methods include Occlusion Sensitivity (Zeiler and Fergus, 2014), which assesses feature relevance by masking parts of the input directly. In contrast, gradient-based methods such as Integrated Gradients (Sundararajan et al., 2017) and LRP (Bach et al., 2015) propagate relevance scores or gradients backward through the model to identify influential features. Additionally, some studies propose textual explanation generation methods that aim to produce human-readable justifications for model predictions (Lei et al., 2016; Camburu et al., 2018; Atanasova et al., 2020). Explainability methods provide explanations of different qualities, and various metrics have been proposed to evaluate them. Common evaluation criteria include faithfulness, plausibility, and stability (Alvarez-Melis and Jaakkola, 2018; Mohseni et al., 2021; Nauta et al., 2023). However, no single metric captures all aspects of explanation quality, and different metrics may yield conflicts. This makes the evaluation of XAI methods a persistent challenge. This challenge is further compounded when selecting the most appropriate explanation technique for a given task. To address this, AutoXAI frameworks (Cugny et al., 2022), inspired by AutoML systems (He et al., 2021), have been proposed to automate the selection and configuration of XAI techniques. The framework aims to adaptively choose the most suitable explanation method and optimize associated hyperparameters based on user-defined objectives. The existing AutoXAI framework works only with the tabular modality of data and has implemented only two methods - SHAP and LIME, which are often considered as baselines when dealing with structured data. Inspired by it, we have created the **o-mega** framework within the domain of information retrieval, focusing on textual data. oMEGA framework incorporates multiple XAI methods, metrics for evaluation, and puts a specific focus on the claim matching task, which is a crucial task in fact-checking.

3 System Description

The **o-mega** tool is designed as a comprehensive framework for automatically selecting and optimizing explainable AI methods within semantic matching application. As illustrated in Figure 1, the system architecture consists of four interconnected modules that work together to identify the most effective explanation approach for a given AI model and dataset: 1) the **model** component that generates predictions requiring explanation, 2) the **method** component, which is a comprehensive space of available XAI algorithms and their configurations, 3) the **metric** part is an evaluation module incorporating both proxy measures and ground-truth annotations, and 4) the **optimization** component is a conditional hyperoptimization engine that

systematically explores and ranks different explainability approaches.

3.1 Methods

For semantic matching and text classification tasks, we require attribution-based XAI methods capable of capturing token-level or word-level importance. Additionally, for semantic matching specifically, these methods must capture cross-sequence interactions as well. We also want the methods to be diverse in their design and computationally efficient, therefore we have selected XAI methods spanning three paradigms: 1) gradient-based; 2) perturbation-based; and 3) architecture-specific. Overall, 11 explainable methods were implemented, 9 of them were from the Captum library. The entire selection is presented in Table 2. We have excluded several attribution methods from the library like Deconvolution, DeepLift, DeepLiftShap, because of irrelevancy for our task, or computational expenses.

3.2 Models

The inherent design of how these post-hoc explainability methods are computed introduces significant architectural rigidness. This introduces a constraint to same extent - the scoping of supported models. We have included an exhaustive list of most widely used semantic matching models. Table 1 presents the list of models which we have thoroughly tested. However, in general, the methods available within the **o-mega** framework should work with any transformer-based model for semantic matching of natural language text.

Model name	Embedding layer
sentence-transformers/gtr-t5-large	encoder.embed_tokens
sentence-transformers/gtr-t5-xl	encoder.embed_tokens
sentence-transformers/sentence-t5-xl	encoder.embed_tokens
sentence-transformers/all-mpnet-base-v2	embeddings.word_embeddings
sentence-transformers/multi-qa-mpnet-base-cos-v1	embeddings.word_embeddings
sentence-transformers/all-MiniLM-L12-v2	embeddings.word_embeddings
BAAI/bge-large-en-v1.5	embeddings.word_embeddings
BAAI/bge-base-en-v1.5	embeddings.word_embeddings
BAAI/bge-small-en-v1.5	embeddings.word_embeddings
llmrails/ember-v1	embeddings.word_embeddings
thenlper/gte-large	embeddings.word_embeddings
intfloat/e5-large-v2	embeddings.word_embeddings
BAAI/bge-large-en-v1.5	embeddings.word_embeddings

Table 1: Overview of tested language models for explainable use in semantic matching

3.3 Metrics

Evaluation module of the **o-mega** tool is the most crucial one, because based on the measurements of quality of computed explanations, the optimization process is guided, and at the output ends the recommendation for best methods regarding the task, data and model used. In order to evaluate how good the provided explanations actually are, it is important to firstly define what is meant by *the explanation quality - what constitutes to a good explanation?* There are two main approaches for how to answer this quality question, the first one is quantitative measurements, the second one is qualitative. While the qualitative measurements are just as much important, because of their subjective nature they can not be expressed by metrics. Therefore, within the o-mega evaluation we have focused primarily on the quantitative approach. On this topic was a lot of work done (Zhou et al., 2021; Markus et al., 2021) about how to find out if the explanation is of high quality - it needs to adhere to two components - *fidelity* and *plausibility*.

Method
Captum-based methods
Occlusion token-level
Input X Gradient
Guided Backprop
Feature Ablation
Kernel Shap
Gradient Shap
LIME
Saliency
Custom implementations
GAE
Conservative LRP
Occlusion word-level

Table 2: Overview of available XAI methods

- **Explanation high in fidelity** reflects the actual underlying behavior of the model, meaning the features deemed as important by explanation method for specific prediction, should be the same which influenced model the most. Metrics designed to test this are based on ablations or perturbations.

- **Explanation high in plausibility** reflects how easy is to comprehend the explanation for human-being, while this being the closest measure possible to qualitative evaluation within the quantitative domain. In order to be able express this semi-subjective property these metrics needs the human annotations - what humans themselves have deemed to be comprehensible explanation given the data and the task.

While during the evaluation is important to score high in both of these measurements, they often

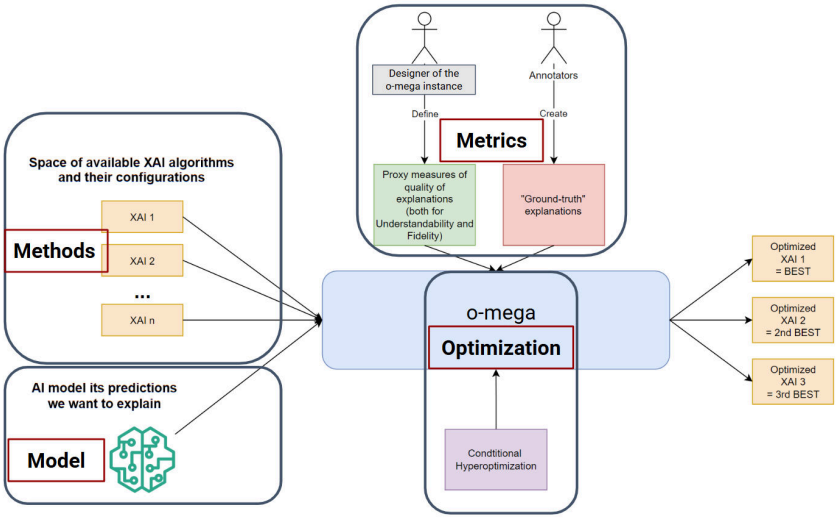


Figure 1: The architecture of the **o-mega** tool.

conflict with each other. An optimal explanation requires to strike a balance between them, while prioritizing one typically comes at the expense of the other. Our **o-mega** framework has available metrics for both categories - fidelity and plausibility which we have deemed as important and user could choose before the optimization if he wants to favor one over the other, or optimize towards both of them. All these metrics are custom implemented versions of established, previously published metrics which work with textual data (DeYoung et al., 2020; Hedström et al., 2023; Attanasio et al., 2023).

Metrics	Category
AOPC Comprehensiveness AOPC Sufficiency	Fidelity
AUPRC Token-level F1 scores Token-level IoU Average Precision Score	Plausibility

Table 3: Overview of available metrics

3.4 Optimization

For hyperparameter optimization, we employed the Optuna library (Akiba et al., 2024), which offers significant advantages over traditional grid search approaches. While grid search exhaustively evaluates all possible parameter combinations, Optuna uses intelligent search strategies such as Tree-structured Parzen Estimator (TPE) and Bayesian optimization that learn from previous evaluations to guide future parameter selection (Feurer and Hutter, 2019). This adaptive approach allows the optimizer to focus computational resources on promising regions of the hyperparameter space, effectively avoiding areas that have already shown poor performance. The efficiency gains become particularly pronounced in high-dimensional hyperparameter spaces, where grid search suffers from the curse of dimensionality. For instance, if we have 5 hyperparameters with 10 possible values each, grid search would require 10^5 = 100,000 evaluations, while Optuna's intelligent sampling can often find near-optimal configurations with significantly fewer trials—often by an order of magnitude or more. This efficiency is crucial in our context, where Captum's explanation methods have numerous hyperparameters that can substantially impact both computational cost and explanation quality. Moreover, Optuna's pruning capabilities allow early termination of unpromising trials, further reducing computational overhead. This is particularly valuable when working with explanation methods that may have expensive evaluation metrics, as the optimizer can quickly identify and abandon parameter configurations that are unlikely to yield good results based on partial evaluations. The result is a more efficient exploration of the hyperparameter space that converges to high-quality explanations faster than traditional optimization approaches.

3.5 Configuration

As a base part of the **o-mega**, the user can select a model from the Huggingface library (See Table 1) or a local pre-trained model. Plausibility and faithfulness have corresponding weights to adjust the priority of each group of metrics (See Figure 6). Then, the form of evaluation of explanations can take place in 3 options: either in token form, or by

combining explanations and masks into sentences, or into words. One example of the base configuration is shown in Figure 2.

```
model_path: "intfloat/multilingual-e5-
    large"
embeddings_module_name: "embeddings.
    word_embeddings"
methods: ["GAE_Explain","Occlusion"]
normalizations: ["without_normalize"]
explanation_maps_token: True
plausability_weight: 0.5
faithfulness_weight: 0.5
multiple_object: False
```

Figure 2: Configuration block specifying the base parameters for hyperoptimization and evaluation of explanations

The user can further configure 6 search strategies from the Optuna library (See Figure 3). These 6 search strategies can be divided into 2 groups: (1)single-object and (2)multi-object hyperoptimization. Within the configuration, plausibility and plausibility values can be considered separately. Within single-object hyperoptimization, we have TPESampler, GPSampler, and BruteForceSampler available for use, and within multi-object hyperoptimization, we can use NSGAIIISampler, NSGAIISampler, and TPESampler.

```
Optuna_parameters:
  sampler: "TPESampler"
  n_trials: 14
  n_startup_trials: 4
  seed: 1000
```

Figure 3: Configuration block for Optuna Sampler and its parameters.

XAI methods also accept a list of configurations for customization. One example is shown in Figure 4, where the hyperparameters of XAI methods can be restrained to a specific search space.

```
method_param:
  Gradient Shap:
    parameters:
      stdevs: (0.1, 0.9, {'step': 0.1})
      n_samples: [10, 15]
  Lime:
    parameters:
      n_samples: [80, 90]
  token_groups_for_feature_mask: true
```

Figure 4: Configuration block specifying hyperparameters for explainable method.

4 Case Study

In this case study, we focus on the task of claim matching, a key component in automated fact checking. The claim matching task is typically formulated as an information retrieval task. Given

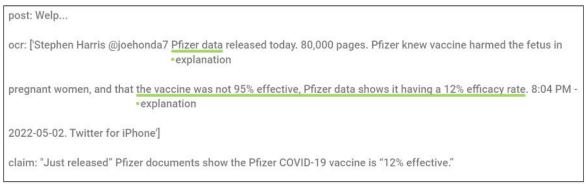


Figure 5: One example of the post-claim pair with annotated explanations.

an input text (e.g., social media posts), the goal is to find an appropriate claim from a database of claims that have already been fact-checked by professional fact-checkers. When a user posts a post making a claim worth fact-checking, the model aims to find a semantically similar claim from a list of previously fact-checked claims. For this study, we utilize the MultiClaim dataset (Pikuliak et al., 2023). To examine interpretability, five human annotators are asked to provide explanations for why specific claims were matched. Examples are given in figure 5. To assess the quality of the annotation, if the majority vote is not determined, we directly throw the sample away. This results in 512 post-claim pairs with human annotations.

4.1 Task Configuration

To generate optimal explanations for this task, we developed a hyperparameter optimization approach that selects the best combination of explanation methods and their configurations. The explanation methods identify significant attributes in both posts and claims by analyzing the cosine similarity between them. We utilized the Captum library for generating explanations, which provides multiple options for modifying explainable method parameters that affect how individual embeddings are processed. Our hyperparameter optimization framework incorporates these configurable parameters as well. Specific methods need to set the model in a specific way. In Figure 6 are shown the model parameters required for each explanation method are shown, such as the necessary layer or gradient settings, which show how different methods require access to specific components of the neural architecture. Figure 7 details the specific tunable parameters for each explanation method, such as perturbation sizes for occlusion methods or baseline values for gradient-based approaches. For the optimization process, we employed the Optuna library, which offers various optimization algorithms. Our evaluation framework uses five metrics which can be organized into two categories: fidelity and

plausibility. Additionally, we included average precision score (APS) as a supplementary metric for plausibility assessment.

4.2 Results and Analysis

Table 4 presents results from **o-mega** optimization which should provide clear guidance for practitioners. Occlusion is the recommended explainability method for this semantic matching task, achieving the best overall performance (0.819 average) by balancing technical accuracy with human interpretability. This automated recommendation eliminates the need for manual trial-and-error testing of different explanation methods. The results reveal that while most methods can accurately capture model behavior (faithfulness >0.94), they vary significantly in producing explanations that users can easily understand (plausibility 0.35-0.68). This means that method selection critically impacts user experience. Occlusion provides best explanations that are both technically correct and accessible to non-experts, while alternatives like ConservativeLRP (0.641) may confuse users despite being technically valid. The optimization process successfully identified the configuration that maximizes both explanation quality dimensions, providing practitioners with a data-driven recommendation rather than relying on popularity or default settings of explanation methods.

Table 5 depicts comparison of single objective samplers. Results reveals TPESampler as the most efficient optimization algorithm for this task, completing the same optimization quality in significantly less time compared to alternatives. While all three samplers identify Saliency as the optimal method with identical performance scores, TPESampler demonstrates superior computational efficiency, requiring only 14 trials versus BruteForceSampler's 35 trials to achieve the same result. BruteForceSampler provides the most thorough exploration but at a substantial computational cost (1.112 hours), making it impractical for real-world applications. GPSampler offers a middle ground with moderate efficiency (0.672 hours, 14 trials) but shows higher memory usage (760MB against 748MB for TPESampler) and experiences duplicate trials, indicating less efficient search space exploration. The main recommendation from the optimization run comparison is that the TPESampler represents the optimal choice, delivering the same explanation quality recommendations as exhaustive search methods while reducing optimization time

Methods	Faithfulness	Plausibility	Average
Occlusion	0.963	**0.675**	**0.819**
Gradient Shap	**0.967**	0.627	0.797
Saliency	0.966	0.621	0.794
GAE_Explain	0.959	0.615	0.787
Lime	0.962	0.599	0.781
Feature Ablation	0.960	0.597	0.779
Occlusion word-level	0.946	0.605	0.776
Kernel Shap	0.952	0.563	0.757
Guided Backprop	0.943	0.516	0.730
Input X Gradient	0.935	0.497	0.716
ConservativeLRP	0.927	0.354	0.641

Table 4: Comparison of Methods with Faithfulness, Plausibility, and Average scores.

by approximately 66%, making automated XAI selection feasible for routine deployment. Lastly, Table 6 shows the multi-objective optimization results. It shows that BruteForceSampler and TPESampler both correctly identify Occlusion as the optimal method, while NSGA-II and NSGA-III variants select the inferior Saliency method. BruteForceSampler achieves the highest performance scores (0.957 faithfulness, 0.646 plausibility) but requires 35 trials and 1.121 hours, while TPESampler delivers nearly identical results (0.958 faithfulness, 0.622 plausibility) with 70% less computation time (14 trials, 0.339 hours). The NSGA variants show identical performance, indicating no benefit from the increased complexity of NSGA-III. For practitioners, TPESampler is the recommended choice, providing the correct method selection with optimal efficiency for multi-objective explainability optimization.

Single objective	BruteForceSampler	GPSampler	TPESampler
method_best	Saliency	Saliency	Saliency
overall_score	0.784	0.78	0.784
best_find_at	13	5	7
peak memory usage (MB)	936.53	760.92	**748.43**
time (hours)	1.112	0.672	**0.379**
number_dup	0	**1**	4
all_trials	35	14	14

Table 5: Comparison of single objective samplers' performance. The best performances are depicted in bold.

Metric	BruteForceSampler	TPESampler	NSGAIISampler	NSGAIIISampler
Total Trials	35	**14**	**14**	**14**
Best Method	Occlusion	Occlusion	Saliency	Saliency
Faithfulness	0.957	**0.958**	0.957	0.957
Plausibility	**0.646**	0.622	0.606	0.606
Peak Memory (MB)	907.08	761.84	**747.27**	749.35
Time (hours)	1.121	0.339	**0.261**	**0.261**
Duplicates	0	5	7	7

Table 6: Comparison of Multi-Objective Samplers' Performance

5 Conclusion

This work presents **o-mega**, an automated hyperparameter optimization tool for explainable AI method selection in semantic matching and text classification tasks. Our evaluation within seman-

tic matching demonstrates that automated optimization successfully identifies optimal XAI configurations, with Occlusion emerging as the best-performing method for post-claim matching applications. By automating the complex process of XAI method selection and configuration, o-mega enables users to deploy transparent AI systems without requiring deep expertise in explainability techniques. This work has demonstrated the possibility of making explainability more accessible and frictionless for real-world applications, particularly in critical domains such as automated fact-checking and disinformation detection.

Limitations

The current implementation of o-mega has several limitations:

- The tool was developed specifically for claim matching as an enhancer for a specific dataset - MultiClaim, although the text classification pipeline is also available, it is still limiting its immediate applicability to other domains
- Evaluation metrics are currently restricted to only established ones, and tools do not include other experimental metric evaluations, such as localization and sparseness
- Computationally expensive explanation methods have not been incorporated yet, but this is also partially desired since high computational resource requirements render the methods less effective, therefore users are less likely to use them

Acknowledgments

We would like to thank all reviewers for their insightful comments and feedback. This work was supported by DisAI - Improving scientific excellence and creativity in combating disinformation with artificial intelligence and language technologies, a project funded by European Union under the Horizon Europe, GA No. 101079164. This work was also partially funded by European Union, under the project lorAI - Low Resource Artificial Intelligence, GA No. 101136646.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2024. Optuna: A next-generation hyperparameter optimization framework. *Optuna Documentation*. Accessed: 2025-07-01.

Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. 2023. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information fusion*, 99:101805.

David Alvarez-Melis and Tommi S Jaakkola. 2018. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. Generating fact checking explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7352–7364, Online. Association for Computational Linguistics.

Giuseppe Attanasio, Eliana Pastor, Chiara Di Bonaventura, and Debora Nozza. 2023. ferret: a framework for benchmarking explainers on transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Robin Cugny, Julien Aligon, Max Chevalier, Geoffrey Roman Jimenez, and Olivier Teste. 2022. Autoxai: A framework to automatically select the most adapted xai solution. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 315–324, New York, NY, USA. Association for Computing Machinery.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2020. Eraser: A benchmark to evaluate rationalized nlp models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458.

Matthias Feurer and Frank Hutter. 2019. *Hyperparameter optimization*. Springer International Publishing.

Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622.

Anna Hedström, Leander Weber, Daniel Krakowczyk, Dilyara Bareeva, Franz Motzkus, Wojciech Samek, Sebastian Lapuschkin, and Marina M-C Höhne. 2023. Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and

beyond. *Journal of Machine Learning Research*, 24(34):1–11.

Ashkan Kazemi, Kiran Garimella, Devin Gaffney, and Scott A. Hale. 2021. Claim matching beyond English to scale global fact-checking. 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 4504–4517, Online. Association for Computational Linguistics.

Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1096.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Aniek F Markus, Jan A Kors, and Peter R Rijnbeek. 2021. The role of explainability in creating trustworthy artificial intelligence for health care: a comprehensive survey of the terminology, design choices, and evaluation strategies. *Journal of biomedical informatics*, 113:103655.

Sina Mohseni, Niloofar Zarei, and Eric D Ragan. 2021. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 11(3-4):1–45.

Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice Van Keulen, and Christin Seifert. 2023. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Computing Surveys*, 55(13s):1–42.

Qiwei Peng, Robert Moro, Michal Gregor, Ivan Srba, Simon Ostermann, Marian Simko, Juraj Podroužek, Matúš Mesarčík, Jaroslav Kopčan, and Anders Søgaard. 2025. Semeval-2025 task 7: Multilingual and crosslingual fact-checked claim retrieval. *arXiv preprint arXiv:2505.10740*.

Matúš Pikuliak, Ivan Srba, Robert Moro, Timo Hromadka, Timotej Smoleň, Martin Melišek, Ivan Vykopal, Jakub Simko, Juraj Podroužek, and Maria Bielikova. 2023. Multilingual previously fact-checked claim retrieval. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16477–16500, Singapore. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Anders Søgaard. 2021. *Explainable natural language processing*. Morgan & Claypool Publishers.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855.

Nguyen Vo and Kyumin Lee. 2020. Where are the facts? searching for fact-checked information to alleviate the spread of fake news. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7717–7731, Online. Association for Computational Linguistics.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer.

Yao Zhou, Haonan Wang, Jingrui He, and Haixun Wang. 2021. From intrinsic to counterfactual: On the explainability of contextualized recommender systems. *arXiv preprint arXiv:2110.14844*.

Appendix: Method Hyperparameters

Methods	Hyperparameters
Occlusion	sliding_window_shapes:(5, 1024), strides:(1, 1024)
Gradient Shap	stdevs:0.1, n_samples:15
Saliency	abs: True
GAE_Explain	
Lime	n_samples:90 ,similarity_func:{'function_name': 'get_exp_kernel_similarity_function', 'parameters': {'distance_mode': 'euclidean', 'kernel_width': 750}}, interpretable_model:{'function_name': 'SkLearnLasso', 'parameters': {'alpha': 1e-10}}
Feature Ablation	
Occlusion_word_level	regex_condition: ".,!?;:…"
Kernel Shap	n_samples:90
Guided Backprop	
Input X Gradient	
ConservativeLRP	

Table 7: Complete list of method hyperparameters

```
model_param:
   Lime:
     similarity_func:
       function_name:
         - captum.attr._core.lime.
             get_exp_kernel_similarity_function
       parameters:
         distance_mode: ["cosine", "euclidean"]
         kernel_width: [450, 750]
     interpretable_model:
       function_name:
         - captum._utils.models.linear_model.
             SkLearnLasso
       parameters:
         alpha: [1e-19, 1e-25]
   GAE_Explain:
     implemented_method: true
     layers:
       module_path_expressions:
         - "hf_transformer.encoder.layer.*.
             attention.self.dropout"
   ConservativeLRP:
     implemented_method: true
     layers:
       store_A_path_expressions:
         - "hf_transformer.embeddings"
       attent_path_expressions:
         - "hf_transformer.encoder.layer.*.
             attention.self.dropout"
       norm_layer_path_expressions:
         - "hf_transformer.embeddings.LayerNorm"
         - "hf_transformer.encoder.layer.*.
             attention.output.LayerNorm"
         - "hf_transformer.encoder.layer.*.output.
             LayerNorm"
   Occlusion_word_level:
     implemented_method: true
```

Figure 6: Configuration block specifying the model parameters for hyperoptimization and evaluation of explanations

```
method_param:
  Lime:
    parameters:
      n_samples: [80, 90]
    token_groups_for_feature_mask: true
  Saliency:
    parameters:
      abs: [true, false]
  Occlusion:
    parameters:
      sliding_window_shapes:
        - [3, 1024]
        - [5, 1024]
      strides:
        - [1, 1024]
        - [1, 512]
    compute_baseline: true
  Gradient Shap:
    parameters:
      stdevs: [0.1, 0.9]
      n_samples: [10, 15]
    compute_baseline: true
  Kernel Shap:
    parameters:
      n_samples: [80, 90]
    compute_baseline: true
  Feature Ablation:
    token_groups_for_feature_mask: true
    compute_baseline: true
  Occlusion_word_level:
    parameters:
      regex_condition:
        - ""
        - ".,!?;:"
  Integrated Gradients:
    parameters:
      n_steps: [60, 40]
```

Figure 7: Configuration block specifying the method parameters for hyper-optimization and evaluation of explanations

EvoAgentX: An Automated Framework for Evolving Agentic Workflows

Yingxu Wang[1] **Siwei Liu**[2] **Jinyuan Fang**[3] **Zaiqiao Meng**[3*]
[1]Mohamed bin Zayed University of Artificial Intelligence [2]University of Aberdeen
[3]University of Glasgow
yingxv.wang@gmail.com, siwei.liu@abdn.ac.uk
j.fang.2@research.gla.ac.uk, zaiqiao.meng@glasgow.ac.uk

Abstract

Multi-agent systems (MAS) have emerged as a powerful paradigm for orchestrating large language models (LLMs) and specialized tools to collaboratively address complex tasks. However, existing MAS frameworks often require manual workflow configuration and lack native support for dynamic evolution and performance optimization. In addition, many MAS optimization algorithms are not integrated into a unified framework. In this paper, we present **EvoAgentX**, an open-source platform that automates the generation, execution, and evolutionary optimization of multi-agent workflows. EvoAgentX employs a modular architecture consisting of five core layers: the basic components, agent, workflow, evolving, and evaluation layers. Specifically, within the evolving layer, EvoAgentX integrates three MAS optimization algorithms, TextGrad, AFlow, and MIPRO, to iteratively refine agent prompts, tool configurations, and workflow topologies. We evaluate EvoAgentX on HotPotQA, MBPP, and MATH for multi-hop reasoning, code generation, and mathematical problem solving, respectively, and further assess it on real-world tasks using GAIA. Experimental results show that EvoAgentX consistently achieves significant performance improvements, including a 7.44% increase in HotPotQA F1, a 10.00% improvement in MBPP pass@1, a 10.00% gain in MATH solve accuracy, and an overall accuracy improvement of up to 20.00% on GAIA. The source code is available at: https://github.com/EvoAgentX/EvoAgentX.

1 Introduction

Multi-agent systems (MAS) are emerging as a powerful paradigm for orchestrating large language models (LLMs) and specialized tools to solve complex tasks collaboratively (Hong et al., 2023; Gao et al., 2024). By coordinating multiple agents with distinct capabilities, such as planning, reasoning, or code generation, MAS decompose intricate problems into controllable subtasks and assign them to agents capable of solving them (Yuan et al., 2024; Zhang et al., 2025a). This flexible and modular architecture makes MAS well-suited for addressing complex real-world problems. As a result, MAS have been widely deployed in applications such as multi-hop question answering (Hong et al., 2023), software engineering automation (Li et al., 2023), code generation (Liu et al., 2025), mathematical problem solving (Gao et al., 2024), and dialogue systems (Shi et al., 2024).

Despite these promising developments, constructing a multi-agent system typically requires manual efforts in defining the roles of agents, specifying task decomposition strategies, designing agent interactions, and configuring execution workflows (Tang et al., 2024; Xiao et al., 2024). Frameworks like CrewAI [1], CAMEL AI (Li et al., 2023), and LangGraph [2] have provided general-purpose frameworks for building such multi-agent systems, but they primarily rely on hand-crafted configurations or rule-based orchestration. This reliance limits scalability and usability, especially when adapting workflows to new tasks or domains. As the complexity of multi-agent systems grows, there is an urgent need for automating workflow construction to reduce manual efforts and facilitate the rapid development of agent-based applications.

Another challenge lies in enabling MAS to evolve and optimize themselves dynamically rather than relying on static, predefined configurations (Zhang et al., 2025a). Real-world tasks, such as multi-hop question answering or software debugging, often involve changing inputs, intermediate outcomes, or increasing task complexities (Xu et al., 2024). These characteristics make it essential for multi-agent systems to support workflow

* Corresponding author.

[1]https://www.crewai.com/
[2]https://www.langchain.com/langgraph

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 643–655
November 4-9, 2025

evolution and optimization to continuously adapt strategies to meet task requirements and respond to varying conditions. By adjusting workflow topologies, modifying prompt templates, and choosing the most suitable tools, MAS can iteratively refine their workflows for improved problem-solving effectiveness (Zhou et al., 2024). However, most existing MAS frameworks lack mechanisms for dynamic workflow evolution or optimization. Although some multi-agent optimization methods like DSPy (Khattab et al., 2023) and TextGrad (Yuksekgonul et al., 2024) provide approaches for prompt refinement or agent orchestration, they remain fragmented and are not integrated into unified platforms, making it difficult for practitioners to apply and compare them consistently and effectively across diverse tasks.

To this end, we propose **EvoAgentX**, an open-source platform designed for the automated generation and evolutionary optimization of multi-agent workflows. As illustrated in Figure 1, EvoAgentX employs a modular architecture consisting of five core layers: basic components, agent, workflow, evolving, and evaluation. The central feature of EvoAgentX is the evolving layer, which seamlessly integrates three state-of-the-art optimization algorithms, TextGrad (Yuksekgonul et al., 2024), AFlow (Zhang et al., 2024b), and MIPRO (Opsahl-Ong et al., 2024), to iteratively refine agent prompts, tool configurations, and workflow topologies. We evaluate EvoAgentX comprehensively across diverse benchmarks, including HotPotQA for multi-hop reasoning (Yang et al., 2018), MBPP for code generation (Austin et al., 2021a), MATH for mathematical problem-solving (Hendrycks et al., 2021), and the GAIA benchmark for real-world multi-agent tasks (Mialon et al., 2023). Experimental results demonstrate that EvoAgentX achieves substantial performance improvements through dynamic workflow evolution, including improvements of 7.44% in HotPotQA F1, 10.00% in MBPP pass@1, 10.00% in MATH solve accuracy, and up to 20.00% overall accuracy on GAIA.

The contributions of our demo are as follows:

- We present EvoAgentX, an open-source platform that enables easy customization and automatic generation of multi-agent workflows from high-level goal descriptions, reducing manual efforts in system design.
- EvoAgentX integrates three optimization algorithms, TextGrad, AFlow, and MIPRO, to enable evolutionary workflow optimization.
- EvoAgentX provides built-in benchmarks and standardized evaluation metrics, supporting dynamic workflow evolution that consistently improves performance on datasets such as HotPotQA, MBPP, and MATH, as well as on real-world benchmarks like GAIA.

2 Related Work

2.1 Multi-Agent Systems

Recent research on multi-agent systems (MAS) has explored how multiple language model agents can collaborate to solve complex tasks by distributing subtasks across specialized components (Wang et al., 2024; Kapoor et al., 2024). Frameworks such as CAMEL AI, CrewAI, and LangGraph provide general-purpose infrastructures that allow users to define agents with distinct roles, specify communication protocols, and design execution logic and interaction patterns. These systems have enabled significant progress in areas such as multi-hop reasoning, dialogue simulation, tool-augmented reasoning, and software engineering automation (Ma et al., 2024). However, most existing MAS frameworks rely on hand-crafted workflows or rule-based orchestration, requiring users to manually predefine agent hierarchies, interactions, and execution strategies for each task (Gao et al., 2024; Islam et al., 2024). This places a substantial burden on users and limits the scalability and adaptability of such systems, particularly when workflows need to be adjusted for new tasks or changing conditions. To address this limitation, EvoAgentX introduces automatic workflow generation from high-level task descriptions, eliminating the need for manual workflow design and significantly reducing human effort in building multi-agent systems.

2.2 Multi-Agent Optimization

Recent work on multi-agent optimization aims to enhance task performance by refining agent prompts, execution strategies, and communication structures (Agarwal et al., 2024; Zhou et al., 2025). Early studies focused on prompt optimization, with methods such as DSPy (Khattab et al., 2023) and TextGrad (Yuksekgonul et al., 2024) demonstrating that prompt-level refinement improves orchestration efficiency. In parallel, frameworks like DyLAN (Liu et al., 2023) and Captain Agent (Song et al., 2024) explored dynamic

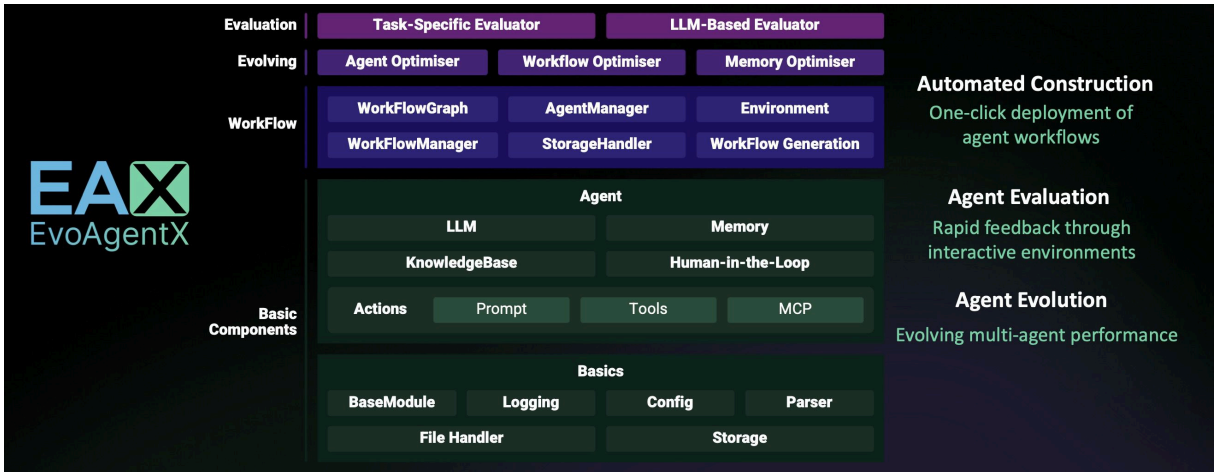


Figure 1: The framework of EvoAgentX. It illustrates the modular architecture including the basic components, agent, workflow, evolving, and evaluation layers.

topology adjustments through reactive modifications to agent sets or communication links. A significant advancement was achieved when multi-agent topology was formalized as a search and optimization problem. Approaches such as AutoFlow (Li et al., 2024) and GPTSwarm (Zhuge et al., 2024) defined topological search spaces using natural language programs or computational graphs and applied reinforcement learning to optimize agent connections. Building on this, ScoreFlow (Wang et al., 2025) and G-Designer (Zhang et al., 2024a) introduced preference learning and deep generative models to produce task-adaptive communication structures. Recent work has further unified prompt and topology optimization, as in ADAS (Hu et al., 2024), FlowReasoner (Gao et al., 2025), and MaAS (Zhang et al., 2025b), which jointly optimize agent configurations, prompts, and execution graphs using meta-search, reinforcement learning, or probabilistic supernets. Despite these advances, most existing frameworks rely on fragmented toolchains or require manual setup, making them difficult to apply consistently across diverse tasks. EvoAgentX addresses this gap by providing an integrated platform that automates multi-agent workflow generation and unifies optimization techniques in a single end-to-end system.

3 System Design

EvoAgentX is an open-source framework designed to automate the generation, execution, evaluation, and evolutionary optimization of agentic workflows. It supports automatic multi-agent workflow generation from high-level task descriptions, seamless integration of evolutionary optimization algorithms such as TextGrad (Yuksekgonul et al., 2024), AFlow (Zhang et al., 2024b), and MIPRO (Opsahl-Ong et al., 2024), and built-in benchmarks with standardized evaluation metrics for systematic performance assessment. To enable these capabilities, as shown in Figure 1, EvoAgentX adopts a modular architecture comprising five core layers: the basic components, agent, workflow, evolving, and evaluation layers.

3.1 Basic Component Layer

The basic components layer forms the foundation of EvoAgentX, providing essential services that ensure the system's stability, scalability, and extensibility. It simplifies infrastructure management, supporting high-level agent design and workflow construction. Core modules include configuration management, logging, file handling, and storage. The configuration manager validates system parameters from structured files (e.g., YAML or JSON), while the logging module tracks system events and performance metrics. File handling components manage workflow states and agent checkpoints, ensuring experiment reproducibility. The storage manager supports both persistent and temporary storage, including caching and checkpointing.

To further enhance the system's flexibility and adaptability, EvoAgentX integrates with a variety of LLMs through frameworks such as OpenRouter [3] and LiteLLM [4] in the basic component layer, enabling seamless integration of LLMs from diverse sources.

[3] https://openrouter.ai/
[4] https://www.litellm.ai/

3.2 Agent Layer

The agent layer serves as the core functional unit of EvoAgentX, enabling the construction of modular, intelligent entities that integrate reasoning, memory, and action execution capabilities in a seamless and flexible manner. Each agent is designed as a composition of a LLM, action modules, and memory components, together supporting flexible, context-aware decision-making and tools.

At the center of the agent architecture, the LLM is responsible for high-level reasoning, response generation, and context interpretation. It is specified through direct instantiation or configuration files and serves as the foundation for all agent operations. Actions define the operational logic of agents. Each action encapsulates a specific task (e.g., summarization, retrieval, API invocation) and consists of a prompt template, input-output format specifications, and optional tool integrations. Formally, an agent a_i is represented as:

$$a_i = \langle \text{LLM}_i, \text{Mem}_i, \{\text{Act}_i^{(j)}\}_{j=1}^{M} \rangle, \quad (1)$$

where Mem_i denotes the memory module, and $\text{Act}_i^{(j)}$ denotes the set of action components.

An example in A.1 illustrates how to create an agent within EvoAgentX.

3.3 Workflow Layer

The workflow layer is another core component of EvoAgentX, supporting the construction, orchestration, and execution of multi-agent workflows in a structured and flexible manner. It provides a formal representation for capturing task dependencies, execution flows, and communication between agents. Each workflow is modeled as a directed graph:

$$\mathcal{W} = (\mathcal{V}, \mathcal{E}), \quad (2)$$

where $\mathcal{V}$ denotes the set of nodes (tasks) and $\mathcal{E}$ represents directed edges encoding dependencies and data flow between tasks. Each node $v \in \mathcal{V}$ corresponds to a `WorkFlowNode`, defining a specific task, its inputs, outputs, associated agents, and execution status (`PENDING`, `RUNNING`, `COMPLETED`, or `FAILED`). Nodes can encapsulate either a set of agents, allowing dynamic selection of optimal actions during execution, or an ActionGraph specifying an explicit sequence of operations. Edges $(v_i, v_j) \in \mathcal{E}$ capture task dependencies, execution order, and optional priority weights for scheduling.

The workflow layer supports both general-purpose workflows (WorkFlowGraph) and streamlined linear workflows (SequentialWorkFlowGraph). The former provides a flexible framework for explicitly defining complex task graphs, including custom nodes, edges, conditional branches, and parallel execution patterns. It allows users to specify detailed task dependencies and exercise fine-grained control over data flow and execution logic. In contrast, the latter is designed for simplicity, automatically inferring graph connections based on task input-output dependencies and generating nodes, agents, and edges without the need for manual graph specification. This dual design facilitates rapid prototyping while preserving the expressiveness needed to model complex structures.

An example in A.2 illustrates how to create a workflow within EvoAgentX.

3.4 Evolving Layer

The evolving layer of EvoAgentX consists of three core components: agent optimizer, workflow optimizer, and memory optimizer. These optimizers provide a unified mechanism for iteratively refining agent configurations, workflow topologies, and memory management strategies. This architecture enables the system to dynamically adapt to changing task requirements, optimize multi-agent coordination, and improve overall performance.

(1) The agent optimizer aims to refine agent prompt templates, tool configurations, and action strategies to enhance each agent's performance across diverse tasks. Formally, for an agent a_i parameterized by its prompt Prompt_i and configuration θ_i, the optimizer seeks to compute

$$(\text{Prompt}_i^{(t+1)}, \theta_i^{(t+1)}) = \mathcal{O}_{\text{agent}}(\text{Prompt}_i^{(t)}, \theta_i^{(t)}, \mathcal{E}), \quad (3)$$

where $\mathcal{O}_{\text{agent}}(\cdot)$ denotes the agent-level optimization operator that updates prompts and configurations based on evaluation feedback, and $\mathcal{E}$ denotes evaluation feedback. The TextGrad (Yuksekgonul et al., 2024) and MIPRO (Opsahl-Ong et al., 2024) optimizers are employed for agent optimization, jointly applying gradient-based prompt tuning, in-context learning, and preference-guided refinement to iteratively align prompts, tool configurations, and agent outputs with task-specific objectives based on the performance signal.

(2) The workflow optimizer focuses on improving task decomposition and execution flow by adjusting the structure of the workflow graph $\mathcal{W} =$

Table 1: Statistics of different benchmarks. We denote Question Answering and Natural Questions as QA and NQ, respectively.

Task	Dataset Name	# Train	# Dev	# Test
QA	NQ	79,168	8,757	3,610
Multi-Hop QA	HotPotQA	90,447	7,405	/
Math	GSM8K	7,473	/	1,319
Math	MATH	7,500	/	5,000
Code Generation	HumanEval	/	/	164
Code Generation	MBPP	/	/	427
Code Generation	LiveCodeBench (v1∼v5)	/	/	400∼880
Code Execution	LiveCodeBench	/	/	479
Test Output Prediction	LiveCodeBench	/	/	442

$(\mathcal{V}, \mathcal{E})$. Formally, its objective is to compute

$$\mathcal{W}^{(t+1)} = \mathcal{O}_{\text{workflow}}(\mathcal{W}^{(t)}, \mathcal{E}), \tag{4}$$

where $\mathcal{O}_{\text{workflow}}(\cdot)$ denotes the workflow-level optimization operator that updates the graph structure based on evaluation feedback, and $\mathcal{E}$ denotes evaluation feedback. The SEW (Liu et al., 2025) and AFlow (Zhang et al., 2024b) optimizers are employed for workflow optimization, iteratively restructuring workflow graphs by reordering nodes, modifying dependencies, and exploring alternative execution strategies guided by the task performance signal and convergence criteria.

(3) The memory optimizer remains under active development. Its objective is to provide structured, persistent memory modules $\mathcal{M}_i$ that enable selective retention, dynamic pruning, and priority-based retrieval (Zeng et al., 2024). Formally, it aims to compute

$$\mathcal{M}_i^{(t+1)} = \mathcal{O}_{\text{memory}}(\mathcal{M}_i^{(t)}, \mathcal{E}), \tag{5}$$

where $\mathcal{O}_{\text{memory}}(\cdot)$ denotes the memory-level optimization operator that updates the agent's memory module based on evaluation feedback, and $\mathcal{E}$ denotes evaluation feedback.

An example in A.3 demonstrates how to use the optimizer within EvoAgentX.

3.5 Evaluation Layer

The evaluation layer of EvoAgentX provides a modular and extensible framework for systematically assessing workflow performance across tasks and benchmarks. It integrates two complementary components: (1) the task-specific evaluator and the LLM-based evaluator. The task-specific evaluator computes domain-relevant metrics by comparing workflow outputs against ground truth labels, supporting validation on datasets such as HotPotQA, MBPP, and MATH. More details about the benchmarks provided in EvoAgentX are shown in Table 1; (2) the LLM-based evaluator leverages the reasoning and generation capabilities of large language models to deliver flexible and context-aware evaluations. It supports qualitative assessments, consistency checking, and dynamic criteria that are not easily captured by static metrics.

Table 2: Performance comparison across different benchmarks. **Bold** indicates the best performance.

Method	HotPotQA (F1%)	MBPP (Pass@1 %)	MATH (Solve %)
Original	63.58	69.00	66.00
TextGrad	**71.02**	71.00	**76.00**
AFlow	65.09	**79.00**	71.00
MIPRO	69.16	68.00	72.30

Formally, given a workflow $\mathcal{W}$ and dataset $\mathcal{D}$, the evaluation process is defined as

$$\mathcal{P} = \mathcal{T}(\mathcal{W}, \mathcal{D}), \tag{6}$$

where $\mathcal{T}(\cdot)$ maps workflow executions to aggregated performance metrics $\mathcal{P}$. The design supports both WorkFlowGraph and ActionGraph structures, allowing evaluation at various abstraction levels.

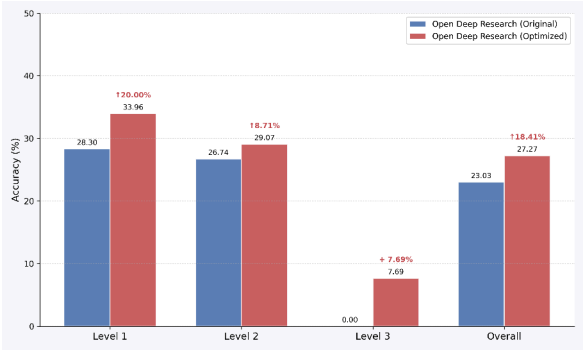
4 Experiments

We evaluate EvoAgentX across three tasks: (1) Evolution Algorithms, which optimize agent configurations and workflow topologies to improve performance; (2) Applications, where EvoAgentX is applied to enhance multi-agent systems on real-world benchmarks; and (3) Case Study, demonstrating EvoAgentX's capability to optimize workflows and enhance agent performance through practical examples.

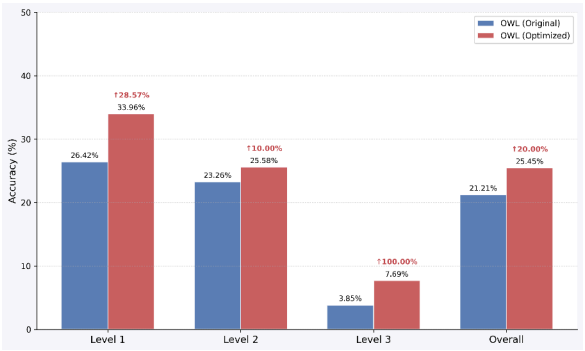
4.1 Evolution Algorithms

Experimental Settings. We evaluate EvoAgentX's evolutionary optimization capabilities by applying three integrated algorithmsm, TextGrad, MIPRO, and AFlow, to iteratively refine agent prompts, tool configurations, and workflow topologies, using metrics such as F1 score for multi-hop reasoning, pass@1 accuracy for code generation, and solve rate for mathematical problem solving.

Datasets. We assess EvoAgentX on HotPotQA for multi-hop question answering requiring compositional reasoning (Yang et al., 2018), MBPP for Python code generation from natural language descriptions(Austin et al., 2021b), and MATH for solving high-school level mathematical problems (Hendrycks et al., 2021).



(a) Performance of Open Deep Research.



(b) Performance of OWL Agent.

Figure 2: Performance improvements of Open Deep Research and OWL Agent on the GAIA benchmark.

Results Analysis. As shown in Table 2, EvoAgentX's optimization algorithms consistently enhance performance across all benchmarks. Specifically, TextGrad substantially improves multi-hop reasoning, increasing the HotPotQA F1 score from 63.58% to 71.02%. AFlow significantly boosts code generation accuracy, raising MBPP pass@1 from 69.00% to 79.00%. Similarly, TextGrad strengthens mathematical reasoning, improving the MATH solve rate from 66.00% to 76.00%. These results demonstrate the EvoAgentX can effectively refine multi-agent workflow topologies to align with task-specific objectives across domains.

4.2 Applications

Experimental Settings. We apply EvoAgentX to optimize existing multi-agent systems on a real-world benchmark. Specifically, We select two representative open-source frameworks from the GAIA leaderboard (Mialon et al., 2023), Open Deep Research [5] and OWL [6], and refine their agent prompts and workflow configurations using EvoAgentX, evaluating performance based on accuracy, which measures the correctness of generated answers against the ground truth.

Results Analysis. As shown in Figure 2, EvoAgentX significantly improves the performance of both Open Deep Research and OWL across all evaluation levels on the GAIA benchmark. For Open Deep Research, the overall accuracy increases by 18.41%, with notable improvements of 20.00% at Level 1, 8.71% at Level 2, and 7.69% at Level 3. Similarly, OWL achieves an overall accuracy improvement of 20.00%, driven by gains of 28.57% at Level 1, 10.00% at Level 2, and a remarkable 100.00% at Level 3. These results demonstrate the effectiveness of EvoAgentX in enhancing real-world multi-agent systems through automated prompt and topology optimization.

4.3 Case Study

We present a case study to illustrate the practical application of EvoAgentX in refining agent prompts and workflow configurations within existing multi-agent systems, including examples from AFlow, TextGrad, and MIPRO. More detailed analysis and results are presented in Appendix A.4.

5 Conclusion

We present **EvoAgentX**, an open-source platform that automates the generation, execution, evaluation and optimization of multi-agent workflows. It addresses key limitations of existing frameworks by eliminating the need for manual workflow design and providing support for dynamic, task-specific optimization. By integrating multiple optimization algorithms, including TextGrad, AFlow, and MIPRO, EvoAgentX enables the iterative refinement of agent prompts, tool configurations, and workflow topologies with minimal manual intervention. Experiments on diverse benchmarks, such as HotPotQA, MBPP, MATH and GAIA, demonstrate that EvoAgentX consistently achieves substantial performance improvements in multi-hop reasoning, code generation, mathematical problem solving, and real-world multi-agent applications. In the future, we will extend **EvoAgentX** with more optimization algorithms, richer tool integration, and long-term memory to further enhance agent adaptability and contextual awareness. We also plan to explore advanced evolution strategies (Fang et al., 2025), including MASS (Zhou et al., 2025), AlphaEvolve (Novikov et al., 2025), and Darwin Gödel Machine (Zhang et al., 2025c), to advance the strategies of multi-agent optimization.

[5] https://github.com/langchain-ai/open_deep_research
[6] https://github.com/camel-ai/owl

References

Eshaan Agarwal, Vivek Dani, Tanuja Ganu, and Akshay Nambi. 2024. Promptwizard: Task-aware agent-driven prompt optimization framework. *arXiv preprint arXiv:2405.18369*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021a. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, and 1 others. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, and 1 others. 2024. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*.

Hongcheng Gao, Yue Liu, Yufei He, Longxu Dou, Chao Du, Zhijie Deng, Bryan Hooi, Min Lin, and Tianyu Pang. 2025. Flowreasoner: Reinforcing query-level meta-agents. *arXiv preprint arXiv:2504.15257*.

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. *arXiv preprint arXiv:2103.03874*.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Shengran Hu, Cong Lu, and Jeff Clune. 2024. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*.

Md Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. Mapcoder: Multi-agent code generation for competitive problem solving. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4912–4944.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pages 161–178. Springer.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, and 1 others. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. 2024. Autoflow: Automated workflow generation for large language model agents. *arXiv preprint arXiv:2407.12821*.

Siwei Liu, Jinyuan Fang, Han Zhou, Yingxu Wang, and Zaiqiao Meng. 2025. Sew: Self-evolving agentic workflows for automated code generation. *arXiv preprint arXiv:2505.18646*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. Optimizing instructions and demonstrations for multi-stage language model programs. *arXiv preprint arXiv:2406.11695*.

Juanming Shi, Qinglang Guo, Yong Liao, and Shenglin Liang. 2024. Legalgpt: Legal chain of thought for the legal large language model multi-agent framework. In *International Conference on Intelligent Computing*, pages 25–37. Springer.

Linxin Song, Jiale Liu, Jieyu Zhang, Shaokun Zhang, Ao Luo, Shijian Wang, Qingyun Wu, and Chi Wang. 2024. Adaptive in-conversation team building for language model agents. *arXiv preprint arXiv:2405.19425*.

Xunzhu Tang, Kisub Kim, Yewei Song, Cedric Lothritz, Bei Li, Saad Ezzini, Haoye Tian, Jacques Klein, and Tegawendé F Bissyandé. 2024. Codeagent: Autonomous communicative agents for code review. *arXiv preprint arXiv:2402.02172*.

Jize Wang, Ma Zerun, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. 2024. Gta: a benchmark for general tool agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yinjie Wang, Ling Yang, Guohao Li, Mengdi Wang, and Bryon Aragam. 2025. Scoreflow: Mastering llm agent workflows via score-based preference optimization. *arXiv preprint arXiv:2502.04306*.

Yihang Xiao, Jinyi Liu, Yan Zheng, Xiaohan Xie, Jianye Hao, Mingzhi Li, Ruitao Wang, Fei Ni, Yuxiao Li, Jintian Luo, and 1 others. 2024. Cellagent: An llm-driven multi-agent framework for automated single-cell data analysis. *arXiv preprint arXiv:2407.09811*.

Yao Xu, Shizhu He, Jiabei Chen, Zihao Wang, Yangqiu Song, Hanghang Tong, Guang Liu, Kang Liu, and Jun Zhao. 2024. Generate-on-graph: Treat llm as both agent and kg in incomplete knowledge graph question answering. *arXiv preprint arXiv:2404.14741*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. 2024. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. *arXiv preprint arXiv:2406.14228*.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. 2024. On the structural memory of llm agents. *arXiv preprint arXiv:2412.15266*.

Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. 2025a. Evoflow: Evolving diverse agentic workflows on the fly. *arXiv preprint arXiv:2502.07373*.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025b. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2024a. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. 2025c. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, and 1 others. 2024b. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö Arık. 2025. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533*.

Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, and 1 others. 2024. Symbolic learning enables self-evolving agents. *arXiv preprint arXiv:2406.18532*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*.

A Appendix

A.1 Creating a Simple Agent in EvoAgentX

In EvoAgentX, a simple agent can be created using the `CustomizeAgent` class, which enables the quick configuration of agents with a specific prompt. To create such an agent, the first step is to configure the large language model (LLM) that will be used by the agent. This is done by defining the LLM settings, such as the model type and API key, using the `OpenAILLMConfig` class. After configuring the LLM, the agent is instantiated by specifying its name, description, and prompt, which defines the task the agent will perform.

The following code demonstrates the process of creating and using a simple agent in EvoAgentX, where the agent is tasked with printing "hello world." The agent is then executed, and the result is retrieved as a message object, with the content of the response extracted and displayed.

```
from evoagentx.models import
    OpenAILLMConfig
from evoagentx.agents import
    CustomizeAgent

# Configure LLM
openai_config = OpenAILLMConfig(
    model="gpt-4o-mini",
    openai_key="YOUR_API_KEY",
    stream=True
)

# Create a simple agent
first_agent = CustomizeAgent(
    name="FirstAgent",
    description="A simple agent that
        prints 'hello world'",
    prompt="Print 'hello world'",
    llm_config=openai_config
)

# Execute the agent
message = first_agent()
print(f"Response from {first_agent.name
    }: {message.content.content}")
```

This approach showcases how a basic agent can be constructed and executed with minimal configuration in EvoAgentX.

A.2 Creating a Simple Workflow in EvoAgentX

In EvoAgentX, workflows enable multiple agents to collaborate sequentially on tasks. A basic sequential workflow involves defining tasks, each with a name, description, input-output specifications, and a prompt. To create such a workflow, the `SequentialWorkFlowGraph` class is used to define and link tasks. An agent manager oversees the agents responsible for executing each task. The workflow is executed by providing the necessary inputs, and the results are collected as outputs.

The following code illustrates the process of creating and using a simple workflow in EvoAgentX, where the workflow includes two tasks: "Planning," in which the agent creates a detailed implementation plan for a given problem, and "Coding," in which the agent implements the solution based on the plan. The workflow is then executed with a specified problem, and the results are retrieved as outputs, with the content of each task's result extracted and displayed sequentially.

```
import os
from dotenv import load_dotenv
from evoagentx.workflow import
    SequentialWorkFlowGraph
from evoagentx.agents import
    AgentManager
from evoagentx.models import
    OpenAILLMConfig, OpenAILLM

# Load environment variables
load_dotenv()
OPENAI_API_KEY = os.getenv("
    OPENAI_API_KEY")

# Configure the LLM
llm_config = OpenAILLMConfig(model=
"gpt-4o-mini", openai_key=OPENAI_API_KEY
    , stream=True)
llm = OpenAILLM(llm_config)

# Define tasks in the sequential
    workflow
tasks = [
    {
        "name": "Planning",
        "description": "Create a
            detailed plan for code
            generation",
        "inputs": [{"name": "problem", "
            type": "str", "required":
            True}],
        "outputs": [{"name": "plan", "
            type": "str", "required":
            True}],
        "prompt": "You are a software
            architect. Create a detailed
             implementation plan for the
             given problem.\n\nProblem:
            {problem}",
        "parse_mode": "str"
    },
    {
        "name": "Coding",
        "description": "Implement the
            code based on the plan",
        "inputs": [{"name": "problem", "
            type": "str", "required":
            True}],
        "outputs": [{"name": "code", "
            type": "str", "required":
            True}],
        "prompt": "You are a developer.
```

```
            Implement the code based on
            the provided plan.\n\
            nProblem: {problem}\
            nImplementation Plan: {plan}
            ",
        "parse_mode": "str"
    }
]

# Create the sequential workflow graph
graph = SequentialWorkFlowGraph(goal=
"Generate code to solve programming
    problems", tasks=tasks)

# Initialize the agent manager and add
    agents
agent_manager = AgentManager()
agent_manager.add_agents_from_workflow
(graph, llm_config=llm_config)

# Create the workflow instance
workflow = WorkFlow(graph=graph,
    agent_manager=agent_manager, llm=llm
    )

# Execute the workflow with inputs
output = workflow.execute(inputs={"
    problem": "Write a function to find
    the longest palindromic substring in
     a given string."})

print("Workflow completed!")
print("Workflow output:\n", output)
```

This example demonstrates how a simple sequential workflow can be created and executed in EvoAgentX, where agents collaborate in a defined sequence to accomplish a task.

A.3 A Simple Example about using Optimizer within EvoAgentX

In EvoAgentX, the AFlow optimizer optimizes multi-agent workflows for tasks like code generation. To use the optimizer, the first step is configuring the LLMs for both optimization and execution, one for optimizing the workflow (e.g., Claude 3.5 Sonnet) and one for task execution (e.g., GPT-4o-mini). The task configuration specifies operators (e.g., Custom, CustomCodeGenerate, ScEnsemble), and the workflow is created using the `SequentialWorkFlowGraph` class. The optimizer is initialized with the paths to the workflow, LLM configurations, and optimization parameters.

The following code demonstrates the process of creating and using an AFlow (Zhang et al., 2024b) optimizer in EvoAgentX, where the optimizer refines a multi-agent workflow for code generation. The optimizer is configured with specific settings, including the selection of LLMs for both optimization and execution. The process begins by configuring the `optimizer_llm` for optimizing the workflow and the `executor_llm` for executing the tasks. The optimizer is run with a benchmark (e.g., the HumanEval benchmark), and the results are retrieved as outputs, with each optimization step and its corresponding result displayed sequentially.

```
import os
from dotenv import load_dotenv
from evoagentx.optimizers import
    AFlowOptimizer
from evoagentx.models import
    LiteLLMConfig, LiteLLM,
    OpenAILLMConfig, OpenAILLM
from evoagentx.benchmark import
    AFlowHumanEval

# Load environment variables
load_dotenv()
OPENAI_API_KEY = os.getenv("
    OPENAI_API_KEY")
ANTHROPIC_API_KEY = os.getenv("
    ANTHROPIC_API_KEY")

# Configure LLMs
claude_config = LiteLLMConfig(model=
"anthropic/
claude-3-5-sonnet-20240620",
    anthropic_key=ANTHROPIC_API_KEY)
optimizer_llm = LiteLLM(config=
    claude_config)

openai_config = OpenAILLMConfig(model="
    gpt-4o-mini", openai_key=
    OPENAI_API_KEY)
executor_llm = OpenAILLM(config=
    openai_config)

# Initialize the benchmark
humaneval = AFlowHumanEval()

# Set up the optimizer
optimizer = AFlowOptimizer(
    graph_path=
    "examples/aflow/code_generation",  #
         Path to the initial workflow
        graph
    optimized_path=
    "examples/
    aflow/humaneval/optimized",  # Path
        to save optimized workflows
    optimizer_llm=optimizer_llm,  # LLM
        for optimization
    executor_llm=executor_llm,  # LLM
        for execution
    validation_rounds=3,  # Number of
        validation rounds
    eval_rounds=3,  # Number of
        evaluation rounds
    max_rounds=20,  # Maximum
        optimization rounds
    task_config=
    EXPERIMENTAL_CONFIG["humaneval"]  #
        Task configuration
)

# Optimize and test the workflow
optimizer.optimize(humaneval)
optimizer.test(humaneval)
```

This illustrates how EvoAgentX dynamically optimizes workflows to improve performance through multiple evaluation rounds.

A.4 Case study

A.4.1 AFlow for Workflow Optimization

In this task, we aim to enhance the efficiency and effectiveness of multi-agent workflows for mathematical problem-solving by optimizing workflows using AFlow, as shown in the Example 1 and Example 2.

The initial workflow used a basic agent configuration to solve mathematical problems with a simple prompt. After optimization with AFlow within EvoAgentX, the workflow was significantly enhanced by incorporating detailed problem analysis, Python code generation, and solution refinement through an ensemble approach. The workflow now involves multiple agents for problem analysis, code generation, and solution refinement, improving both accuracy and clarity. This optimization results in a more comprehensive and precise problem-solving process.

```
class Workflow:

    def __init__(
        self,
        name: str,
        llm_config: LLMConfig,
        benchmark: Benchmark
    ):
        self.name = name
        self.llm = create_llm_instance(
            llm_config)
        self.benchmark = benchmark
        self.custom = operator.Custom(
            self.llm)

    async def __call__(self, problem: str
        ):

        """
        Implementation of the workflow
        """

        solution = await self.custom(
            input=problem, instruction =
            prompt_custom.
        SOLVE_MATH_PROBLEM_PROMPT)

        return solution['response']
```

Example 1: The workflow before optimization.

```
SOLVE_MATH_PROBLEM_PROMPT = r"""
Given the math problem, its analysis,
    and a Python code solution, provide
    a detailed step-by-step solution.
    Ensure your explanation is clear and
     thorough. If the code solution is
    relevant, incorporate its logic into
     your explanation.

Problem, Analysis, and Code Solution:
    """

FORMAT_ANSWER_PROMPT = r"""

Given the problem and its solution,
    extract the final numerical answer
    and format it in a box using LaTeX
    notation \boxed{}. Ensure the answer
     is accurate and properly formatted.

Problem and Solution: """

class Workflow:

    def __init__(
        self,
        name: str,
        llm_config: LLMConfig,
        benchmark: Benchmark
    ):
        self.name = name
        self.llm = create_llm_instance(
            llm_config)
        self.benchmark = benchmark
        self.custom = operator.Custom(
            self.llm)
        self.programmer = operator.
            Programmer(self.llm)
        self.sc_ensemble = operator.
            ScEnsemble(self.llm)

    async def __call__(self, problem: str
        ):
        """
        Implementation of the workflow
        """
        analysis = await self.programmer(
            problem=problem, analysis="
            Analyze the math problem and
            provide a step-by-step
            approach to solve it.")

        code_solution = await self.
            programmer(problem=problem,
            analysis=f"Generate Python
            code to solve this problem: {
            problem}")

        solutions = []
        for _ in range(3):
            solution = await self.custom(
                input=problem + f"\
                nAnalysis: {analysis['
                output']}\nCode Solution:
                 {code_solution['output
                ']}", instruction=
            prompt_custom.
            SOLVE_MATH_PROBLEM_PROMPT)
            solutions.
            append(solution['response'])

        best_solution = await self.
        sc_ensemble(solutions=solutions,
            problem=problem)

        final_answer = await self.custom(
            input=f"Problem: {problem}\
```

```
            nSolution: {best_solution['
            response']}", instruction=
        prompt_custom.
            FORMAT_ANSWER_PROMPT)

        return final_answer['response']
```

Example 2: The workflow after AFlow optimization.

A.4.2 TextGrad for Prompt Optimization

In this task, we aim to improve the efficiency and effectiveness of multi-agent workflows for mathematical problem-solving by optimizing prompts using TextGrad, as demonstrated in Example 5 and Example 6.

The initial prompt employed a minimalistic instruction, directing the agent to solve mathematical problems by simply providing the final answer in a boxed format. Following optimization with TextGrad within EvoAgentX, the prompt was substantially refined to support a structured, step-by-step reasoning process. The enhanced prompt guides the agent to assess problem complexity, apply appropriate mathematical principles, and generate executable Python code. It further emphasizes logical coherence, justification of each solution step, and the use of explanatory transitions to enhance interpretability. These improvements lead to increased solution accuracy and significantly improve the clarity and transparency of the reasoning process.

```
"""
Answer the math question. The answer
    should be in box format, e.g., \\
    boxed{{123}}\n
"""
```

Example 3: The prompt before optimization.

```
"""
Begin by assessing the complexity of the
     math problem to determine the
    appropriate level of detail required
    . For complex problems, provide a
    brief introduction to set the
    context and explain the relevance of
     key mathematical concepts. For
    simpler problems, focus on
    delivering a direct and concise
    solution.


Identify and apply relevant mathematical
     properties or theorems that can
    simplify the problem-solving process
    , such as the arithmetic sequence
    property. Prioritize methods that
    offer a concise and efficient
    solution, minimizing unnecessary
    steps while maintaining clarity.


Solve the problem using the most direct
    and appropriate mathematical
    methodologies, ensuring each
    calculation step is accurate.
    Clearly explain the reasoning behind
     each step, enhancing understanding
    by providing brief explanations of
    why specific mathematical properties
     or methods are applicable.


Maintain a smooth and coherent logical
    flow throughout the solution, using
    transitional phrases to connect
    different parts of the problem-
    solving process. Where applicable,
    compare alternative methods to solve
     the problem, discussing the
    benefits of each approach to provide
     a comprehensive understanding.


Encourage the use of visual aids, such
    as diagrams or charts, to illustrate
     complex concepts and enhance
    comprehension when necessary.
    Explicitly state and verify any
    assumptions made during the problem-
    solving process, clarifying why
    certain methodologies are chosen.


Conclude with a verification step to
    confirm the solution's correctness,
    and present the final answer in a
    consistent format, such as \\boxed{{
    answer}}. Ensure that the final
    expression is in its simplest form
    and that all calculations are
    accurate and justified.


Problem: <input>{problem}</input>


"""
```

Example 4: The prompt after TextGrad optimization.

A.4.3 MIPRO for Prompt Optimization

In this task, we aim to improve the efficiency and effectiveness of multi-agent workflows for mathematical problem-solving by optimizing prompts using MIPRO, as demonstrated in Example 5 and Example 6.

The initial prompt was designed to provide a mathematical solution in a simple boxed format, without elaborating on the solution steps or offering detailed explanations. Following optimization with MIPRO within EvoAgentX, the prompt was significantly improved to guide the agent through a comprehensive, step-by-step solution process. The enhanced prompt now incorporates intermediate

steps, clear explanations of relevant mathematical concepts, and a thorough breakdown of the problem-solving approach. Each step is carefully articulated, ensuring a deeper understanding of the solution. This optimization not only improves the accuracy of the solution but also enhances the clarity and transparency of the reasoning process.

```
"""
Answer the math question. The answer
    should be in box format, e.g., \\
    boxed{{123}}\n
"""
```

Example 5: The prompt before optimization.

```
"""
Please solve the following math problem,
     providing a detailed and clear
    solution process for better
    understanding. Ensure that the final
     answer is presented in a boxed
    format, according to the LaTeX
    convention (e.g., \(\boxed{{123}}\))
    .


Make sure to include any necessary
    intermediate steps, calculations, or
     explanations that lead you to the
    final answer.


**Problem:** {problem}


Examples:
1. Problem: The function $f(x)$
    satisfies\n\\[f(x + y) = f(x) + f(y)
     + 2xy\\]for all real numbers $x$
    and $y.$  If $f(1) = 4,$ then find
    $f(8).$
Output:
Setting $x = y,$ we get
[f(2x) = 2f(x) + 2x^2.]Then
\begin{align*}
f(2) &= 2f(1) + 2 \cdot 1^2 = 10,
f(4) &= 2f(2) + 2 \cdot 2^2 = 28,
f(8) &= 2f(4) + 2 \cdot 4^2 = \boxed
    {88}.
\end{align*}

2. Problem: The product $ab = 1200$, $a$
     is an integer, and $b$ is an odd
    integer. What is the largest
    possible value of $b$?
Output:
Factoring out the highest power of 2
    from 1200, we find that $1200=2^4\
    cdot75$. Therefore, the largest
    possible value of $b$ is $\boxed{75}
    $.


3. Problem: What is the product (in base
     10) of the first and last digits of
     the base-6 representation of $682_
    {10}$?
Output:
We begin by converting $682_{10}$ into
    base-6. We see that $6^3=216$ is the
     largest power of 6 that is less
    than 682, and that $3\cdot216=648$
    is the largest multiple of 216 that
    is less than 682. This leaves us
    with a remainder of $682-648=34$,
    which we can express as $5\cdot6
    ^1+4\cdot6^0$. So, $682_{10}=3\cdot6
    ^3+0\cdot6^2+
5\cdot{6^1}+4\cdot6^0=3054_6$. The first
     and last digits are 3 and 4,
    respectively, making the product of
    the two equal to $\boxed{12}$.


4. Problem: Compute $817_9 - 145_9 - 266
    _9$.  Express your answer in base 9.
Output:
$817_9 - 145_9 - 266_9 = 817_9 - (145_9
    + 266_9) = 817_9 - 422_9 = \boxed
    {385_9}$.

"""
```

Example 6: The prompt after MIPRO optimization.

OpenRLHF: An Easy-to-use, Scalable and High-performance RLHF Framework

Jian Hu[*], **Xibin Wu, Wei Shen, Jason Klein Liu**[†], **Zilin Zhu**
Weixun Wang, Songlin Jiang, Haoran Wang, Hao Chen,
Bin Chen, Weikai Fang, Xianyu, Yu Cao, Haotian Xu, Yiming Liu

OPENRLHF Team
janhu9527@gmail.com

Abstract

Large Language Models (LLMs) fine-tuned via Reinforcement Learning from Human Feedback (RLHF) and Reinforcement Learning with Verifiable Rewards (RLVR) significantly improve the alignment of human-AI values and further raise the upper bound of AI capabilities, particularly in reasoning-intensive, long-context Chain-of-Thought (long-CoT) tasks. However, existing RLHF (or RLVR) frameworks commonly face challenges such as inference bottlenecks and complexity barriers, restricting their accessibility for newcomers. To bridge this gap, we introduce **OpenRLHF**, a user-friendly, scalable, and easy-to-learn open-source RLHF framework built upon Ray, vLLM, DeepSpeed, and HuggingFace Transformers, featuring a simplified design, clear code structure, and comprehensive documentation to facilitate entry for researchers and practitioners. Experimental results show that OpenRLHF achieves superior training efficiency with speedups ranging from 1.22× to 1.68× across different model sizes compared to state-of-the-art frameworks, while requiring significantly fewer lines of code for implementation. OpenRLHF is publicly available at https://github.com/OpenRLHF/OpenRLHF, and has already been adopted by leading institutions to accelerate RLHF research and learning.

1 Introduction

Large Language Models (LLMs) fine-tuned via **Reinforcement Learning from Human Feedback (RLHF)** and **Reinforcement Learning with Verifiable Rewards (RLVR)** have markedly advanced human-AI alignment and elevated the upper bound of AI capabilities (Christiano et al., 2017; Stiennon et al., 2020; Guo et al., 2025; Shen et al., 2025). These approaches enable models to better conform to human intentions and values while achieving superior reasoning performance. Notably, models such as GPT-4 (Achiam et al., 2023), DeepSeek-R1 (Guo et al., 2025), and Claude (Askell et al., 2021) excel at complex reasoning tasks by generating detailed step-by-step rationales, commonly referred to as *Chain-of-Thought* (CoT) outputs.

However, RLHF and RLVR training methodologies—especially those employing Proximal Policy Optimization (PPO)—face significant computational challenges. In particular, the inference phase often accounts for over 90% of the total RLHF (or RLVR) runtime, as models need to generate thousands of tokens during each inference step. Consequently, there is an increasing demand for efficient and scalable frameworks that reduce inference overhead and simplify distributed RLHF and RLVR training workflows.

Existing RLHF and RLVR systems such as DeepSpeed-Chat (Yao et al., 2023), TRL (von Werra et al., 2020), and ColossalChat (Li et al., 2023) have made noteworthy progress in distributed computation and memory efficiency. Conversely, industrial-grade solutions like Nemo-aligner (Shen et al., 2024a) and ChatLearn (alibaba, 2017) offer advanced optimizations at the modeling and framework levels. While Verl (Sheng et al., 2024), a framework proposed after our initial development, also provides sophisticated optimizations (e.g., its 3D-Hybrid engine), these industrial solutions generally feature tightly coupled and specialized designs that introduce considerable complexity, steep learning curves, and accessibility barriers for newcomers and academic researchers. Yet, their tightly coupled and specialized designs introduce considerable complexity, steep learning curves, and accessibility barriers for newcomers and academic researchers. Therefore, there remains a pressing need for an RLHF and RLVR framework that balances high performance, scalability, and ease of use—one that is straightforward enough for researchers new to the field, yet adaptable to

[*]Full contributor list available in Appendix C.
[†]Ubiquant

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 656–666
November 4-9, 2025

diverse and evolving workloads.

In this paper, we present **OpenRLHF**, a simple, high-performance, fully open-source framework supporting both RLHF and RLVR, built upon Ray (Liang et al., 2018), vLLM, DeepSpeed (Yao et al., 2023), and HuggingFace Transformers (Wolf et al., 2020). OpenRLHF offers three key contributions:

- **First Ray-Based Open-Source RLHF and RLVR Architecture:** Leveraging Ray's flexible distributed computing primitives, OpenRLHF enables streamlined orchestration and resource management for RLHF and RLVR workflows, significantly simplifying distributed operations and deployments while enhancing usability and flexibility.
- **3D Parallelism with DeepSpeed-ZeRO and Ring Attention:** To enable seamless and efficient scalability for large models, OpenRLHF integrates automatic tensor parallelism (AutoTP) provided by DeepSpeed-ZeRO and implements sequence parallelism via ring attention. This streamlined integration realizes an efficient "3D" parallel strategy—combining tensor, data, and sequence parallelism—without requiring complex engineering or extensive user configuration.
- **First Accelerated CoT Inference with vLLM:** Addressing the critical inference bottleneck in long-chain-of-thought RLHF and RLVR workloads, OpenRLHF incorporates the state-of-the-art vLLM inference engine. This integration accelerates inference through token-level parallel decoding, advanced caching, and optimized dynamic batching, thereby substantially improving overall training runtime efficiency.
- **Asynchronous Dataflow and Remote Engine Interactions:** OpenRLHF supports asynchronous dataflow and remote engine communication to maximize system throughput and resource utilization during distributed RLHF training. In this architecture, rollout engines, actor engines, and remote engines operate independently and communicate via message passing, enabling immediate processing as soon as data becomes available. This design reduces idle time, improves pipeline efficiency, and enhances scalability and flexibility across large distributed GPU clusters. By leveraging asynchronous remote engine interactions, OpenRLHF can be easily extended to support scalable **agent RL** training.

Together, these innovations position OpenRLHF as an accessible yet powerful framework suitable for both efficient experimentation in academic environments and practical deployment scenarios. Already adopted by leading institutions and companies—including CMU, MIT, Microsoft, and HKUST—OpenRLHF demonstrates broad applicability and impact across the research community (see Appendix A for details on the framework's broad impact and adoption). We publicly release OpenRLHF to foster openness, accelerate research, and promote innovation within the RLHF and RLVR ecosystem.

2 Related Work

Reinforcement Learning from Human Feedback and Reinforcement Learning with Verifiable Rewards. Reinforcement Learning from Human Feedback (RLHF) has emerged as a pivotal paradigm for aligning large language models with human preferences (Christiano et al., 2017; Stiennon et al., 2020). The foundational RLHF framework trains a reward model from human preference data and optimizes the language model using reinforcement learning algorithms such as PPO (Schulman et al., 2017). This approach has been successfully deployed in prominent models including InstructGPT (Ouyang et al., 2022), ChatGPT, and GPT-4 (Achiam et al., 2023). Building upon RLHF, Reinforcement Learning with Verifiable Rewards (RLVR) leverages automatically verifiable signals from mathematical verification, code execution, or other automated evaluation mechanisms (Guo et al., 2025; Shen et al., 2025; Cobbe et al., 2021). While PPO remains dominant for its stability (Schulman et al., 2017; Ziegler et al., 2019), alternative approaches such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) have gained attention for computational efficiency. However, both RLHF and RLVR methods require substantial computational resources and sophisticated distributed training strategies. The computational challenges have motivated various optimization techniques, including efficient inference engines (Kwon et al., 2023), memory-efficient training strategies (Rajbhandari et al., 2020), and distributed orchestration frameworks (Yao et al., 2023). Nevertheless, existing solutions often sacrifice either performance for simplicity or accessibility for optimization, creating a gap that OpenRLHF aims to address.

RLHF (RLVR) Frameworks. The computational

complexity of RLHF and RLVR training has driven specialized framework development. General-purpose RL frameworks (Liang et al., 2018; Dhariwal et al., 2017; Shen et al., 2020) designed for small-scale networks fail to address LLM-specific challenges and typically employ multi-controller architectures with complex inter-process communication, creating steep learning curves. RLHF-specific frameworks face significant performance-accessibility trade-offs. Open-source solutions like TRL (von Werra et al., 2020), DeepSpeed-Chat (Yao et al., 2023), and ColossalChat (Li et al., 2023) provide accessible implementations but often lack sophisticated orchestration capabilities and struggle with inference optimization. Industrial frameworks like Nemo-aligner (Shen et al., 2024a), ChatLearn (alibaba, 2017), and Verl (Sheng et al., 2024) offer advanced optimizations including 3D parallelism and memory management. However, they feature tightly coupled architectures requiring substantial engineering expertise and extensive infrastructure setup, creating accessibility barriers for academic researchers. These systems adopt static resource allocation paradigms, leading to suboptimal utilization and limited adaptability. Most frameworks inadequately address both inference bottlenecks and usability challenges simultaneously, forcing users to choose between high performance and ease of use.

3 Design of OpenRLHF

3.1 Overview: Ray-based RLHF architecture

OpenRLHF is the first open-source, Ray-based RLHF architecture that assigns a batch of GPUs to distinct roles and manages both data flow and workflow among these roles using Ray's scheduling capabilities. As illustrated in Figure 1, it defines two primary roles: the **rollout engine**, responsible for response generation to given prompts, and the **ZeRO engine**, which computes *logprobs*, *reference policy logprobs*, and handles model training (For a detailed design of the PPO workflow, refer to Appendix B, and for in-depth implementation tips, consult Blog (Shen et al., 2024b).)

Additionally, we leverage **VLLM** as the rollout engine, enabling efficient response generation with minimal GPU memory usage. For model training, we adopt DeepSpeed, which implements 3D parallelism, including automatic tensor parallelism, ZeRO/data parallelism, and sequence parallelism, to train both the actor and value models efficiently.

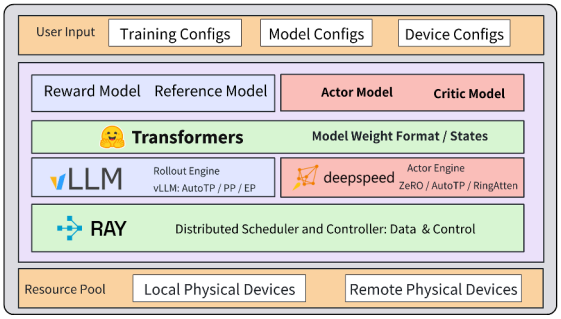


Figure 1: Overall architecture of OpenRLHF. The system assigns GPUs to two primary roles: rollout engines dedicated to response generation and actor engines responsible for computing log-probabilities and model training. OpenRLHF leverages Ray for distributed scheduling, integrates VLLM to achieve efficient response rollout with low GPU memory usage, and employs DeepSpeed-Zero for 3D parallelism (including tensor, data, and sequence parallelism) to enable efficient training. The underlying models are instantiated with flexible Transformer architectures, making the system easy to extend and adapt for diverse scenarios.

Core reason for ease of use: simplified model slicing, seamless integration, and flexible scheduling. The exchange of model weights between the rollout engine and the training engine is enabled by a flexible slicing and partitioning pipeline. **Hugging Face Transformer** (Jain, 2022) models are instantiated and trained using **DeepSpeed ZeRO**, **AutoTP**, and **Ring-Attention** (Liu et al., 2023) model parallelism. These model slices are then efficiently transferred to **VLLM** through **AutoTP** and **AutoPP**, which dynamically partition the models into sub-modules. The **Ray-based scheduling mechanism** enables seamless switching between different model parallelism modes, such as hybrid engine and asynchronous training. **This streamlined workflow significantly reduces complexity, making the system highly user-friendly and easy to extend.** Compared with architectures such as DeepSpeed-Chat, Transformer Reinforcement Learning (TRL), or other mainstream frameworks, OpenRLHF supports asynchronous dataflow and remote engine interactions, significantly improving the overall efficiency of the training process and the agent workflow.

3.2 Distributed and Efficient System Design

3D Parallelism with DeepSpeed-ZeRO and Ring Attention. To enable seamless and efficient scalability for large models, OpenRLHF integrates the latest automatic tensor parallelism (AutoTP) fea-

ture from DeepSpeed-ZeRO. In many industrial-grade RLHF architectures, users previously needed to manually specify an injection policy for each transformer model, identifying the linear layers and attention outputs that required communication between data-parallel ranks. In contrast, OpenRLHF leverages DeepSpeed-ZeRO's new capability to support automatic tensor parallelism for HuggingFace models by default: when kernel injection is not enabled and an injection policy is not provided, DeepSpeed automatically determines and applies the necessary policy at runtime. **This greatly simplifies the user experience and extends robust tensor parallelism support to a wider range of models, removing the need for complex engineering or manual configuration.**

In addition, OpenRLHF implements sequence parallelism through ring attention. Ring attention employs a ring-based communication topology, efficiently distributing attention computation for long sequences across multiple GPUs while minimizing both memory usage and communication overhead. This is especially critical for modern RLHF and RLVR workloads involving long chain-of-thought (CoT) reasoning, where conventional attention computation can become a major scalability bottleneck. By combining AutoTP, data parallelism, and ring attention-based sequence parallelism, OpenRLHF empowers large-scale, efficient, and highly usable RLHF model training on flexible GPU clusters.

Accelerated CoT Inference with vLLM. As LLMs advance in reasoning, RLHF and RLVR pipelines increasingly face inference bottlenecks, especially with long chain-of-thought (CoT) outputs. For models like OpenAI-o1 and DeepSeek-R1, **CoT generation can dominate training time, making efficient long-form inference crucial for scalability.** To address this challenge, OpenRLHF integrates the vLLM inference framework, which is specifically designed for high-throughput and memory-efficient LLM serving. vLLM provides a streamlined interface for generating RLHF samples and supporting frequent model weight updates.

The core innovation in **vLLM** is efficient management of attention key and value memory with **PagedAttention** (Kwon et al., 2023). This technique significantly reduces memory waste to less than 4%, enabling the batching of more sequences and enhancing GPU utilization and throughput. **PagedAttention** also supports efficient memory sharing for advanced sampling methods, such as parallel sampling and beam search, reducing memory usage by up to 55% and further boosting inference efficiency. Besides **PagedAttention**, **vLLM** has several other advantages, including continuous batching of incoming requests, fast model execution with CUDA Graph, and CUDA kernels optimized with FlashAttention and FlashInference, which enable rapid and memory-efficient attention computations. It also features speculative decoding for faster inference and chunked prefill to reduce latency on long sequences. Collectively, these enhancements make **vLLM** highly effective for large-scale, long-sequence inference, especially suitable for RLHF and RLVR pipelines, where efficiency and scalability are crucial.

Asynchronous Dataflow and Remote Engine Interactions. OpenRLHF supports asynchronous dataflow and remote engine communication to maximize system throughput and resource utilization during distributed RLHF training. In this architecture, rollout engines, actor engines, and remote engines operate independently and communicate via message passing, enabling immediate processing as soon as data becomes available. Fully asynchronous execution is especially important in the chain-of-thought (CoT) era, where inference involves generating multi-step reasoning that can vary greatly in length and computational cost. In synchronous frameworks, the slowest CoT generation can bottleneck the whole pipeline and waste resources. In contrast, OpenRLHF's asynchronous design allows each engine to operate at its own pace, ensuring hardware is utilized even for long or variable CoT tasks. This not only accelerates training and evaluation, but also enables efficient scaling and supports dynamic agent RL workflows. Leveraging asynchronous remote engine interactions, OpenRLHF is readily extensible for scalable **agent RL** training in modern, CoT-centric environments.

4 Experiments

4.1 Performance Comparison

Long CoT Experiment Setup To ensure the applicability of the compared methods in current RLHF workflows, we conduct our experimental evaluation under a long-chain-of-thought (CoT) generation scenario. Given computational resource constraints, we focus our comparison on the long CoT RLVR setting, benchmarking OpenRLHF against Verl, currently the state-of-the-art frame-

Model Size	1K OpenRLHF (sec)	1K Verl (sec)	2K OpenRLHF (sec)	2K Verl (sec)	4K OpenRLHF (sec)	4K Verl (sec)	8K OpenRLHF (sec)	8K Verl (sec)	Avg. Speedup
1.5B	14.9	16.2	26.5	33.1	61.6	65.5	113.0	134.8	1.22×
7B	16.0	17.3	30.3	47.3	90.3	101.3	226.4	232.4	1.42×
14B	25.5	28.5	51.0	74.3	136.3	202.8	328.6	511.1	1.68×

Table 1: The average training time (seconds) per step for different model sizes and context window lengths compared between OpenRLHF and Verl. The speedup is calculated as the geometric mean of Verl time / OpenRLHF time across all sequence lengths.

work for RLHF training. We evaluate the training efficiency of OpenRLHF (v0.8.5) and Verl (v0.4.0) by measuring the average per-step training time (in seconds) across various model sizes (1.5B, 7B, and 14B parameters) and maximum generation lengths (1K, 2K, 4K, and 8K tokens). To ensure the base models can produce sufficiently long contextual outputs for stress testing, we adopt the DeepSeek open-source distilled Qwen series. All models are fine-tuned using the Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) algorithm (Yu et al., 2025) under identical hyperparameter settings. Experiments are conducted on 8 NVIDIA H200 140GB GPUs using PyTorch 2.7 (Imambi et al., 2021) and the ZeRO Stage 3 optimizer or Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023). For each configuration, the local batch size is set to 1 to mitigate the risk of out-of-memory errors, with a maximum input context length of 1024 tokens. The reported values represent the average training time per step, excluding the first 10 steps.

Long CoT Performance Analysis The experimental results in Table 1 show that OpenRLHF consistently achieves superior training speed compared to Verl across all configurations. OpenRLHF delivers speedups ranging from 1.22× for the 1.5B model to 1.68× for the 14B model, with performance advantages becoming more pronounced as model size and context length increase. For instance, in the 14B-8K setting, OpenRLHF achieves a 1.56× speedup (328.6 seconds vs. 511.1 seconds), while the 7B-2K configuration shows a 1.56× speedup (30.3 seconds vs. 47.3 seconds). These speedup improvements can be attributed to OpenRLHF's algorithmic design, including the DAPO optimization strategy, which effectively mitigates memory overhead and computational bottlenecks under long-context scenarios. The consistent speedup across different scales underscores OpenRLHF's efficiency advantages in contemporary RLHF pipelines.

General RLVR Experiment To ensure a fair and controlled evaluation of training efficiency in reinforcement learning fine-tuning, we compare OpenRLHF with the optimized TRL framework on the GSM8K dataset (Cobbe et al., 2021) using the GRPO algorithm over a single training epoch. Both systems are configured with identical hyperparameters and run on the same hardware setup to isolate the effect of framework design on performance. The GRPO algorithm is selected due to its relevance in reward-based fine-tuning scenarios, and GSM8K serves as a representative benchmark for arithmetic reasoning tasks. In terms of training efficiency, OpenRLHF demonstrates a substantial advantage, completing one epoch in 1,657 seconds, compared to the 5,189 seconds required by TRL, representing approximately a 3.1× speedup. The result highlights the superior efficiency and maintainability of OpenRLHF's implementation, which benefits from a streamlined design and targeted system-level optimizations.

General RLHF Experiment To evaluate the training efficiency of modern RLHF frameworks, we compare OpenRLHF with the optimized DeepSpeed-Chat (DSChat) implementation. The experiment involves fine-tuning 1,024 prompts using the PPO algorithm for one epoch under identical hardware and hyperparameter configurations. This setup ensures that observed differences in performance are attributable solely to differences in system design and optimization strategies between the two frameworks. In terms of training time, OpenRLHF completes the task in 236.8 seconds, significantly outperforming DSChat, which requires 855.09 seconds, resulting in a 3.6× speedup. This performance gain is primarily driven by two system-level innovations in OpenRLHF: the use of vLLM for accelerated token generation and Ray for efficient distributed execution. Collectively, these design choices result in a more scalable and excellent RLHF framework.

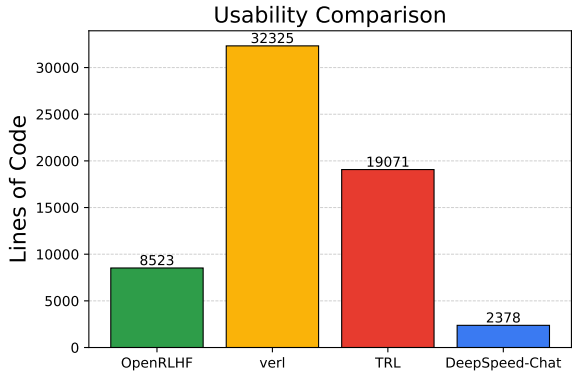


Figure 2: Core code complexity comparison across RLHF frameworks (lines of code). Lower values indicate simpler implementation and better maintainability.

4.2 Usability Comparison

As illustrated in Figure 2, OpenRLHF achieves a compelling balance between implementation conciseness and training performance. Despite being the second most concise framework with only 8,523 lines of code—significantly fewer than TRL (19,071) and Verl (32,325)—OpenRLHF demonstrates a clear performance advantage across standard RLHF benchmarks. This streamlined codebase not only facilitates easier comprehension and modification for developers but also reduces the engineering overhead associated with integration into custom pipelines. In addition to its lightweight design, OpenRLHF offers comprehensive support for various reinforcement learning fine-tuning paradigms, including Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO) (Rafailov et al., 2023), Reward Model (RM), and Process Reward Model (PRM) (Lightman et al., 2023). This broad functionality, combined with its modular and well-documented architecture, significantly lowers the barrier to entry for both research and production use. Overall, OpenRLHF's design exemplifies high usability by combining minimal code complexity with extensive functionality and competitive performance.

5 Limitations

While OpenRLHF demonstrates significant advantages in balancing performance and accessibility, several limitations should be acknowledged. Despite our optimization efforts, OpenRLHF may not match the peak performance of highly specialized industrial frameworks that benefit from dedicated engineering teams and extensive resources. As a community-driven open-source project without dedicated economic support, OpenRLHF faces resource constraints in rapidly integrating cutting-edge features, potentially resulting in delays compared to well-funded commercial frameworks. Currently, the framework focuses primarily on language models and does not support Vision-Language Models or other multimodal architectures, limiting its applicability to multimodal AI alignment research. Additionally, OpenRLHF's modular design introduces dependencies on external systems such as Ray, vLLM, and DeepSpeed, where updates in these upstream systems may require maintenance work or introduce compatibility issues. Despite these limitations, we believe OpenRLHF's contributions to accessibility and democratization of RLHF research provide significant value to the community.

6 Conclusion

We presented OpenRLHF, a simple yet high-performance open-source framework that bridges the gap between performance and usability in RLHF and RLVR training. By integrating Ray's distributed computing, vLLM's inference optimization, DeepSpeed ZeRO's memory efficiency, and ring attention's sequence parallelism, OpenRLHF delivers four key innovations: Ray-based architecture for streamlined orchestration, 3D parallelism for efficient scaling, accelerated CoT inference addressing the critical bottleneck, and asynchronous dataflow for maximum throughput. The framework's broad adoption across leading institutions and companies—from academic courses at CMU to production deployments at major tech companies—validates its real-world effectiveness. OpenRLHF's influence on subsequent frameworks and its role in democratizing RLHF research demonstrate its significant contribution to the field. By open-sourcing this framework, we aim to foster innovation, accelerate research progress, and enable broader participation in aligned AI development.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

alibaba. 2017. chatlearn. https://github.com/alibaba/ChatLearn.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, and 1 others. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. 2017. Openai baselines. https://github.com/openai/baselines.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. 2021. Pytorch. *Programming with TensorFlow: solution for edge computing applications*, pages 87–104.

Shashank Mohan Jain. 2022. Hugging face. In *Introduction to transformers for NLP: With the hugging face library and models to solve problems*, pages 51–67. Springer.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Shenggui Li, Hongxin Liu, Zhengda Bian, Jiarui Fang, Haichen Huang, Yuliang Liu, Boxiang Wang, and Yang You. 2023. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proceedings of the 52nd International Conference on Parallel Processing*, pages 766–775.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. Rllib: Abstractions for distributed reinforcement learning. In *International conference on machine learning*, pages 3053–3062. PMLR.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Hao Liu, Matei Zaharia, and Pieter Abbeel. 2023. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Gerald Shen, Zhilin Wang, Olivier Delalleau, Jiaqi Zeng, Yi Dong, Daniel Egert, Shengyang Sun, Jimmy Zhang, Sahil Jain, Ali Taghibakhshi, Markel Sanz Ausin, Ashwath Aithal, and Oleksii Kuchaiev. 2024a. Nemo-aligner: Scalable toolkit for efficient model alignment. *Preprint*, arXiv:2405.01481.

Wei Shen, Xiaonan He, Chuheng Zhang, Qiang Ni, Wanchun Dou, and Yan Wang. 2020. Auxiliary-task based deep reinforcement learning for participant selection problem in mobile crowdsourcing. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1355–1364.

Wei Shen, Jian Hu, Pengyu Zhao, Xiaonan He, and Lichang Chen. 2024b. Advanced tricks for training large language models with proximal policy optimization. https://swtheking.notion.site/eb7b2d1891f44b3a84e7396d19d39e6f?v=01bcb084210149488d730064cbabc99f&pvs=74. Notion Blog.

Wei Shen, Guanlin Liu, Zheng Wu, Ruofei Zhu, Qingping Yang, Chao Xin, Yu Yue, and Lin Yan. 2025. Exploring data scaling trends and effects in reinforcement learning from human feedback. *arXiv preprint arXiv:2503.22230*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

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.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

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, and 3 others. 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.

Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, and 1 others. 2023. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales. *arXiv preprint arXiv:2308.01320*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, and 1 others. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

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*.

A Broad Impact and Adoption

OpenRLHF has achieved significant adoption and impact across both academic and industrial communities since its release. The framework's balance of high performance and accessibility has made it a preferred choice for diverse applications ranging from research experimentation to production deployment.

OpenRLHF has been formally integrated into academic curricula, notably being adopted by The CMU Advanced Natural Language Processing Spring 2025 course as a core teaching framework. This integration demonstrates the framework's educational value and accessibility for students new to RLHF. The framework has also received recognition from major technology communities, including an invitation to present at the PyTorch Expert Exchange 2025, highlighting its technical contributions to the broader deep learning ecosystem.

The framework has been widely adopted by leading technology companies and research institutions, including Google, ByteDance, Baidu, NVIDIA, Tencent, China Telecom, Vivo, NexusFlow, JSC, UC Berkeley's Starling Team, Meituan, and HKUST. This diverse adoption across major tech companies, telecommunications providers, and academic institutions demonstrates OpenRLHF's versatility and robustness in handling various scales of RLHF workloads, from research prototyping to production-scale deployments.

OpenRLHF's modular design and extensible architecture have enabled it to serve as a foundation for specialized frameworks. Notable examples include LMM-R1 for multimodal reinforcement learning, MARTI for advanced reasoning tasks, and MM-EUREKA for multi-modal applications. These derivative frameworks demonstrate OpenRLHF's flexibility in supporting diverse research directions and its role as a platform for innovation in the RLHF ecosystem.

The design principles and technical innovations introduced in OpenRLHF have influenced subsequent framework development in the field. Several prominent frameworks including Verl, Alibaba ROLL, SLIME, and Open-Reasoner-Zero have acknowledged OpenRLHF's contributions in their documentation and publications, citing its distributed architecture design, Ray-based orchestration approach, and integration strategies as influential to their own development. This acknowledgment reflects OpenRLHF's role in advancing the state-of-the-art in RLHF system design and establishing best practices for distributed RLHF training.

Beyond direct usage and citations, OpenRLHF has contributed to democratizing RLHF research by lowering the barrier to entry for academic researchers and smaller organizations. The framework's emphasis on ease of use without sacrificing performance has enabled broader participation in RLHF research, fostering innovation across diverse research communities that might otherwise lack the resources for complex distributed training setups.

B PPO Workflow Design

This section presents OpenRLHF's comprehensive PPO-based RLHF training workflow, which orchestrates multiple specialized engines to efficiently handle the complex multi-stage training process (as shown in Figure 3). The OpenRLHF PPO workflow consists of four main stages executed iteratively: (1) **Rollout Generation**, where the current policy generates responses to prompts; (2) **Reward Computation**, where responses are evaluated using a trained reward model; (3) **Advantage Estimation**, where advantages and returns are calculated using GAE (Generalized Advantage Estimation); and (4) **Policy Optimization**, where the policy is updated using PPO loss. This pipeline repeats for multiple iterations until convergence.

The training begins with the **Rollout Engine** generating responses for a batch of prompts using the current policy π_θ. A batch of prompts $\{x_1, x_2, ..., x_B\}$ is sampled from the training dataset, and the Rollout Engine, equipped with vLLM for efficient inference, generates responses $\{y_1, y_2, ..., y_B\}$ using the current policy. During generation, the engine records action log-probabilities $\log \pi_\theta(y_i|x_i)$ and attention masks. The generated sequences (x_i, y_i) along with their metadata are collected for subsequent processing. The Rollout Engine leverages vLLM's optimizations including continuous batching, KV-cache management, and PagedAttention to maximize throughput during this inference-heavy stage.

Once rollouts are generated, the system computes rewards and reference policy log-probabilities. The trained reward model R_ϕ evaluates each prompt-response pair to produce scalar rewards $r_i = R_\phi(x_i, y_i)$. Simultaneously, the frozen reference policy π_{ref} computes log-probabilities $\log \pi_{\text{ref}}(y_i|x_i)$ for KL regularization, and the critic

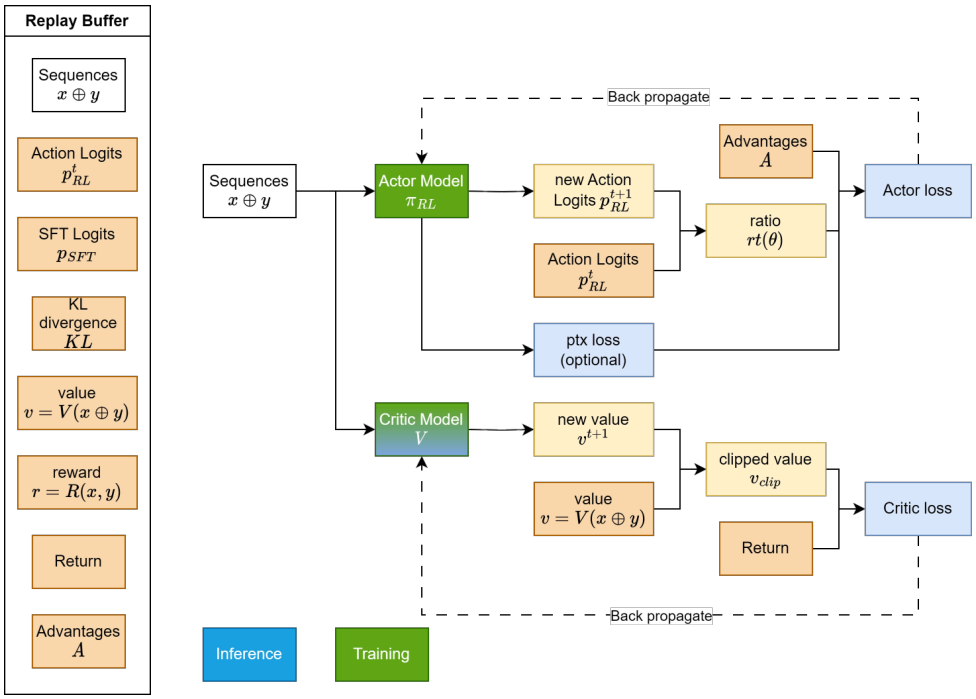


Figure 3: Overall PPO workflow of OpenRLHF.

network V_ψ estimates state values $V_\psi(x_i, y_{i,:t})$ for advantage computation. These computations can be parallelized across multiple GPUs and are often batched together for efficiency.

The system then computes advantages using Generalized Advantage Estimation (GAE). First, temporal difference residuals are calculated: $\delta_t = r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t)$. Then, advantages are computed using GAE: $A_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}$, and discounted returns are calculated as $R_t = A_t + V_\psi(s_t)$. The advantage computation incorporates KL penalty terms to prevent the policy from deviating too far from the reference policy: $r'_t = r_t - \beta\text{KL}[\pi_\theta||\pi_{\text{ref}}]$.

The final stage updates the policy using PPO's clipped objective function. The **ZeRO Engine** computes the policy loss using the clipped surrogate objective: $L^{\text{CLIP}}(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$, where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio. The critic loss is computed as $L^V(\psi) = \mathbb{E}[(V_\psi(s_t) - R_t)^2]$. The critic loss is computed as $L^V(\psi) = \mathbb{E}[(V_\psi(s_t) - R_t)^2]$. The total loss combines policy and value losses: $L_{\text{total}} = L^{\text{CLIP}} + c_1L^V + c_2S[\pi_\theta](s_t)$, where c_1 and c_2 are coefficients for value loss and entropy bonus respectively, and $S[\pi_\theta](s_t)$ is the entropy of the policy distribution to encourage exploration. The ZeRO Engine performs gradient computation and model parameter updates using DeepSpeed ZeRO optimizations for memory efficiency and scalability.

Throughout this workflow, OpenRLHF's Ray-based architecture enables seamless coordination between different engines while maintaining computational efficiency. The Rollout Engine and ZeRO Engine can operate on different GPU clusters optimized for their respective workloads—inference-optimized hardware for rollout generation and training-optimized hardware for policy updates. Ray's distributed scheduling automatically handles data transfer, synchronization, and fault tolerance across the distributed system. This asynchronous execution model allows the system to overlap computation stages where possible, significantly improving overall training throughput compared to traditional synchronous RLHF implementations.

The entire PPO training loop continues for multiple epochs, with each epoch processing multiple batches of rollout data. Early stopping criteria based on KL divergence thresholds and performance metrics prevent policy collapse and ensure stable training. The modular design allows for easy integration of advanced techniques such as advantage normalization, gradient clipping, and adaptive KL penalty coefficients, making OpenRLHF highly customizable for different research and production scenarios.

C Full Contributors

- **Ray Integration:** Jian Hu, Xibin Wu, Songlin Jiang
- **vLLM Integration:** Jian Hu, Xibin Wu, Songlin Jiang, Kaichao You (vLLM Team)
- **Ring Attention:** Zilin Zhu (Zhipu), Zhibo Zhou (Vivo), gzpan(GitHub User), Jian Hu
- **Supervised Learning:** Haoran Wang and Xianyu
- **DeepSpeed and Transformers Integration:** Jian Hu, Xibin Wu, Songlin Jiang, Bin Chen, Yiming Liu
- **Asynchronous Agentic RL:** Haotian Xu and Jian Hu
- **Single Controller Architecture:** Jian Hu
- **PPO Implementation:** Jian Hu
- **GRPO Implementation:** Jason Klein Liu
- **KL Control Mechanism:** Yiming Liu and Jason Klein Liu
- **RLVR Experiment Design:** Jason Klein Liu
- **RLHF Experiment Design:** Jian Hu
- **Dynamic Sampling:** Jian Hu
- **Dynamic Batching:** Hao Chen
- **Documentation:** Jian Hu
- **Testing and Bug Reports:** Weikai Fang, Wei Shen, Weixun Wang
- **Paper Writing and Presentations:** Wei Shen, Jason Klein Liu, Weixun Wang, Yu Cao, Jian Hu

A more complete list can be found in the OpenRLHF commit and release history.

ConfReady: A RAG based Assistant and Dataset for Conference Checklist Responses

Michael Galarnyk[1*✉] **Rutwik Routu**[2*†] **Vidhyakshaya Kannan**[3*†] **Kosha Bheda**[1]
Prasun Banerjee[1] **Agam Shah**[1] **Sudheer Chava**[1]

[1] Georgia Institute of Technology [2] Duke University [3] Sai University
✉ Corresponding author: mgalarnyk3@gatech.edu

Abstract

The ARR Responsible NLP Research checklist website states that the "checklist is designed to encourage best practices for responsible research, addressing issues of research ethics, societal impact and reproducibility." Answering the questions is an opportunity for authors to reflect on their work and make sure any shared scientific assets follow best practices. Ideally, considering a checklist before submission can favorably impact the writing of a research paper. However, previous research has shown that self-reported checklist responses don't always accurately represent papers. In this work, we introduce ConfReady, a retrieval-augmented generation (RAG) application that can be used to empower authors to reflect on their work and assist authors with conference checklists. To evaluate checklist assistants, we curate a dataset of 1975 ACL checklist responses, analyze problems in human answers, and benchmark RAG and Large Language Model (LM) based systems on an evaluation subset. Our code is released under the AGPL-3.0 license on GitHub, with documentation covering the user interface and Python package.

1 Introduction

In order to submit a paper to conferences under the Association for Computational Linguistics like ACL, COLING, CoNLL, EMNLP, and NAACL, authors are required to submit their answers to the ARR Responsible NLP Research checklist[1]. The checklist was mostly developed through a combination of the NLP Reproducibility Checklist (Dodge et al., 2019), the reproducible data checklist (Rogers et al., 2021), and the NeurIPS 2021 Paper Checklist Guidelines[2]. The goal of this process is to address reproducibility, societal impact, and potential ethical issues. Authors are expected to discuss limitations, artifact usage, computational details, human involvement, and use of AI assistants. Starting with EMNLP 2025, checklist responses will be published as appendices alongside accepted papers[3], in order to "help with transparency" and encourage authors to "think more carefully about these issues when they know their answers will be visible to the broader community."

This follows an earlier pilot at ACL 2023, where checklist responses—covering 19 questions per paper—were appended to accepted submissions. For example, question A2 asks: *"Did you discuss any potential risks of your work?"* If authors respond "yes," they must cite the relevant section; if "no," they are expected to provide a justification. However, prior work has noted cases of low-effort or bad-faith responses, such as identical answers across questions and falsely reporting code availability (Magnusson et al., 2023). The ACL 2023 program chairs suggested that checklist sloppiness correlates with sloppiness elsewhere in the work and a lower acceptance rate (Rogers et al., 2023).

To mitigate unreliable checklist answers, conferences have explored Large Language Model (LM) based systems to assist authors (Goldberg et al., 2024), though these were not evaluated against human-written checklist submissions and do not reflect the full complexity of real submissions. To address this, LMs can be augmented with tools like retrieval-augmented generation (RAG), which integrates information retrieval with generative models (Lewis et al., 2020). This approach improves accuracy and relevance, especially for question-answering tasks requiring up-to-date or domain-

* These authors contributed equally.
† Work done as a Volunteer Research Assistant at Georgia Institute of Technology.
[1] https://aclrollingreview.org/responsibleNLPresearch/
[2] https://neurips.cc/Conferences/2021/PaperInformation/PaperChecklist
[3] https://aclrollingreview.org/responsible-nlp-checklist-appendices

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 667–677
November 4-9, 2025

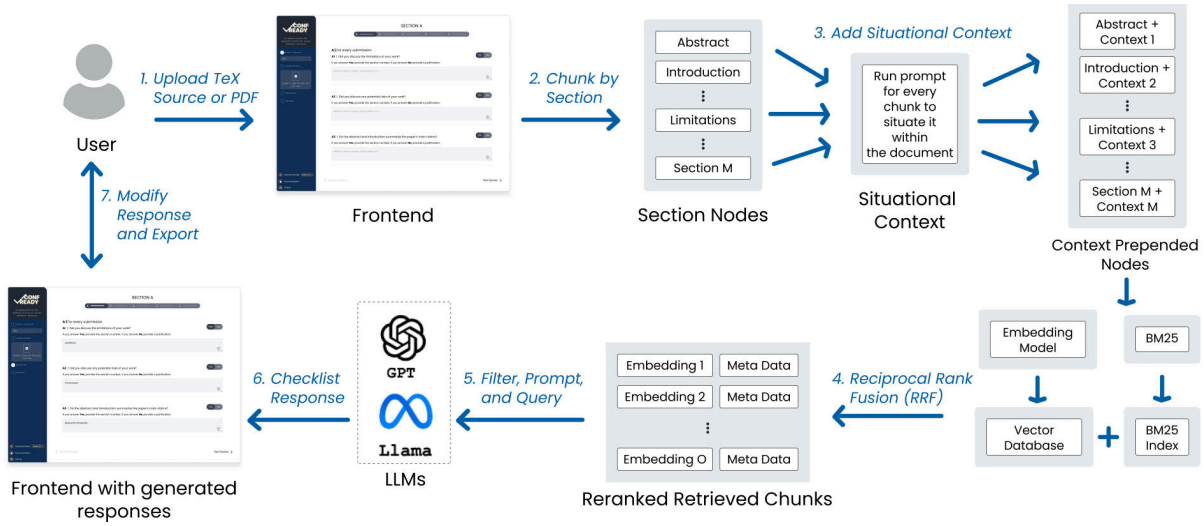


Figure 1: Users can upload either TeX source (single .tex file or a zipped folder) or a PDF to the frontend to receive an LM-generated checklist response, which can then be modified and exported.

specific knowledge (Karpukhin et al., 2020).

In this work, we introduce **ConfReady**, a RAG tool that helps with checklist responses grounded in their paper's TeX source or PDF[4]. To evaluate ConfReady, we compile the ConfReady dataset of real checklist submissions (see Section 3) and evaluate its outputs against human-written answers (see Section 4).

Our contributions are the following:

- **ConfReady Tool**: An end-to-end system for generating checklist responses from TeX source (single .tex file or zipped folder) or PDF, with a user-friendly interface and pip-installable backend.
- **Checklist Dataset**: A structured dataset of 1975 ACL papers with parsed checklist responses and metadata, enabling analysis and benchmarking.
- **Backend Evaluation**: A comparison of RAG and LM only backends on 93 ACL papers, with accuracy measured against human-provided checklist responses.

2 ConfReady

Figure 1 presents the ConfReady pipeline. Users start by uploading either TeX source (single .tex or zipped folder) or a PDF on the frontend. The system then processes the document through a seven-step workflow: (1) upload input, (2) chunk by section, (3) add situational context to each section, (4) perform Reciprocal Rank Fusion (RRF), (5) construct and send prompts to the LM, (6) generate checklist responses, and (7) allow users to review, edit, and export the results. The user-facing interface, shown in Figure 2, supports this workflow with features like section-level navigation, editable response fields, and progress tracking. Appendix A details the design rationale behind each interface component.

2.1 Parsing, Chunking, and Embedding

Parsing Users can upload either TeX Source (single .tex file or zipped folder) or PDF. When TeX Source is provided, the document is parsed to remove comments and pre-abstract content. For PDFs, a simplified pipeline is applied.

Contextual Chunking ConfReady uses contextual chunks (Anthropic, 2024) to reduce retrieval failure rates. Each chunk is annotated with metadata (e.g., section title, neighboring chunk identifiers), and a short LM-generated situational summary is prepended to the chunk before embedding using the following prompt:

> **Prompt:** Provide a concise (50–100 tokens) situational summary for this chunk, capturing its role in the larger section.

Embeddings The enriched chunks (metadata, situational summary, original text) are then converted

[4]A video demonstration is available at https://youtu.be/sNhpKJLfArc?si=0CMCe1nEFwFFUibw, with documentation and a pip-installable package at https://confready-docs.vercel.app.

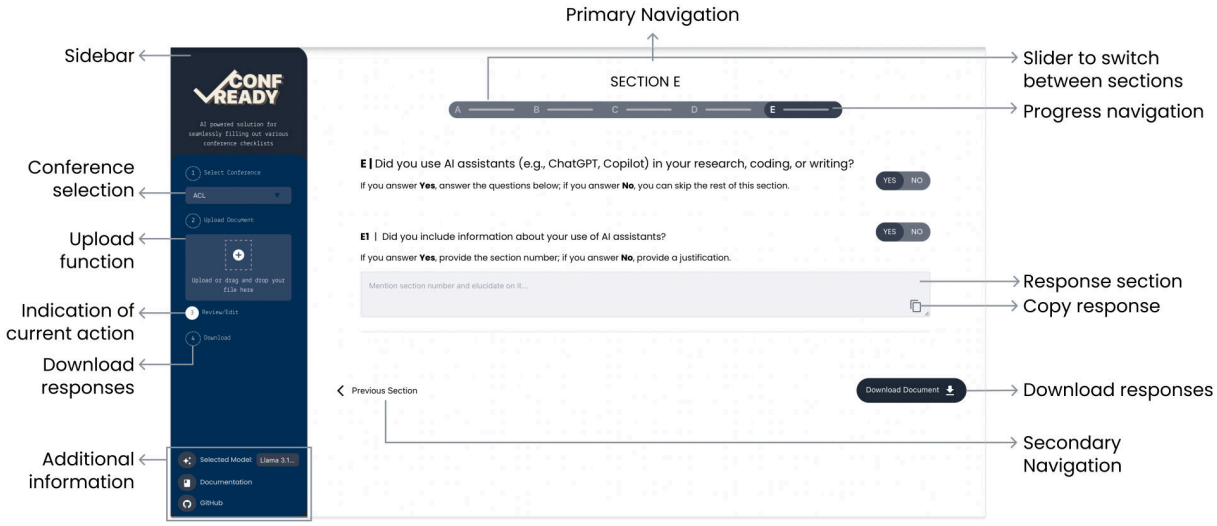


Figure 2: Features of the ConfReady user interface include: an upload function within the sidebar, primary navigation with a slider to switch between sections and progress navigation, and a generated response field with a copy function. The design rationale behind the features is listed in Appendix A.

into vector representations and stored locally. The embedding backend is modular and can be configured to support different models.

2.2 Retrieval, Fusion, and Reranking

Retrieval Our retrieval workflow is designed to combine the strengths of dense vector embeddings and sparse lexical search.

Dense Retrieval with Embeddings Each chunk is embedded using OpenAI's `text-embedding-3-large`. Queries are also embedded and compared against the document embeddings using cosine similarity.

Sparse Retrieval with BM25 In parallel, we build a lexical index over all contextualized chunks using the BM25 algorithm. This sparse retrieval step is especially effective at capturing exact matches, uncommon terminology, and keyword-based relevance that dense models may miss.

Score Fusion (Vector + BM25) To balance semantic and lexical relevance, we perform RRF on the top results from both the dense and sparse retrieval components.

LM-Based Reranking The fused results are passed through a reranking module, which uses an LM to evaluate the relevance of each chunk in the context of the original query. The top-k chunks from this reranking step are selected as final inputs for generation.

CRAG vs. NRAG We refer to the full workflow described above—retrieval, fusion, and reranking with contextual chunking—as CRAG, short for Contextual Retrieval-Augmented Generation (Anthropic, 2024). For comparison, we also define a baseline, NRAG (Naive Retrieval-Augmented Generation), which uses basic document chunks in place of contextual ones.

2.3 Prompt Design

ConfReady uses modular instructions that can be adapted to different conference checklists and checklist versions. The ACL 2023 checklist, for instance, contained 19 questions labeled A1–D5. Each question is mapped to a dedicated prompt with a uniform structure: Introduction, Question, Additional Context, and Output Structure. Figure 3 shows the prompt for A1 (*"Did you discuss the limitations of your work?"*).

The prompt is designed to provide the LM with the same information humans should consider when answering the question. The "Question" corresponds to an individual question in the checklist. The "Additional Context" is information provided from Guidelines for Answering Checklist Questions on the ACL Rolling Review website[6].

The *Output Structure* specifies that the response should be a JSON object with `answer`, `section name`, and `justification` as keys. Restricting output to JSON has been shown to improve clas-

[6] https://aclrollingreview.org/responsibleNLPresearch/

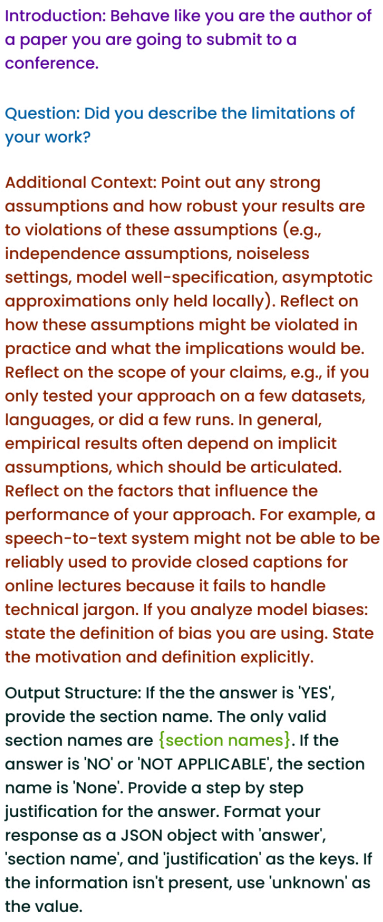


Figure 3: Example prompt for question A1. *Introduction*: instructs the LM to assume the role of an author. *Question*: the checklist item the LM must address. *Additional Context*: provides supporting guidance drawn from checklist documentation. *Output Structure*: instructs the LM to return JSON with fields for `answer`, `section name`, and `justification`. *Section Names*: lists valid section names parsed directly from the parsed files.

sification accuracy and reproducibility (Tam et al., 2024; Es et al., 2024). Section names are taken directly from the parsed files, so the LM is limited to choosing among sections that appear in the paper.

2.4 Frontend with Generated Responses

After inference, the LM output is passed to the frontend in a structured JSON format with three fields: `answer`, `section name`, and `justification`. When the answer is `"Yes"`, the corresponding section name is displayed in the interface (Figure 2). When the answer is `"No"`, the section name defaults to `"unknown"`, and the justification is shown instead.

Backend Hallucination Mitigation RAG applications are known to reduce hallucination inherent in LMs (Shuster et al., 2021). To further mitigate the two hallucination categories, factuality and faithfulness, outlined by Huang et al. (2023), the application implements several safeguards. First, to address instruction inconsistency (a type of faithfulness hallucination) where the LM deviates from the instruction to return a single section, the system withholds a response. Second, because section names are parsed directly from the paper, only those names are allowed as valid answers.

User Checklist Modification LM-generated answers are intended to assist authors rather than replace their judgment. Users must review each response for accuracy, and questions concerning the use of AI assistants in research, coding, or writing must be answered manually. As a final safeguard, authors are required to validate responses before export. The validated checklist can then be exported as a Markdown document. Markdown was chosen because it is lightweight, easy to convert into other formats (e.g., PDF and TeX), and widely adopted in open-source ecosystems such as GitHub READMEs and Hugging Face model cards (Yang et al., 2024).

2.5 System Architecture

User Interface The frontend is built with React[7], a JavaScript library for creating modular and interactive web applications. For consistent and responsive styling across devices, we use TailwindCSS[8], a utility-first CSS framework that accelerates UI development with predefined class utilities.

API Orchestration and Backend Workflow The backend is implemented using Flask[9], a lightweight Python web framework that manages communication between the frontend and backend. It handles file uploads, runs processing scripts (e.g., TeX parsing), and orchestrates interactions with the RAG pipeline. RAG is implemented by LlamaIndex (Liu, 2022), which integrates external context into the LM inference. To maintain real-time communication between the backend and frontend, a server-side event endpoint on the Flask server streams updates to the client during critical stages of the file processing workflow.

[7] https://reactjs.org
[8] https://tailwindcss.com/
[9] https://flask.palletsprojects.com/en/3.0.x/

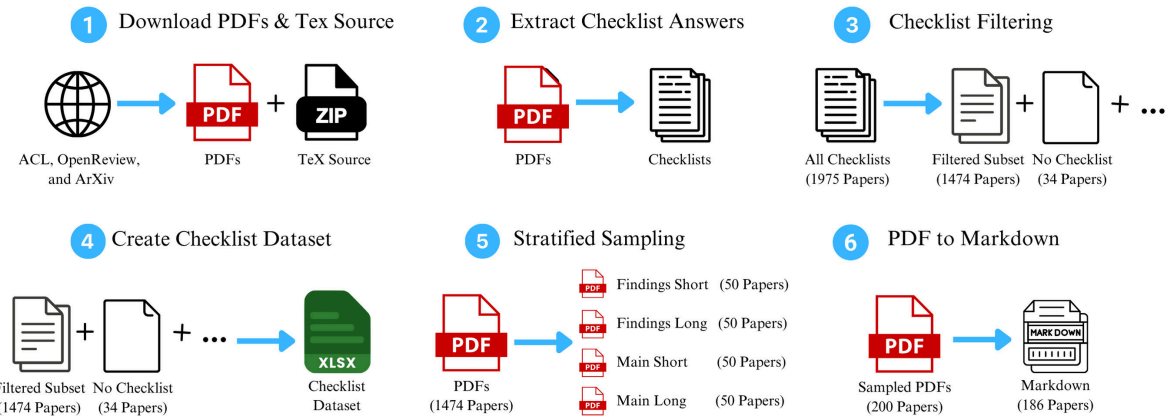


Figure 4: Checklist dataset generation pipeline with six stages: (1) download PDFs and TeX sources for each paper, (2) extract and verify checklist responses, (3) analyze and filter out problematic checklists (e.g., blank or missing responses), (4) organize the remaining checklists and metadata into a structured Excel file, (5) apply stratified sampling across categories, and (6) convert the sampled PDFs into Markdown using the `marker` library[5] for evaluation.

3 ConfReady Dataset

To evaluate ConfReady and understand how authors engage with checklist questions in practice, we curate a dataset of 1975 ACL 2023 papers, the most recent major Association for Computational Linguistics venue to publish checklists alongside accepted submissions. While prior work has documented issues with checklist reliability—such as identical answers across questions or unsupported claims of code availability (Magnusson et al., 2023)—no prior work has open-sourced a large dataset of real, author-written checklist responses. Our dataset addresses this gap, supporting both large-scale analysis of checklist quality and benchmarking of RAG and LM systems.

Figure 4 illustrates the six-stage pipeline used to construct the dataset: (1) download PDFs and arXiv TeX sources for each paper, (2) extract and verify checklist responses, (3) analyze checklists and create a filtered subset without issues (e.g., blank, missing, etc.), (4) organize the remaining checklists and metadata into a structured Excel file, (5) apply stratified sampling to select 50 papers from each category (ACL Findings Short, Findings Long, Main Short, Main Long), and (6) convert the sampled PDFs into Markdown using the `marker` library[10], leaving 186 papers after excluding a small number with conversion failures. Due to this, ConfReady prefers TeX source input when available.

Dataset Statistics To assess the quality and consistency of checklist submissions, we conducted a detailed analysis of all 1975 papers. Table 1 summarizes key statistics, revealing several notable inconsistencies. Some papers omitted checklists entirely (*No Checklist*), while others appended blank templates (*Blank Checklist*) or didn't reference a single section in their responses (*No Section Names*). Notably we also had relatively few disclosures about AI use in writing (*AI Use in Writing*). Finally, to support evaluations of RAG and standalone LM backends on TeX, we also filtered out papers without corresponding arXiv TeX sources.

	Findings		Main	
Metric	Short	Long	Short	Long
All Collected Papers				
Total Papers	189	712	164	910
No Checklist	6	11	3	14
Blank Checklist	8	36	10	60
All Yes Responses	5	7	2	9
No Section Names	7	15	3	28
AI Use in Writing	13	39	11	60
Not on arXiv	50	190	37	190
Evaluation Sample				
Total Papers	46	43	47	50
Avg Tokens (Paper)	12563	19865	14049	24547

Table 1: Summary statistics grouped by track (Findings/Main) and paper length (Short/Long).

Token Length Analysis Figure 5 presents token count distributions across the four evaluation subsets. On average, ACL Main papers are longer than ACL Findings papers in both short and long categories.

[10] https://github.com/datalab-to/marker

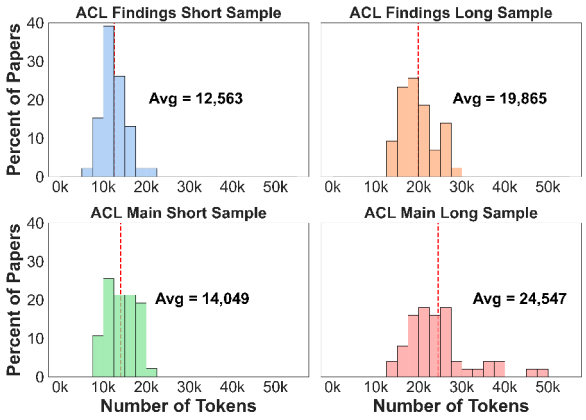


Figure 5: Token count distributions for ACL 2023 papers across four categories in our evaluation subset: ACL Findings Short, ACL Findings Long, ACL Main Short, and ACL Main Long. The y-axis indicates the percentage of papers; the x-axis shows token counts (measured using the Llama-3 tokenizer). Red dashed lines indicate mean token counts.

Final Structure The ConfReady dataset preserves parsed checklist question–answer pairs, justification text, referenced sections, and metadata flags indicating issues such as incomplete or blank submissions. Further details on the data collection and analysis process are provided in Appendix C.

4 Evaluation

	Findings	Main
Model	Long	Long
RAG Framework on TeX		
CRAG (Llama-3.1-405B)	81.72	81.93
CRAG (Llama-3.3-70B)	78.07	78.78
CRAG (GPT-4o)	80.58	79.86
NRAG (Llama-3.1-405B)	78.44	77.36
NRAG (Llama-3.3-70B)	74.64	73.65
NRAG (GPT-4o)	80.43	73.22
LM on TeX		
Llama-3.1-405B	78.87	75.83
Llama-3.3-70B	78.69	79.20
GPT-4o	80.54	78.45
LM on MD		
Llama-3.1-405B	79.86	77.26
Llama-3.3-70B	80.97	81.09
GPT-4o	82.27	77.38

Table 2: Accuracy comparison of RAG, LMs on TeX, and LMs on PDFs for ACL Main (Long) and ACL Findings (Long) papers.

We evaluate Llama-3.1–405B, Llama-3.3–70B (Meta et al., 2024), and GPT-4o (OpenAI, 2023a) on the evaluation sample. Due to compute limits, experiments focus on long-form ACL submissions.

Models are tested in three settings: (i) RAG on TeX with CRAG and NRAG, (ii) LM on parsed TeX, and (iii) LM on MD. Human-annotated answers serve as references, allowing us to evaluate how effectively models can reflect on ethical considerations, reproducibility, and societal impacts in each setup.

Results CRAG consistently outperforms NRAG, confirming the benefit of section-aware retrieval, RRF, and LM reranking (see Table 2). LM on Markdown performs competitively with LM on TeX, and in some cases better, echoing previous work showing that structured Markdown can improve model fidelity (Min et al., 2024; Jain et al., 2025; Galarnyk et al., 2025). Nevertheless, ConfReady favors TeX input with RAG, since TeX parsing better preserves section structure and reduces extraction errors, whereas PDF-to-Markdown conversion is prone to failures that can disrupt retrieval and alignment. Finally, GPT-4o and other models often perform better on Findings than on Main papers, suggesting that longer Main submissions (see Figure 5) introduce added difficulty for checklist answering.

Error Analysis Figure 6 shows accuracy by checklist question for ACL Findings Long. Question C1—*"Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?"*- had the lowest LM accuracy. Appendix B presents typical failure cases on C1. A closer review of these disagreements revealed a recurring pattern: LMs often answered "NO" unless all three components were explicitly mentioned, while human justifications typically cited only section references (e.g., "Section 4" or "Appendix C1") without clarifying what details were provided. This highlights a stricter model standard for completeness versus more lenient human interpretations.

5 Conclusion and Future Work

This paper introduces ConfReady, a LM-based system which can be used to empower authors to reflect on their work and act as an assistant to help authors with conference checklists. With ConfReady, authors can get a LM checklist response that they use to reflect on their work or modify it

before submitting. We hope that the open-source application will be responsibly used as an assistant and tool for reflection. As a future work, we are still working on improving the project across several different directions:

- **Local LMs**: While ConfReady currently uses commercial providers to avoid the overhead of self-hosting, future versions will support local open-weight models to enable private, offline usage in settings where data sensitivity or API constraints are a concern.
- **Other Conference Checklists**: ConfReady currently supports checklists from conferences under the Association for Computational Linguistics (e.g., ACL, COLING, CoNLL, EMNLP, and NAACL). While NeurIPS support has been implemented, adapting ConfReady to other venues will require adjustments for different checklist structures and question formats.

Ethics Statement

Structured Output Format A major issue with incorporating LMs into applications is their failure to follow output format inconsistency (faithfulness hallucination). We mitigate this by requiring responses in JSON format, similar to the JSON mode in the OpenAI and Gemini APIs (Gemini Team et al., 2024). Additionally, some libraries such as Instructor[11] also require JSON.

User Reliance and Scope Analysis of ChatGPT usage shows that non-work messages now comprise over 70% of interactions, up from 53% in 2024 (Chatterji et al., 2025). This reflects a broadening role for LMs in everyday life, including learning and creative expression, beyond their original productivity-focused scope. At the same time, models can provide fluent but incomplete or outdated answers when knowledge falls outside their training data (Shah et al., 2025). These patterns highlight the importance of using ConfReady as an aid for reflection and editing, not as a replacement for author responsibility.

References

Saleh Afroogh, Ali Akbari, Emmie Malone, Mohammadali Kargar, and Hananeh Alambeigi. 2024. Trust in ai: progress, challenges, and future directions. *Humanities and Social Sciences Communications*, 11(1):1–30.

Anthropic. 2024. Introducing contextual retrieval. Accessed: 2025-07-01.

Aaron Chatterji, Tom Cunningham, David Deming, Zoë Hitzig, Christopher Ong, Carl Shan, and Kevin Wadman. 2025. How people use chatgpt. Technical report, OpenAI & Harvard University.

Frederick G. Conrad, Mick P. Couper, Roger Tourangeau, and Andy Peytchev. 2010. The impact of progress indicators on task completion. *Interacting with Computers*, 22(5):417–427.

Meredith Davis and Jamer Hunt. 2017. *Visual communication design: An introduction to design concepts in everyday experience*. Bloomsbury Publishing.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. 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 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Michael Galarnyk, Agam Shah, Dipanwita Guhathakurta, Poojitha Nandigam, and Sudheer Chava. 2025. How inclusively do LMs perceive social and moral norms? In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4859–4869, Albuquerque, New Mexico. Association for Computational Linguistics.

Gemini Team et al. 2024. Gemini: A family of highly capable multimodal models.

Alexander Goldberg, Ihsan Ullah, Thanh Gia Hieu Khuong, Benedictus Kent Rachmat, Zhen Xu, Isabelle Guyon, and Nihar B. Shah. 2024. Usefulness of llms as an author checklist assistant for scientific papers: Neurips'24 experiment.

Mariam Guizani. 2022. A decade of information architecture in hci: A systematic literature review. *arXiv preprint arXiv:2202.13412*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions.

[11] https://github.com/instructor-ai/instructor

Arihant Jain, Purav Aggarwal, and Anoop Saladi. 2025. AutoChunker: Structured text chunking and its evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 983–995, Vienna, Austria. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Jerry Liu. 2022. LlamaIndex.

Ian Magnusson, Noah A. Smith, and Jesse Dodge. 2023. Reproducibility in NLP: What have we learned from the checklist? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12789–12811, Toronto, Canada. Association for Computational Linguistics.

Meta et al. 2024. The llama 3 herd of models.

Dehai Min, Nan Hu, Rihui Jin, Nuo Lin, Jiaoyan Chen, Yongrui Chen, Yu Li, Guilin Qi, Yun Li, Nijun Li, and Qianren Wang. 2024. Exploring the impact of table-to-text methods on augmenting LLM-based question answering with domain hybrid data. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 464–482, Mexico City, Mexico. Association for Computational Linguistics.

Pawarat Nontasil and Chatpong Tangmanee. 2024. Investigating the impact of progress indicator design on user perception of delay. *Journal of System and Management Sciences*, 14:333–344.

OpenAI. 2023a. Gpt-4 technical report. Technical report, OpenAI. Available at https://doi.org/10.48550/arXiv.2303.08774.

Anna Rogers, Timothy Baldwin, and Kobi Leins. 2021. 'just what do you think you're doing, dave?' a checklist for responsible data use in NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4821–4833, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anna Rogers, Marzena Karpinska, Jordan Boyd-Graber, and Naoaki Okazaki. 2023. Program chairs' report on peer review at acl 2023. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages xl–lxxv, Toronto, Canada. Association for Computational Linguistics.

Agam Shah, Liqin Ye, Sebastian Jaskowski, Wei Xu, and Sudheer Chava. 2025. Beyond the reported cutoff: Where large language models fall short on financial knowledge.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models.

Xiao Yang, Wei Liang, and Jie Zou. 2024. Navigating dataset documentations in ai: A large-scale analysis of dataset cards on huggingface. In *Proceedings of The Twelfth International Conference on Learning Representations*. ICLR.

A Features of the User-Interface

The features incorporated into the user interface, along with the underlying rationale for their design, are detailed below. These design decisions aim to enhance usability, ensure accessibility, and support an intuitive and efficient user experience throughout the checklist completion workflow.

1. *Side Bar/Upload:* The side bar incorporates the visual identity of the platform. It has been visualized to resemble file tabs to help users connect with the overarching action being performed using visual connotation (Davis and Hunt, 2017). It contains the upload function which allows users to upload their paper's TeX source (single .tex file or zipped folder) and visual indication of the user's current action. The bottom of the sidebar contains a model selector and links to the documentation[12] and GitHub[13] placed according to information hierarchy principles (Guizani, 2022).

2. *Conference Selection:* Users select a conference checklist. Currently, the platform allows users to select from ACL checklists, NeurIPS, and NeurIPS Datasets and Benchmarks (NeurIPS D&B).

[12] https://confready-docs.vercel.app/docs/walkthrough

[13] https://github.com/gtfintechlab/ConfReady

3. *Primary navigation:* The top bar of the interface provides the users with functionality of switching between sections. It also indicates the progress for each section, keeping the users informed through visually represented data (Nontasil and Tangmanee, 2024).

4. *Secondary navigation:* To refrain from disrupting the user's workflow while performing important tasks like checking or editing responses, the secondary navigation allows movement to the next page without needing to return to the primary navigation. The intention with this navigation is reducing extraneous cognitive overload.

5. *Response sections:* Responses are filled in and users need to verify responses.

6. *Download:* Only after users have reviewed each section are they allowed to download all of their responses from either the sidebar or from the Download button in the final section. The intention is to encourage users to be responsible for verification of AI-driven results (Afroogh et al., 2024).

The ConfReady user journey is shown in the ConfReady documentation[14]. To use the application, users upload the TeX source (single .tex file or zipped folder) or PDF. Next, in order to enhance task completion (Conrad et al., 2010), a progress screen appears to let users know of the backend RAG progress. This feature was added to the platform after informal interviews where it was noted that users wanted to get some indication on how long they needed to wait before they can check/edit responses, and download results.

B Qualitative Analysis of Checklist Responses

We provide qualitative insights into system behavior on ACL 2023 checklists. Details of dataset construction and statistics are provided in Appendix C.

B.1 Discrepancies on Question C1

Question C1 of the ACL checklist asks: *"Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?"* To better understand the low accuracy on this question, we reviewed a sample of cases where the LM response diverged from the human-provided checklist. Table 3 shows representative examples. In all cases, human annotators answered "YES" while the LM answered "NO." This reflects a recurring failure mode: the LM applied a stricter standard, requiring explicit mentions of model parameters, compute budget, and infrastructure to justify a "YES," while human authors often responded affirmatively based on partial evidence.

Notably, compute resource reporting was almost always absent. The LM typically acknowledged some modeling information but still answered "NO" if all three elements were not present. This conservative behavior highlights a mismatch between human and model expectations of completeness.

B.2 ACL Edge Cases

While the ConfReady system accurately processed most ACL submissions, several edge cases highlighted current limitations. One issue involved exceeding the model's context length limit, leading to inference failures. Additionally, some model-generated responses were phrased awkwardly or contained placeholder-like text (e.g., "x and y should be..."), suggesting room for prompt refinement or output filtering.

C Checklist Dataset Construction

We describe the pipeline used to construct the structured checklist dataset.

C.1 Data Collection and Extraction

Due to the scale of the dataset, most of the collection was automated using Python scripts. The multi-stage pipeline proceeded as follows:

1. **Link Retrieval** Automated scripts using `requests` and `BeautifulSoup` were used to identify and retrieve arXiv links as well as ACL Anthology PDF links for each paper.

2. **PDF Retrieval and Preprocessing** Normalized PDF links were obtained from ACL Anthology URLs. To extract checklists, only the last two pages of each ACL paper were parsed using `PyPDF2`, since checklists consistently appeared there.

3. **Checklist Parsing** Each checklist was parsed using conference-specific regex templates.

[14] https://confready-docs.vercel.app/docs/walkthrough

Human Response	LM Response	Human Justification	LM Justification
Yes	No	Section A.1	The text does not discuss the experimental setup, including hyperparameter search and best-found hyperparameter values. While it mentions experiments and evaluations of different models, it does not provide details on how hyperparameters were tuned or selected.
Yes	No	Appendix D	The paper does not report the number of parameters in the models used, the total computational budget, or computing infrastructure used. Although it mentions the use of specific models like BERT and the number of GPUs used for training, it does not provide detailed information about the model sizes or computational resources.
Yes	No	Section 3.2	The text does not mention the number of parameters in the models used, the total computational budget (e.g., GPU hours), or computing infrastructure used. The models used are RoBERTa-large models, but the text does not provide information about the number of parameters or computational resources used.
Yes	No	Please see Section 4.1.	The text does not mention the number of parameters in the models used, the total computational budget (e.g., GPU hours), or computing infrastructure used.

Table 3: Discrepancies on checklist question C1 in ACL Findings Long. The justifications provided are from CRAG and MD outputs generated using Llama 3.1 405B. All example have human answers "Yes" and LM answers "No," reflecting stricter model standards for reporting parameters, compute budget, and infrastructure.

ACL checklists featured labeled questions (e.g., A1) with symbolic ticks and optional justifications; answers were matched using proximity-based heuristics.

4. **Structured Storage** The extracted checklist information was standardized, manually reviewed, and stored in structured Excel sheets for incremental updates. Each paper was also mapped to its arXiv `.tar.gz` TeX source to allow linking between TeX and PDF files.

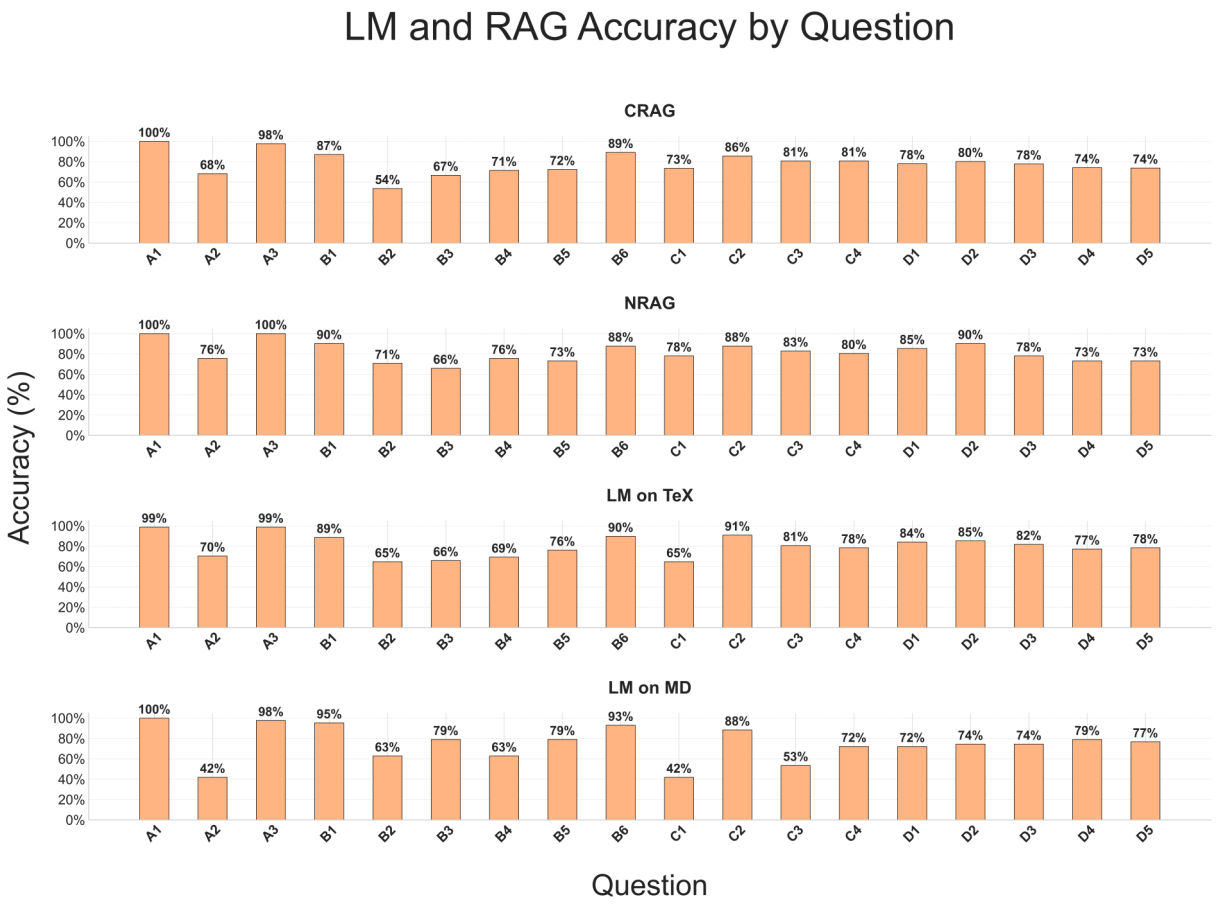


Figure 6: Accuracy by checklist question for ACL Findings Long on Llama-3.1-405B across four setups: CRAG, NRAG, LM on TeX, and LM on MD.

TokenSmith: Streamlining Data Editing, Search, and Inspection for Large-Scale Language Model Training and Interpretability

Mohammad Aflah Khan[1]*, **Ameya Godbole**[2]*,
Johnny Tian-Zheng Wei[2], **Ryan Wang**[2], **James Flemings**[2],
Krishna P. Gummadi[1], **Willie Neiswanger**[2], **Robin Jia**[2]
[1]Max Planck Institute for Software Systems, [2]University of Southern California
Correspondence: afkhan@mpi-sws.org, ameyagod@usc.edu

Abstract

Understanding the relationship between training data and model behavior during pretraining is crucial, but existing workflows make this process cumbersome, fragmented, and often inaccessible to researchers. We present TokenSmith, an open-source library for interactive editing, inspection, and analysis of datasets used in Megatron-style pretraining frameworks such as GPT-NeoX, Megatron, and NVIDIA NeMo. TokenSmith supports a wide range of operations including searching, viewing, ingesting, exporting, inspecting, and sampling data, all accessible through a simple user interface and a modular backend. It also enables structured editing of pretraining data without requiring changes to training code, simplifying dataset debugging, validation, and experimentation. TokenSmith is designed as a plug-and-play addition to existing large language model pretraining workflows, thereby democratizing access to production-grade dataset tooling.

TokenSmith is hosted on GitHub[1], with accompanying documentation and tutorials[2]. A demonstration video is also available on YouTube.[3]

1 Introduction

The barrier to pretraining large language models from scratch has been rapidly declining, driven by improved access to GPUs, a growing number of open-source frameworks, and the widespread sharing of technical knowledge. As a result, academic groups, open-source organizations and hobbyists are increasingly able to conduct meaningful pretraining research (Biderman et al., 2023; Azerbayev et al., 2023; Chi et al., 2023; Yin et al., 2023; Gupta et al., 2023; Horawalavithana et al., 2022; Ibrahim et al., 2024; Gao et al., 2025a,b; Zeng et al., 2024).

However, a persistent challenge across existing frameworks is the lack of robust tooling for inspecting and interacting with the training data. Tasks such as debugging loss spikes by tracing relevant datapoints, generating modified datasets for counterfactual experiments or decontamination, and even viewing specific batches or sequences remain cumbersome in open-source setups. For example, producing a counterfactual dataset typically requires manually identifying files, ensuring token alignment, and re-tokenizing the entire corpus (a process that can take over a day for large datasets).

We introduce TokenSmith, a toolkit designed to make this process seamless. TokenSmith addresses these gaps by providing intuitive abstractions for editing, inspecting, and managing datasets, thereby enabling faster iteration and deeper insight throughout the pretraining workflow.

TokenSmith is built on top of Megatron-LM (Shoeybi et al., 2020), a widely adopted and scalable framework for large language model pretraining. Several popular libraries such as GPT-NeoX (Andonian et al., 2023) and NVIDIA NeMo[4] are also based on Megatron-LM, sharing its dataset format and training pipeline structure. Beyond these, a number of direct forks leverage Megatron-LM for model training, such as K2[5] (Liu et al., 2025), MAP-NEO[6] (Zhang et al., 2024), Megatron-Llama[7], and Apertus[8]. The framework also serves as a foundation for research into more efficient training methods (Qi et al., 2023, 2024; Ao et al., 2024; Yuan et al., 2024; Wan et al., 2025), with

*Equal contribution
[1]https://github.com/aflah02/TokenSmith
[2]https://aflah02.github.io/TokenSmith/
[3]https://www.youtube.com/watch?v=cDO8VE9fZvU
[4]https://github.com/NVIDIA/NeMo
[5]https://github.com/LLM360/k2-train
[6]https://github.com/multimodal-art-projection/Megatron-LM-NEO
[7]https://github.com/alibaba/Megatron-LLaMA
[8]https://huggingface.co/collections/swiss-ai/apertus-llm-68b699e65415c231ace3b059

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 678–687
November 4-9, 2025

public implementations available.[9,10,11] Finally, adoption extends beyond NVIDIA GPUs, as GPU providers such as AMD also maintain Megatron-LM support.[12]

This shared foundation allows TokenSmith to natively support all three frameworks and other forks with minimal integration overhead. Additionally, TokenSmith is designed to be extensible, making it easy to add support for other frameworks.

2 Library Offerings

We build TokenSmith for practitioners and researchers working directly with large-scale language model pretraining. Our goal is to make it significantly easier to address a wide range of research questions and engineering challenges that arise when working with massive datasets and opaque training processes. Instead of offering a monolithic interface, we emphasize modularity and extensibility, providing intuitive abstractions through a simple frontend and a well-documented backend that can be adapted to different workflows. Here we describe the key functionalities we support: Inspect, Sample, Edit, Export, Ingest, and Search.

2.1 Inspecting and Sampling Datasets

Understanding the relationship between training data and model behavior is a recurring challenge in large-scale pretraining. Practitioners often need to trace issues back to specific sequences or isolate dataset subsets for hypothesis testing. Typical questions include:

- *How can we trace and identify sequences that correlate with sudden spikes in training loss?*
- *Between two model checkpoints, what new data did the model encounter, and how might it explain improvements or regressions in performance?*
- *Are there tokenization issues or formatting inconsistencies that may have gone unnoticed?*
- *What happens if the model is trained only on a specific subset, such as domain-specific documents or repeated early-stage data?*
- *Can we sample sequences based on properties such as length, content patterns, or document metadata?*

TokenSmith provides a unified set of tools to support both deep inspection and flexible sampling:

- **Precise inspection utilities** allow users to locate and analyze data at the level of individual sequences, batches, or training steps using indices or global step numbers.
- **Sampling utilities** support extracting subsets of the data based on custom policies, enabling rapid prototyping, ablation studies, and behaviorally targeted dataset construction.
- **Modular integration** through a backend API allows seamless incorporation into training pipelines, analysis scripts, or interactive UIs.

By enabling structured, reproducible interrogation of the training dataset, TokenSmith empowers practitioners to move from anecdotal debugging to systematic, data-driven understanding of model behavior.

2.2 Editing Datasets

As training progresses or evaluation findings emerge, practitioners frequently encounter the need to modify the dataset. These changes are often motivated by new insights or requirements, such as:

- *Removing specific batches that correlate with spikes in training loss*
- *Filtering out examples that may result in test set leakage or data contamination*
- *Creating counterfactual variants of the dataset for controlled experiments, such as ablation studies or robustness analysis*

To support such use cases, TokenSmith offers a flexible editing interface that allows for **targeted edits**, enabling users to directly specify and modify individual sequences.

These capabilities allow researchers to iterate on dataset versions without re-engineering the training pipeline. Edits can be performed programmatically and are fully compatible with the inspection and sampling modules of the library. This makes it possible to run sophisticated data-centric experiments, such as testing the effect of subtle perturbations, while maintaining full control over what the model sees during training.

[9] https://github.com/thunlp/Seq1F1B
[10] https://github.com/sail-sg/zero-bubble-pipeline-parallelism
[11] https://github.com/kwai/Megatron-Kwai
[12] https://github.com/ROCm/Megatron-LM

2.3 Exporting Datasets

Beyond sampling, reproducibility and interoperability are essential in dataset-centric research. Some recurring challenges are:

- *How can we verify and share a specific version of the dataset used for a paper, in a reproducible format compatible with popular libraries like HuggingFace Datasets?*
- *Can we export only specific batches/sequences of a dataset that show interesting trends to avoid sharing large binaries?*

To this end, TokenSmith includes **export tools** for converting datasets (entirely or in parts) into formats such as JSONL and CSV which are also HuggingFace compatible. These tools enable seamless sharing, integration with external pipelines, and long-term reproducibility of experimental results.

2.4 Ingesting Datasets

Curating datasets for large-scale pretraining often begins with converting diverse data sources into a format compatible with Megatron-style frameworks. However, this step is frequently underdocumented and error-prone. A common challenge is: *How can we ingest and tokenize new datasets into the Megatron binary format without writing custom conversion pipelines?*

TokenSmith addresses this through streamlined **ingestion utilities** that support converting standard formats such as JSONL and CSV into the required .bin/.idx representation. These tools reduce the overhead of dataset preparation and ensure seamless compatibility with Megatron-based pretraining workflows.

2.5 Searching Datasets

As pretraining datasets grow in size and complexity, being able to efficiently search and retrieve relevant content becomes essential for both debugging and targeted experimentation. Practitioners and researchers often encounter challenges such as:

- *How can we locate all occurrences of a specific phrase, token, or n-gram to inspect or remove sensitive or duplicate content?*
- *Can we curate the likely continuations given a naive n-gram model to contrast the generation likelihoods of our LLMs?*
- *Is it possible to trace model behaviors to specific textual patterns or domains within the training set?*
- *How do we efficiently support search at scale without loading the entire dataset into memory?*

To address these challenges, TokenSmith builds abstractions over Tokengram[13], an efficient n-gram indexing and search tool optimized for large-scale corpora. This allows users to perform fast searches over pre-tokenized corpora. Integrating Tokengram in TokenSmith provides support for end-to-end data interventions. For example, the search results can be processed with the Inspect and Export utilities for easy sharing with your collaborators or downstream post-processing. The matched documents from the search results can be processed with the Edit utilities. This might be useful to mask out toxic text, anonymize documents in place, etc.

By making dataset search fast and programmatically accessible, TokenSmith empowers users to build more informed training sets, track down model behaviors to specific training signals, and conduct controlled data-centric research at scale.

3 Practical Case Studies

In this section, we provide examples of how TokenSmith could simplify pipelines for NLP research.

3.1 Training Dynamics of Memorization

Several research groups have attempted to study memorization of natural (Huang et al., 2024; Jagielski et al., 2023) or counterfactually curated (Chang et al., 2025; Wei et al., 2024) data during LM pretraining. In order to study training dynamics, these projects relied on one of two approaches:

1. **Re-tokenizing the corpus:** Wei et al. (2024) studied the memorization of watermarks in pre-training by injecting randomized strings in groups of documents. They re-tokenize the entire corpus along with different sets of watermarked documents. Note that the number of modified documents is a small fraction of the full corpus; thus, they spend considerable time re-tokenizing unchanged data. Moreover, their approach cannot control the order of unchanged documents in different training runs.

[13] https://github.com/EleutherAI/tokengrams

2. **Modifying the training library**: Huang et al. (2024) and Chang et al. (2025) study verbatim memorization of documents in the early, middle, and late stages of pre-training by injecting curated/synthetically generated documents in specific training sequences. They achieve this by modifying the data loader and training loop of the pre-training library (Andonian et al., 2023; Team OLMo et al., 2024). This engineering-intensive approach requires a deep understanding of the underlying pre-training libraries. Moreover, this may decrease the training efficiency of the library.

In contrast, TokenSmith allows you to directly edit the tokenized dataset (§ 2.2). This allows you to perform the same experiments (1) without having to re-tokenize documents that haven't changed between training runs, and (2) without modifying the pre-training libraries.

3.2 Identifying Causes of Instability

Team OLMo et al. (2024) highlight that sudden spikes in training loss can lead to instability further along in pre-training and worse final model performance. In order to debug the cause of the instability, they inspect the batches of data that caused the spikes and identify that the batches contain training sequences with repeated n-grams. The Inspect and Export tools in TokenSmith provide a straightforward interface to extract batches based on the step number (where the loss spike occurred).

4 Library Design

TokenSmith is designed to support two complementary modes of interaction: a Pythonic API for seamless integration into existing training or analysis pipelines, and a visual UI for interactive exploration and inspection.

4.1 Overview: Megatron Dataset Format

Megatron-LM's indexed format uses two files per data split. The .bin file contains raw token sequences (packed back-to-back) as a flat array of integers. The .idx file contains metadata and pointers into the .bin. Specifically, the index begins with a header (version, dtype, number of sequences, number of documents) and then lists, for each sequence, its length (number of tokens) and its byte offset in the .bin. It also records, for each document, which sequences belong to it. In effect, .idx lets the dataset class reconstruct which slice of the big token array corresponds to each example.

The consistency of this format across libraries allows TokenSmith to support them out-of-the-box with minimal adaptations, enabling seamless interoperability and inspection without requiring separate data handling logic for each framework.

4.2 Pythonic API

TokenSmith exposes a modular, object-oriented API that allows users to programmatically ingest, edit, search, sample, inspect, and export datasets. This API is well-suited for integration into training scripts, research notebooks, or batch processing workflows. Figures 1, 2, 3, 4 and 5 illustrate the core API patterns:

- **Figure 1** illustrates how users can configure and execute token-level search queries using a Tokengram-backed index over the dataset.
- **Figures 2 and 3** showcase the interfaces for inspecting specific sequences and sampling data according to user-defined policies.
- **Figure 4** presents the editing interface, which supports fine-grained, targeted modifications to existing sequences.
- **Figure 5** demonstrates the dataset ingestion and export utilities, which allow users to import new corpora and export subsets or modified datasets in standard formats.

The library is designed to latch onto your existing pretraining environments with minimal configuration (instructions outlined clearly in the README).[14]

```python
from tokensmith.manager import DatasetManager

dataset_manager = DatasetManager()

dataset_manager.setup_search(
    bin_file_path="tokenized_data.bin",
    search_index_save_path="search_index_tokenized_data.idx",
    vocab=2**16,
)

tokenized_string = [67, 45, 99] # List of tokens

count = dataset_manager.search.count(test_sample)
isPresent = dataset_manager.search.contains(test_sample)
positions = dataset_manager.search.positions(test_sample)
```

Figure 1: Search API

```python
from tokensmith.manager import DatasetManager

dataset_manager = DatasetManager()

dataset_manager.setup_edit_inspect_sample_export(
    dataset_prefix='data_tokenized_text_document',
    batch_info_save_prefix='batch_info',
    train_iters=100, train_batch_size=16, train_seq_len=2048, seed=42,
)

sample_0 = dataset_manager.inspect.inspect_sample_by_id(sample_id=0)

batch_0 = dataset_manager.inspect.inspect_sample_by_batch(
    batch_id=0, batch_size=4
)
```

Figure 2: Inspect API

```python
from tokensmith.manager import DatasetManager

dataset_manager = DatasetManager()

dataset_manager.setup_edit_inspect_sample_export(
    dataset_prefix='data_tokenized_text_document',
    batch_info_save_prefix='batch_info',
    train_iters=100, train_batch_size=16, train_seq_len=2048, seed=42,
)

def sparse_sample_policy(start_index, num_samples, step_size=10):
    return [start_index + i * step_size for i in range(num_samples)]

def fibonacci_batch_policy(max_batch_id):
    fib = [1, 1]
    while fib[-1] < max_batch_id:
        fib.append(fib[-1] + fib[-2])
    return [f for f in fib if f < max_batch_id]

sparse_samples = dataset_manager.sample.get_samples_by_policy(
    policy_fn=sparse_sample_policy, start_index=50,
    num_samples=4, step_size=25
)

fib_batches = dataset_manager.sample.get_batches_by_policy(
    policy_fn=fibonacci_batch_policy, batch_size=2, max_batch_id=25,
)
```

Figure 3: Sample API

```python
from tokensmith.manager import DatasetManager

dataset_manager = DatasetManager()

dataset_manager.setup_edit_inspect_sample_export(
    dataset_prefix='data_tokenized_text_document',
    batch_info_save_prefix='batch_info',
    train_iters=100, train_batch_size=16, train_seq_len=2048, seed=42,
)

dataset_manager.edit.inject_and_preview(
    text="This is a test sentence", tokenizer=tokenizer,
    injection_loc=20, injection_type="seq_shuffle", add_eos_token=True,
    dry_run=True  # Set to False to not only preview but also perform
)
```

Figure 4: Edit API

4.3 Interactive UI

To enable visual exploration and rapid debugging, TokenSmith provides an intuitive user interface built with Streamlit.[15] The UI serves both as a reference implementation and a customizable layer for users to extend based on their workflows. It supports interactive search, batch and sequence level inspection, and document viewing. Figures 6a and 6b show examples of the inspect and document viewing pages, while Figure 8 (Appendix B) illustrates different search modes. Users can browse individual sequences or batches to trace issues such

[14] https://github.com/aflah02/tokensmith?tab=readme-ov-file#-quick-start

[15] https://streamlit.io/

```python
from tokensmith.manager import DatasetManager

dataset_manager = DatasetManager()

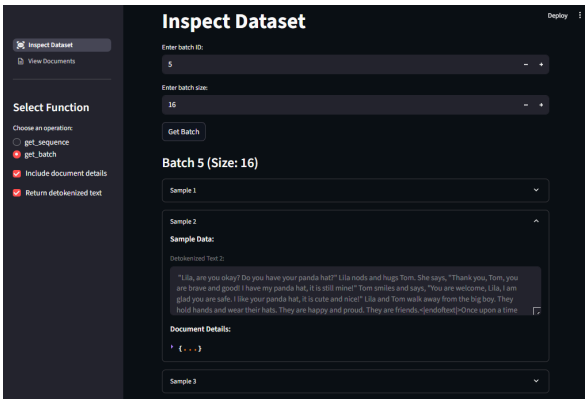
dataset_manager.ingest.ingest_from_jsonl(
        input_jsonl_path='data.jsonl', output_prefix='data_tokenized',
        vocab_path='tokenizer.json', neox_dir='gpt-neox',
        workers=8, append_eod=True, dataset_impl='mmap',
        tokenizer_type='HFTokenizer',
 )

dataset_manager.setup_edit_inspect_sample_export(
    dataset_prefix='data_tokenized_text_document',
    batch_info_save_prefix='artifacts/batch_info',
    train_iters=100, train_batch_size=16, train_seq_len=2048, seed=42,
)

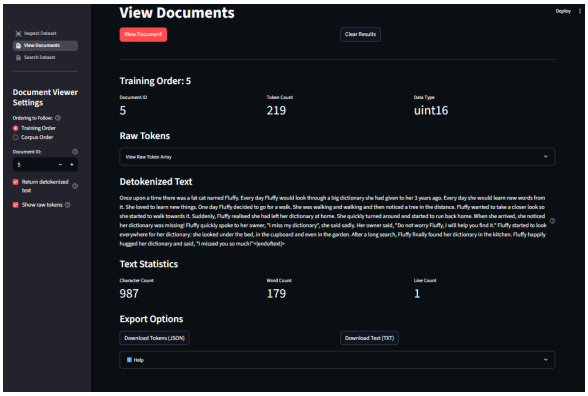
manager.export.export_sequences(
    sequence_indices=[100, 200, 300],
    output_path="exports/sequences.csv", format_type="csv",
)

manager.export.export_batch_range(
    start_batch=5, end_batch=10,  # Exports batches 5-9
    batch_size=32, output_path="batch_range.csv", format_type="csv",
    return_detokenized=True, tokenizer=tokenizer, flatten_batches=True
)
```

Figure 5: Ingest and Export API



(a) Inspect UI showing detailed information for a selected batch, including tokenized and detokenized sequences and document metadata.



(b) View Document UI for browsing individual documents and their tokenized representations.

Figure 6: TokenSmith inspection and viewing interfaces for exploring dataset contents at the batch and document levels.

as loss spikes, search for specific phrases, locate them within documents, and explore next token distributions, all through simple point and click interactions.

Together, the API and UI provide a unified and flexible interface to dataset management, enabling both hands-on experimentation and automated workflows at scale.

4.4 Design Patterns

TokenSmith is built with a strong emphasis on clean software engineering to ensure ease of use, extensibility, and long-term maintainability. Its architecture follows established design patterns to provide a clear separation of concerns, enable safe experimentation, and support both research and production environments. These design choices also make it easier for contributors to extend and integrate the toolkit with custom workflows. A detailed breakdown of the patterns used, including handler-based modularity, a facade interface, and runtime configurability, is provided in Appendix A.

5 Benchmarking

While there are no established baselines for many of the operations supported by TokenSmith we compare against the de facto workflows that practitioners currently rely on to achieve similar outcomes. These comparisons highlight the complexity and overhead of existing approaches, and demonstrate how TokenSmith streamlines them.

- **Editing a dataset to produce a counterfactual version:** Consider the case where a researcher wants to generate a modified dataset that differs by only a few examples from an original corpus spanning hundreds of billions of tokens.
  - *Current workflow:* Manually identify and replace relevant files, ensure token alignment (if needed), and re-tokenize the entire dataset. This process is brittle and time-consuming; for example, tokenizing a 500B-token corpus can take over a day depending on the system configuration.
  - *With TokenSmith:* Users can programmatically perform targeted or randomized edits directly on the tokenized dataset using our editor interface. This removes the need to reason about token boundaries or initiate a full re-tokenization pass, significantly reducing iteration time and engineering overhead.
- **Sampling Sequences According to Custom Policies:**
  - *Without TokenSmith :* Practitioners must manually extract sequences from the binary files, align them with training indices, implement policy-based filtering logic, and reconstruct the final subset (often requiring non-trivial changes to the existing pipeline).
  - *With TokenSmith :* The sampling API abstracts away these complexities, allowing users to specify high-level parameters and a custom policy function to obtain the desired subset with minimal effort.

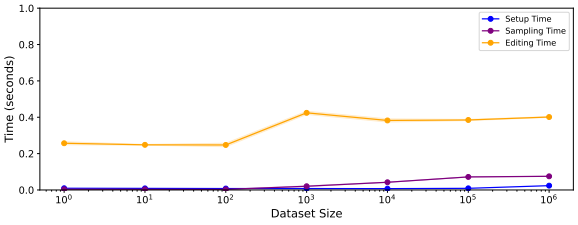


Figure 7: Execution time for setup, sampling, and editing operations across varying dataset sizes. For sampling and editing, the reported times correspond to 100 operations.

We also examine the scalability of our system by measuring the time taken for three representative operations as dataset size increases.

5.1 Dataset Setup

We measure the time required to execute the `setup_edit_inspect_sample_export` method, which initializes all necessary handlers based on the user's configuration. This step introduces negligible overhead, remaining under 0.03 seconds even for corpora with one million documents.

5.2 Sampling Performance

We randomly sample 100 sequence indices and measure the total retrieval time. This process is repeated five times, and we report the average and standard deviation. A single-item fetch is performed beforehand to warm up the system. Sampling remains stable across dataset sizes. For example, sampling 100 sequences from a 1M-document corpus takes under 0.1 seconds.

5.3 Editing Time

To assess editing performance, we insert the sentence (`This is a test sentence.`) at 100 ran-

dom positions across the dataset, following an initial warm-up edit. This operation is repeated five times, with average time and standard deviation reported. Edit latency remains under 0.5 seconds even for a 1M-document corpus, indicating minimal sensitivity to dataset size.

We showcase the benchmarking results in Figure 7. These results highlight that TokenSmith offers low-latency interactivity, making it well-suited for both rapid experimentation and large-scale dataset manipulation. For search-related benchmarks, we refer readers to Tokengrams' evaluations,[16] as TokenSmith directly integrates Tokengrams for all token-level search operations.

Before each benchmark measurement, we reinstantiate the `DatasetManager` and explicitly trigger garbage collection using `gc.collect()` from Python's `gc` module.[17] The benchmarking script, along with the corresponding results, is available in the repository.[18]

6 Conclusion

We present TokenSmith, a modular and extensible toolkit designed to streamline dataset-centric workflows in Megatron-style LLM pretraining. By offering intuitive abstractions for ingesting, editing, inspecting, sampling, searching, and exporting training data, TokenSmith fills a crucial gap in current open-source infrastructure. TokenSmith's support for multiple backends, efficient operations at scale, and dual interface (Pythonic API and visual UI) makes it accessible to researchers, practitioners, and hobbyists alike. As the LLM ecosystem increasingly embraces open and reproducible research, we believe TokenSmith will serve as a practical foundation for understanding, debugging, and experimenting with the data that drives modern language models.

Acknowledgments

We thank the EleutherAI team for open-sourcing Tokengrams and GPT-NeoX, and for their helpful responses to our questions. We also acknowledge the contributions of NVIDIA's Megatron and GPT-NeoX repositories, which serve as foundational components in our work. The results presented in this work used compute resources from the National AI Research Resource Pilot, with support from NVIDIA, including NVIDIA's DGX Cloud product and the NVIDIA AI Enterprise Software Platform. This work was supported in part by a gift from the USC-Amazon Center on Secure and Trusted Machine Learning, and the National Science Foundation under Grant No. IIS-2403436. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Finally, we note that large language models were used to assist in editing and refining the writing of this paper.

Licensing

TokenSmith is released under the Apache 2.0 license. This permissive license allows for both academic and commercial use, as well as modification and redistribution, making it suitable for a wide range of research and production workflows.

References

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, and 2 others. 2023. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch.

Sun Ao, Weilin Zhao, Xu Han, Cheng Yang, Zhiyuan Liu, Chuan Shi, and Maosong Sun. 2024. Seq1f1b: Efficient sequence-level pipeline parallelism for large language model training. *arXiv preprint arXiv:2406.03488*.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W. Ayers, Dragomir Radev, and Jeremy Avigad. 2023. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *Preprint*, arXiv:2302.12433.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. 2025. How do large language models

[16] https://github.com/EleutherAI/tokengrams?tab=readme-ov-file#performance

[17] https://docs.python.org/3/library/gc.html

[18] https://github.com/aflah02/TokenSmith/tree/main/benchmarking

acquire factual knowledge during pretraining? In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Ta-Chung Chi, Ting-Han Fan, Alexander Rudnicky, and Peter Ramadge. 2023. Dissecting transformer length extrapolation via the lens of receptive field analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13522–13537, Toronto, Canada. Association for Computational Linguistics.

E. Gamma. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley professional computing series. Pearson Education.

Tianyu Gao, Alexander Wettig, Luxi He, Yihe Dong, Sadhika Malladi, and Danqi Chen. 2025a. Metadata conditioning accelerates language model pre-training. *Preprint*, arXiv:2501.01956.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. 2025b. How to train long-context language models (effectively). *Preprint*, arXiv:2410.02660.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. 2023. Continual pre-training of large language models: How to (re)warm your model? *Preprint*, arXiv:2308.04014.

Sameera Horawalavithana, Ellyn Ayton, Shivam Sharma, Scott Howland, Megha Subramanian, Scott Vasquez, Robin Cosbey, Maria Glenski, and Svitlana Volkova. 2022. Foundation models of scientific knowledge for chemistry: Opportunities, challenges and lessons learned. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 160–172, virtual+Dublin. Association for Computational Linguistics.

Jing Huang, Diyi Yang, and Christopher Potts. 2024. Demystifying verbatim memorization in large language models. *Preprint*, arXiv:2407.17817.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. 2024. Simple and scalable strategies to continually pre-train large language models. *Preprint*, arXiv:2403.08763.

Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. 2023. Measuring forgetting of memorized training examples. In *The Eleventh International Conference on Learning Representations*.

Zhengzhong Liu, Bowen Tan, Hongyi Wang, Willie Neiswanger, Tianhua Tao, Haonan Li, Fajri Koto, Yuqi Wang, Suqi Sun, Omkar Pangarkar, Richard Fan, Yi Gu, Victor Miller, Liqun Ma, Liping Tang, Nikhil Ranjan, Yonghao Zhuang, Guowei He, Renxi Wang, and 6 others. 2025. Llm360 k2: Building a 65b 360-open-source large language model from scratch. *Preprint*, arXiv:2501.07124.

R.C. Martin. 2003. *Agile Software Development: Principles, Patterns, and Practices*. Alan Apt series. Pearson Education.

Penghui Qi, Xinyi Wan, Nyamdavaa Amar, and Min Lin. 2024. Pipeline parallelism with controllable memory. *Preprint*, arXiv:2405.15362.

Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. 2023. Zero bubble pipeline parallelism. *Preprint*, arXiv:2401.10241.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-lm: Training multi-billion parameter language models using model parallelism. *Preprint*, arXiv:1909.08053.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2024. 2 OLMo 2 Furious. *Preprint*, arXiv:2501.00656.

Xinyi Wan, Penghui Qi, Guangxing Huang, Min Lin, and Jialin Li. 2025. Pipeoffload: Improving scalability of pipeline parallelism with memory optimization. *Preprint*, arXiv:2503.01328.

Johnny Wei, Ryan Wang, and Robin Jia. 2024. Proving membership in LLM pretraining data via data watermarks. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13306–13320, Bangkok, Thailand. Association for Computational Linguistics.

Junqi Yin, Sajal Dash, Feiyi Wang, and Mallikarjun Shankar. 2023. Forge: Pre-training open foundation models for science. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '23, New York, NY, USA. Association for Computing Machinery.

Tailing Yuan, Yuliang Liu, Xucheng Ye, Shenglong Zhang, Jianchao Tan, Bin Chen, Chengru Song, and Di Zhang. 2024. Accelerating the training of large language models using efficient activation rematerialization and optimal hybrid parallelism. In *Proceedings of the 2024 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC'24, USA. USENIX Association.

Zhiyuan Zeng, Qipeng Guo, Zhaoye Fei, Zhangyue Yin, Yunhua Zhou, Linyang Li, Tianxiang Sun, Hang Yan, Dahua Lin, and Xipeng Qiu. 2024. Turn waste into worth: Rectifying top-k router of moe. *Preprint*, arXiv:2402.12399.

Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, Raven Yuan, Tuney Zheng, Wei Pang, Xinrun Du, Yiming Liang, Yinghao Ma, Yizhi Li, Ziyang Ma, Bill Lin, and 26 others. 2024. Map-neo: Highly capable and transparent bilingual large language model series. *Preprint*, arXiv:2405.19327.

A Design Patterns Employed in the Library

TokenSmith is structured around well-established software design principles that promote modularity, extensibility, and maintainability. The internal architecture is intentionally clean and componentized to accommodate both research prototyping and production-scale workflows. Below, we describe the primary design patterns used in the codebase.

Handler Pattern (Command or Service Object) (Gamma, 1995) Each major functional area (editing, inspecting, sampling, exporting, and searching) is encapsulated within its own handler class. For instance, EditHandler, InspectHandler, SampleHandler, ExportHandler, and SearchHandler each manage their domain-specific logic while exposing a consistent interface. This clear separation of concerns makes the system easier to extend, test, and reason about.

Facade Pattern (Gamma, 1995) The DatasetManager class serves as a unified entry point to the system. It orchestrates the initialization of handlers and provides a high-level API to the end user. This shields users from internal complexities and reduces the cognitive load involved in accessing multiple capabilities.

Dependency Injection Rather than relying on tight coupling or global state, handlers receive references to the DatasetManager or its specific components during initialization. This inversion of control enhances testability and supports future decoupling and modular reuse.

Type Hinting and Forward References To avoid circular dependencies while retaining strong type safety, the library uses TYPE_CHECKING blocks and string-based type annotations (e.g., 'DatasetManager'). This allows static analyzers and IDEs to provide full support while maintaining clear dependency boundaries.[19]

Strategy Pattern (Configurable Behavior) (Gamma, 1995) Handlers expose methods whose behavior can be configured at runtime via parameters. For example, the EditHandler supports multiple injection strategies. This makes the system adaptable for experimentation without requiring internal changes to core logic.

Template Method Pattern (Gamma, 1995) Export operations follow a template structure in which base methods (such as export) provide a standard workflow but allow subclasses or extensions to override certain steps. This approach encourages consistent behavior while allowing flexibility for future extensions or format support.

Validation and Defensive Programming Robust input validation and error handling are systematically applied across the codebase. Although not a formal design pattern, this practice contributes significantly to the reliability and maintainability of the library, especially in high-scale or adversarial settings.

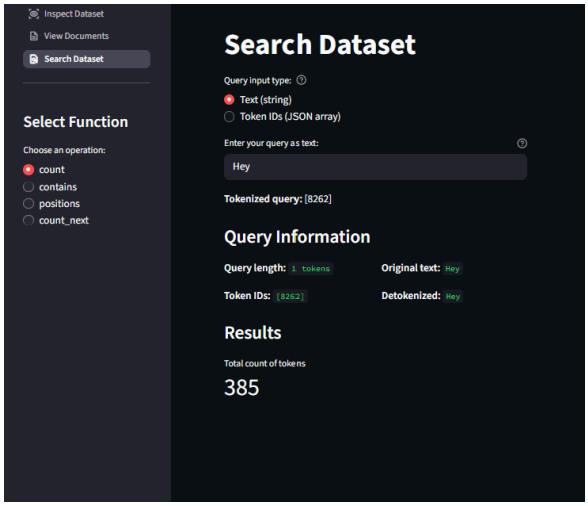
Modular Package Structure The code is divided into clearly defined submodules (edit, inspect, sample, export, search), with each exposing a single handler class through its __init__.py. This supports the Single Responsibility Principle (Martin, 2003) and allows contributors to quickly locate, understand, and extend functionality.

These design decisions collectively ensure that TokenSmith remains extensible and maintainable as it grows to support additional backends, workflows, and research use cases.

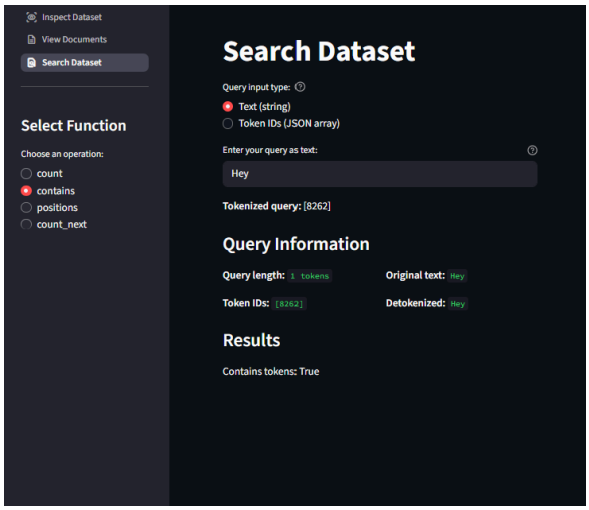
B User Interface

Figure 8 showcases different components of the TokenSmith search interface, illustrating how users can query token counts, presence, positions of occurrence, and likely next tokens using an n-gram model—all through intuitive visual tools.

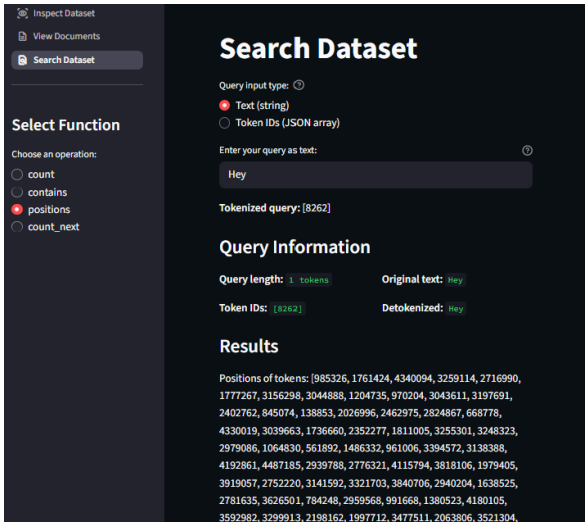
[19] https://peps.python.org/pep-0484/



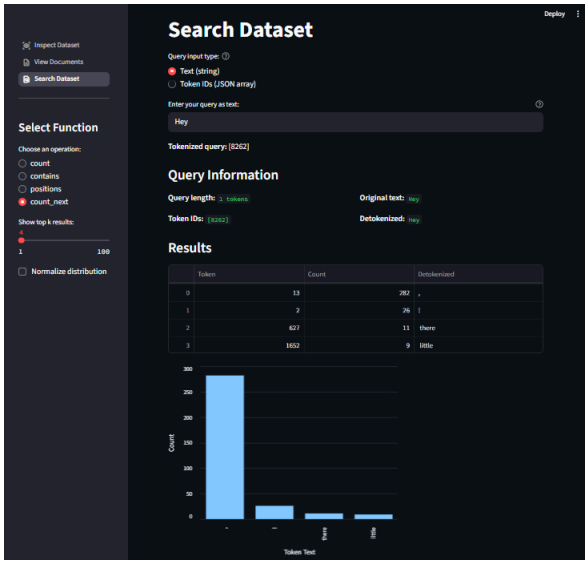
(a) Search for count



(b) Search for presence



(c) Search for positions of occurrence



(d) Search for likely next tokens using an n-gram model

Figure 8: TokenSmith search UI showing different search strategies: filtering by count, containment, positional match, and context-based continuation.

LLM×MapReduce-V3: Enabling Interactive In-Depth Survey Generation through a MCP-Driven Hierarchically Modular Agent System

Yu Chao[1*] **Siyu Lin**[2*] **Xiaorong Wang**[3] **Zhu Zhang**[1] **Zihan Zhou**[3]
Haoyu Wang[4] **Shuo Wang**[1†] **Jie Zhou**[3] **Zhiyuan Liu**[1†] **Maosong Sun**[1†]
[1]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, Tsinghua University
[2]Peking University [3]Modelbest Inc. [4]Nanyang Technological University

Abstract

We introduce **LLM×MapReduce-V3**, a hierarchically modular agent system designed for long-form survey generation. Building on the prior work, LLM×MapReduce-V2, this version incorporates a multi-agent architecture where individual functional components, such as skeleton initialization, digest construction, and skeleton refinement, are implemented as independent model-context-protocol (MCP) servers. These atomic servers can be aggregated into higher-level servers, creating a hierarchically structured system. A high-level planner agent dynamically orchestrates the workflow by selecting appropriate modules based on their MCP tool descriptions and the execution history. This modular decomposition facilitates human-in-the-loop intervention, affording users greater control and customization over the research process. Through a multi-turn interaction, the system precisely captures the intended research perspectives to generate a comprehensive skeleton, which is then developed into an in-depth survey. Human evaluations demonstrate that our system demonstrates strong performance in both content depth and length, highlighting the strength of MCP-based modular planning.[1]

1 Introduction

Generating high-quality long-form survey articles poses significant challenges to AI Agent systems (Li et al., 2024; Zhang et al., 2025b; Qin et al., 2024). First, the system must effectively aggregate and synthesize information from a large collection of reference materials (Nakano et al., 2021; Asai et al., 2023; Huang et al., 2024), which span various subdomains and perspectives. Second, it requires the ability to construct a coherent and comprehensive skeleton that guides the organization of content at a global level (Wang et al., 2024; Wen et al., 2025). Third, as the upstream component, reference exploitation exerts a significant influence (Elovic, 2024), thus many AI-based survey generation tools have begun to focus on improving the quality of search results before the writing stage begins. Addressing these challenges demands controllable mechanisms for information acquisition, structural planning, and content generation.

To address these challenges, we propose **LLM×MapReduce-V3**, an interactive and self-organized modular agents system for long-form survey generation. Our system builds upon the design of LLM×MapReduce-V2 (Wang et al., 2025), but extends it in several important ways to enable modularity, adaptability, and dynamic planning (Qiu et al., 2025b,a). At the core of our approach is the use of the model context protocol (MCP), a standardized function-calling mechanism that allows tools and modules to be composed as independent MCP servers (Raseed, 2025; Unleash, 2025). We proposes a multi-stage workflow of document digestion, skeleton construction and refinement, and survey writing. Building upon this foundation, we re-architect the system into a multi-agent paradigm, wherein core functionalities are decomposed across specialized agents, and the algorithmic workflow is encapsulated as a suite of tools in MCP servers (Anthropic, 2024) to facilitate agent-level invocation and coordination.

A key improvement in our system lies in pipeline flexibility with an agent planner (Moura et al., 2024; Link-AGI, 2024). Rather than following a fixed control flow, the agent receives a set of available MCP tools along with previous outputs, and dynamically selects the next module to invoke. This enables non-linear, adaptive workflows tailored to the specific needs of each writing task. To support user intent alignment, we also introduce

[*]Equal contribution.
[†]Corresponding authors.
[1]The code is publicly available at https://github.com/thunlp/LLMxMapReduce.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 688–695
November 4-9, 2025

human-in-the-loop interaction. Users begin by providing a target topic and basic writing instructions. The system then engages in a multi-turn dialogue to identify the user's preferred and fine-grained research perspectives.

To thoroughly examine the performance of LLM×MapReduce-V3, we conduct a human evaluation to assess system effectiveness. Domain experts compare the output of our system with those from other popular deep research systems across multiple topics. The results suggest that our system generates more informative outlines and higher-quality, more in-depth survey articles. Our main contributions are summarized as follows:

- **Methodological Innovation**: We propose the first MCP-based modular agent system for academic survey generation, enabling unprecedented customization and institutional integration while maintaining survey quality standards.
- **Architectural Advancement**: We propose a dynamic, LLM-driven planner that supports multi-stage module orchestration for adaptive, non-static workflows.
- **User-Centric Design**: We design a human-in-the-loop interaction framework that ensures generated surveys align with user expertise and research perspectives, bridging the gap between automation and scholarly rigor.

2 Related Work

2.1 AI-Powered Automated Research

Early automated research systems focused on web-based information retrieval, with WebGPT (Nakano et al., 2021) and Self-RAG (Asai et al., 2023) pioneering LLM-based web browsing through human feedback or self-reflection mechanisms for adaptive retrieval and generation. Recent advances include sophisticated autonomous agents like GPT-Researcher (Elovic, 2024) with dual-agent architectures, ResearchAgent (Huang et al., 2024) for iterative research idea generation through multi-agents collaboration, and products including Perplexity Deep Research (AI, 2024) and ChatGPT Deep Research (OpenAI, 2024). While some closed-source deep research systems can produce valuable results, we contend that flexible user involvement is critical for a truly practical system. We therefore propose an open-source, hierarchically modular system specifically designed to facilitate and support human intervention throughout the research process.

2.2 Survey Generation

LLM-driven survey generation has emerged with AutoSurvey (Wang et al., 2024), which introduced a four-stage methodology that addresses context limitations and evaluation challenges, while InteractiveSurvey (Wen et al., 2025) advances through personalized, interactive generation with continuous user customization of reference categorization and content synthesis. SurveyX (Liang et al., 2025), on the other hand, focuses on extracting topics and content by pre-organizing the literature into the form of an attribute tree. Despite these advances, existing systems often lock users into a rigid, "all-or-nothing" paradigm. Stemming from inflexible integration approaches, they lack the necessary interfaces for iterative refinement and specialized customization. In response, we adopt MCP to modularize our algorithmic workflow, enabling the agents to iteratively select and invoke appropriate tools for survey generation.

2.3 Model Context Protocol and Modular Self-Organized Agent System

MCP (Anthropic, 2024) establishes open standards for connecting AI assistants to diverse tool sources through a unified client-server architecture. Recent applications demonstrate its potential for multi-agents intelligence (Raseed, 2025) and scalable systems (Unleash, 2025). Alita (Qiu et al., 2025b) leverages MCP to autonomously construct and reuse external capabilities through task-related protocols, achieving scalable agentic reasoning with minimal pre-definition. AgentDistill (Qiu et al., 2025a) enables training-free knowledge transfer via reusable MCP boxes, allowing smaller agents to achieve performance comparable to large LLM systems. These MCP-based self-evolution approaches demonstrate the its potential for adaptive agent systems. Our work first introduces the MCP-based hierarchically modular survey generation system, combining protocol standardization with specialized academic optimizations to enable unprecedented user customization and institutional integration while maintaining the quality of survey paper generation.

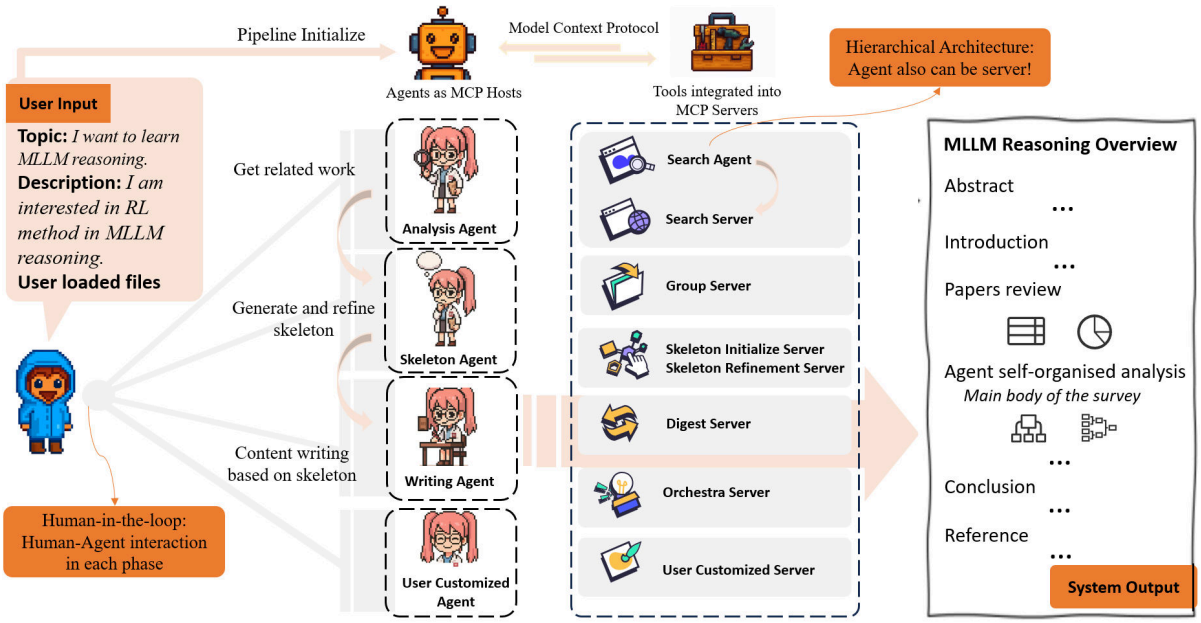


Figure 1: Our agent-server ecosystem pipeline. Users begin by specifying a topic, optionally adding detailed descriptions or uploading documents. The Analysis Agent interprets user intent and coordinates with the Search Agent to retrieve and organize relevant literature. The Skeleton Agent then generates and refines an outline, which is used by the Writing Agent to complete the paper.

3 Hierarchically Modular Agent System

Our system employs a multi-agent paradigm where specialized agents handle distinct phases of survey generation, each independently connecting to both algorithm-internal and user-provided MCP servers.

3.1 System Design

Let $\mathcal{A} = \{A_1, A_2, A_3\}$ denote the set of specialized agents, where A_1, A_2, A_3 denote the Analysis Agent, Skeleton Agent, and the Wirting Agent, respectively. The MCP server ecosystem can be represented as $\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5, S_6, S^*\}$, including the Search Server, Group Server, Skeleton Initialization Server, Digest Server, Skeleton Refinement Server, Orchestra Server, and user customized servers (i.e., S^*).

At each tool-calling round, the connection between the Agent and the Server is defined as $\mathcal{E}$, which is determined based on the previous output and the Agent's current plan.

$$\mathcal{E} = \text{MCP}(A_i(\text{output}_{i-1}, \text{plan}), \phi(\mathcal{A}_i))$$

The agent-server mapping is defined as:

$$\phi : \mathcal{A} \to 2^{\mathcal{S}}$$

where $\phi(A_i)$ specifies the server subset accessible to agent A_i.

The system architecture forms a directed graph $G = (\mathcal{A}, \mathcal{E}, \mathcal{S})$ where $\text{MCP}(\mathcal{A}_i, \mathcal{S}_i)$ represents agents communicate through standardized MCP interfaces.

Each server $S_i \in \mathcal{S}$ exposes a tool collection $\mathcal{T}(S_i) = \{t_{i,1}, t_{i,2}, ..., t_{i,k_i}\}$ through standardized MCP protocols. Tool invocation is formalized as:

$$\text{invoke} : \mathcal{A} \times \mathcal{T} \times \mathcal{I} \to \mathcal{O}$$

where $\mathcal{I}$ and $\mathcal{O}$ represent input and output spaces respectively.

3.1.1 Analysis Agent

The Analysis Agent orchestrates the initial literature processing and manages interaction with the Search Agent. Given a user-defined research topic and initial description, it first explores users' unwritten needs by multi-turn dialogue with users, thus enhance both retrieval width and depth.

It then invokes the search agent for references acquisition and integrates materials uploaded by the user. Subsequently, the agent applies clustering algorithms from the group server to group the references by thematic relevance and methodological similarity, constructing a structured representation of the retrieve result. Finally, the agent initializes a tree graph to store the grouped references, and all downstream data structures are organized by this tree.

3.1.2 Skeleton Agent

The Skeleton Agent is responsible for constructing a globally organized and content-aware skeleton, serving as the structural backbone of the generated survey article. It operates under the coordination of an orchestration module and proceeds through three main stages: *Skeleton Initialization*, *Digest Construction*, and *Skeleton Refinement*.

3.1.3 Writing Agent

The Writing Agent executes the final content synthesis, transforming the refined skeleton into coherent survey sections. This agent integrates literature digests, maintains citation consistency, and ensures academic writing standards while preserving the logical flow established by the skeleton structure.

To enhance generation quality, users may customize the writing agent to adapt to varying formatting specifications, support figure generation, and address other individualized requirements. Here we implement a dedicated figure server to support the generation of Mermaid-style diagrams.

3.2 External and Replaceable Agents

3.2.1 Search Agent

The Search Agent provides literature retrieval capabilities through the interface. This agent can be seamlessly replaced with alternative search implementations, including domain-specific databases, institutional repositories, or specialized academic search engines.

In parallel, the agent is also capable of acting as an MCP server. In our implementation, the Search Agent is supported by a dedicated search server, which integrates four core tools: query generation, web retrieval, crawling, and similarity analysis. This high-level server operates under the coordination of the Analysis Agent.

3.2.2 Other Agents

The system's extensibility is reflected in its support for user-defined agents—such as academic critics, formatting agent, and domain-specific analyzers—enabling institutional customization and specialized workflows while preserving overall coherence through standardized interfaces.

4 MCP Implementation Framework

Our implementation leverages MCP's client-server architecture to encapsulate core functionalities as reusable, composable modules. Each functional component operates as an independent MCP server, exposing standardized tool interfaces for agent invocation.

4.1 Native Server Construction

We re-organize the procedure proposed in LLM×MapReduce-V2 (Wang et al., 2025) into several MCP servers.

4.1.1 Group Server

The Group Server serves as a content-based organizer that preprocesses the retrieved reference corpus. Before structural planning begins, this module clusters the documents into coherent topical groups. This pre-organization significantly reduces topic fragmentation and provides a more stable input for downstream outline construction. The grouping strategy is guided by both thematic relevance and methodological similarity.

4.1.2 Orchestra Server

The orchestration of the skeleton construction process is managed by the Orchestra Server, which acts as a lightweight planner based on an LLM backbone. At each stage, it takes the current intermediate outputs as input, along with formal descriptions of the available servers. Based on these inputs, it generates next-step instructions. This centralized coordination allows the system to adaptively guide the overall skeleton-building process, ensuring alignment between user intent, module capability, and evolving outline structure. The Orchestra Server thus serves as a dynamic planner and command generator that orchestrates multi-module collaboration across the pipeline.

4.1.3 Skeleton Initialization Server

Given the refined research angles and grouped references provided by the Analysis Agent, the Skeleton Initialization Server constructs a high-level section-wise outline. This initial structure serves as a coarse-grained scaffold, segmenting the article into major thematic areas that reflect distinct facets of the research topic. Each section is derived by mapping the user-specified focus onto prominent subfields within the references, ensuring both topical coverage and logical segmentation.

4.1.4 Digest Server

The Digest Construction Server enhances the initial skeleton by generating content aware revision signals derived from the reference documents. It includes two steps: firstly, for each reference document, the system prompts an LLM to generate a

brief summary along with suggestions for improving the current outline. These digests reflect how individual sources align or conflict with the existing skeleton. Secondly, the system aggregates all digests and suggestions, merges redundant feedback, and synthesizes a consolidated revision plan. This output serves as a high-level guide for improving the outline's coverage, organization, and alignment with the source materials.

4.1.5 Skeleton Refine Server

Skeleton Refinement Server applies an iterative multi-layer convolution-inspired process to optimize the structure for coherence, consistency, and informativeness. This refinement mechanism operates both within sections (to enhance semantic alignment among digests) and across sections (to improve global representation and eliminate redundancy). This process simulates the multi-layer grouped convolution in Convolutional Neural Networks (CNN), which effectively integrates intra-group information and expand the contextual "receptive field" during information aggregation through a multi-layer iterative method. When the references are effectively aggregated, Skeleton Refine Server produces an in-depth and fine-grained survey skeleton, which also determines the structural integrity of the subsequent survey writing.

4.1.6 Iterative Refinement Through Multi-turn Tool-use

The refinement process is conceptualized as a multi-turn, tool-based self-evolution framework, wherein the agent acts as a dynamic coordinator. This coordination is guided by intermediate outputs and, when available, user feedback.

The Orchestra Server implements a planning function:

$$\pi : \mathcal{H} \times \mathcal{C} \to \mathcal{T}^*$$

where $\mathcal{H}$ denotes execution history, $\mathcal{C}$ represents current context, and $\mathcal{T}^*$ is the space of tool sequences.

At each decision point t, given skeleton state $x^{(t)} \in \mathcal{X}$ and history $h^{(t)} \in \mathcal{H}$, the agent selects action:

$$u^{(t)} = \pi(x^{(t)}, h^{(t)})$$

The refinement process operates through iterative state transitions:

$$x^{(t+1)} = f(x^{(t)}, u^{(t)})$$

Through this mechanism, the system progressively refines the survey skeleton and content analysis, enabling adaptive improvement across multiple iterations.

At each refinement iteration, skeleton agent consults the orchestra server to determine the optimal sequence of actions based on current skeleton state and available feedback. The Orchestra Server analyzes the current context and returns structured action plans which includes subsequent MCP server invocations, enabling adaptive workflow management based on real-time assessment of skeleton quality and user requirements.

Skeleton agent employs the Digest Server to generate targeted literature summaries for specific skeleton sections.

These digests are then integrated into the skeleton structure through coordinated calls to theSkeleton Server, which maintains structural consistency while incorporating new content elements.

4.2 Agent Integration and Replaceable Invocation

Our orchestration mechanism treats each MCP server as a callable tool, associated with relevant metadata, which allows the central planner to make informed decisions regarding the sequencing of modules.

Additionally, the system supports user-defined extensions, enabling the integration of custom MCP servers or external servers tailored to specific agents or tasks.

5 Human-Agent Interaction

Our system incorporates human feedback at key decision points to enhance survey quality and ensure alignment with user objectives. The interaction mechanism unfolds in structured phases that iteratively capture user input and refine system outputs.

5.1 Consensus Achievement

Topic Consensus Phase begins with user-defined topics and goals, followed by LLM-generated topic analysis and expansion. Through multi-turn dialogue, the system clarifies the research scope, identifies critical perspectives, and develops search strategies. This phase continues until a consensus is reached between the user and the AI on the survey's focus and coverage.

Legend: ✓ = Full Support; ∼ = Limited Support; × = No Support

System	User Interaction	Modular Design	MCP Support	Custom Tools	Survey Focus	Open Source
Commercial Platforms						
Perplexity DR (AI, 2024)	✓	×	×	×	×	×
OpenAI DR (OpenAI, 2024)	✓	×	×	×	×	×
Gemini DR (Citron, 2024)	✓	×	×	×	×	×
Manus AI (Monica (Butterfly Effect AI), 2025)	✓	×	×	×	×	×
Deep Research Systems						
WebGPT (Nakano et al., 2021)	✓	∼	×	×	×	×
GPT-Researcher (Elovic, 2024)	∼	✓	✓	✓	×	✓
ResearchAgent (Huang et al., 2024)	✓	✓	×	∼	×	∼
Self-RAG (Asai et al., 2023)	×	✓	×	×	×	✓
CoSearchAgent (Chen et al., 2024)	∼	∼	×	×	×	×
OpenResearcher (Liu et al., 2024)	×	∼	×	×	×	∼
Search-o1 (Zhang et al., 2025a)	×	∼	×	×	×	✓
Agent-R1 (Chen et al., 2025)	×	∼	×	×	×	✓
Multi-Agent Frameworks						
CrewAI (Moura et al., 2024)	∼	✓	✓	✓	×	✓
AutoAgent (Link-AGI, 2024)	×	✓	×	∼	×	✓
Alita (Qiu et al., 2025b)	×	✓	✓	✓	×	✓
Survey Generation Systems						
AutoSurvey (Wang et al., 2024)	×	×	×	×	✓	✓
InteractiveSurvey (Wen et al., 2025)	✓	∼	×	×	✓	✓
SurveyX (Liang et al., 2025)	∼	∼	×	×	✓	✓
LLM×MapReduce-V3 (Ours)	✓	✓	✓	✓	✓	✓

Table 1: Comparison of Survey Generation and Deep Research Systems

5.2 Feedback Integration

Outline Refinement Phase presents the generated survey skeleton for user review and modification. Users may request structural adjustments, section reordering, or changes in content emphasis. The system processes this feedback via the skeleton server, ensuring coherence while accommodating user preferences.

Quality Assurance Integration enables users to assess intermediate outputs at each stage, providing feedback that influences subsequent module executions. This ensures the final survey reflects both user expertise and the system's ability to synthesize comprehensive literature.

6 Comparison and Evaluation

As shown in Table 1, our LLM×MapReduce-V3 system is uniquely equipped to meet the comprehensive demands of academic survey generation. While existing solutions excel in specific areas, commercial platforms in user interaction, research systems in modularity, and frameworks in MCP integration, none provide an all-encompassing solution. Our system is the first to holistically integrate comprehensive user interaction, a modular design, MCP standardization, custom tool integration, and survey-specific optimizations.

The evaluation results in Table 2 indicate that

System	Skeleton	Length	Quality
Gemini DR	54.54%	9.09%	42.86%
Manus AI	27.27%	9.09%	0.00%
Our work	18.18%	81.81%	57.14%

Table 2: Human evaluation results. We recruited five human expert reviewers to evaluate articals generated by Gemini DeepResearch, Manus AI, and our system across eleven topics. Reviewers cast their votes based in three criteria: skeleton, length, and quality.

Manus AI just generates preliminary outlines and contents lacks depth. Gemini DeepResearch provides superior depth, structural coherence, and fluency. Compared to other systems, ours provides broader coverage, particularly in literature reviews, and produces significantly longer contexts.

7 Conclusion

LLM×MapReduce-V3 introduces a modular, MCP-based architecture that enables automated research assistance. By supporting open integration of customizable agents and servers, it overcomes the rigidity of traditional closed-agent systems. Through standardized interfaces, our system enables flexible composition of community-developed components to meet diverse research needs. The proposed system also shows strong potential for broader application in knowledge-

intensive tasks. With human-in-the-loop design, it achieves better alignment with human objectives. Despite existing challenges, such as fixed workflows and complex data exchange, we advocate for open, adaptable agent ecosystems that evolve with advancing tools and demands.

Ethics Statement

This work focuses on the development of an open-source, modular agent system for academic survey generation. Our system is intended to support and augment human researchers, not to replace them. We emphasize human-in-the-loop design to ensure transparency, user control, and alignment with scholarly standards. No sensitive personal data or copyrighted materials were used in model training or evaluation. All evaluation was conducted with the consent of expert annotators. We acknowledge the potential risks of misuse in automating academic writing and encourage responsible deployment with clear disclosure of AI assistance.

Acknowledgement

This work is supported by the AI9Stars community. We also thank the anonymous reviewers for their insightful suggestions.

References

Perplexity AI. 2024. Perplexity deep research.

Anthropic. 2024. Introducing the model context protocol.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Yihang Chen, Weizhi Zhang, Yangning Li, and 1 others. 2024. Cosearchagent: A lightweight collaborative search agent with large language models. *arXiv preprint arXiv:2402.06360*.

Yihang Chen, Weizhi Zhang, Yangning Li, and 1 others. 2025. Agent-r1: Training powerful llm agents with end-to-end reinforcement learning. *arXiv preprint arXiv:2503.07891*.

Dave Citron. 2024. Try deep research and gemini 2.0 flash experimental. Google Blog.

Assaf Elovic. 2024. Gpt-researcher: Autonomous agent designed for comprehensive online research.

Jinhao Huang, Qingyun Gu, Liangming Tran, Hao Chen, Yuning Li, Haotian Ren, Zirui Yao, Yitao Wang, and Diyi Yang. 2024. Researchagent: Iterative research idea generation over scientific literature with large language models. *arXiv preprint arXiv:2404.07738*.

Yushi Li, Yiming Zhang, Jing Chen, Zhen Wang, Hanming Liu, and Yiming Zhang. 2024. Hellobench: Evaluating long text generation capabilities of large language models. *arXiv preprint arXiv:2409.16191*.

Jinhao Liang, Yihang Chen, Haozheng Zhang, and 1 others. 2025. Surveyx: An end-to-end solution for automated survey generation. *arXiv preprint arXiv:2501.12345*.

Link-AGI. 2024. Autoagent: A fully-automated and zero-code framework for llm agents.

Yuxuan Liu, Haozheng Chen, Weizhi Zhang, and 1 others. 2024. Openresearcher: Unleashing ai for accelerated scientific research. *arXiv preprint arXiv:2408.06941*.

Monica (Butterfly Effect AI). 2025. Manus (ai agent). Autonomous artificial intelligence agent.

João Moura and 1 others. 2024. Crewai: Framework for orchestrating role-playing, autonomous ai agents. `https://github.com/crewAIInc/crewAI`.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2024. Deep research with chatgpt.

Yuxia Qin, Zhen Wang, Yongqi Chen, and Hanming Liu. 2024. Factuality of large language models: A survey. *arXiv preprint arXiv:2411.15993*.

Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, and 1 others. 2025a. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *arXiv preprint arXiv:2506.14728*.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, and 1 others. 2025b. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*.

Harun Raseed. 2025. The model context protocol (mcp): A new standard for multi-agent intelligence in ai systems. *Medium*.

Unleash. 2025. Mcp for ai agents: Enabling modular, scalable agentic systems.

Haoyu Wang, Yujia Fu, Zhu Zhang, Shuo Wang, Zirui Ren, Xiaorong Wang, Zhili Li, Chaoqun He, Bo An, Zhiyuan Liu, and Maosong Sun. 2025. Llm×mapreduce-v2: Entropy-driven convolutional test-time scaling for generating long-form articles from extremely long resources. *Preprint*, arXiv:2504.05732.

Yidong Wang, Qi Guo, Wenjin Yao, Hongbo Zhang, Xin Zhang, Zhen Wu, Meishan Zhang, Xinyu Dai, Min Zhang, Qingsong Wen, and 1 others. 2024. Autosurvey: Large language models can automatically write surveys. *arXiv preprint arXiv:2406.10252*.

Zhiyuan Wen, Jiannong Cao, Zian Wang, Beichen Guo, Ruosong Yang, and Shuaiqi Liu. 2025. Interactivesurvey: An llm-based personalized and interactive survey paper generation system. *arXiv preprint arXiv:2504.08762*.

Weizhi Zhang, Yangning Li, Yihang Chen, and 1 others. 2025a. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2502.09049*.

Yiming Zhang, Yongqi Chen, Zhen Wang, and Hanming Liu. 2025b. Mastering ultra-long text generation via reinforcement learning. *arXiv preprint arXiv:2506.18841*.

GraDeT-HTR: A Resource-Efficient Bengali Handwritten Text Recognition System utilizing Grapheme-based Tokenizer and Decoder-only Transformer

Md. Mahmudul Hasan*, **Ahmed Nesar Tahsin Choudhury***,
Mahmudul Hasan, **Md. Mosaddek Khan**
Computer Science and Engineering, University of Dhaka
{mdmahmudul-2020215620, ahmednesartahsin-2020115612,
mahmudul-2019917803}@cs.du.ac.bd, mosaddek@du.ac.bd

Abstract

Despite Bengali being the sixth most spoken language in the world, handwritten text recognition (HTR) systems for Bengali remain severely underdeveloped. The complexity of Bengali script—featuring conjuncts, diacritics, and highly variable handwriting styles—combined with a scarcity of annotated datasets makes this task particularly challenging. We present **GraDeT-HTR**[1], a resource-efficient Bengali handwritten text recognition system based on a **Gra**pheme-aware **De**coder-only **T**ransformer architecture. To address the unique challenges of Bengali script, we augment the performance of a decoder-only transformer by integrating a grapheme-based tokenizer and demonstrate[2] that it significantly improves recognition accuracy compared to conventional subword tokenizers. Our model is pretrained on large-scale synthetic data and fine-tuned on real human-annotated samples, achieving state-of-the-art performance on multiple benchmark datasets.

1 Introduction

Optical Character Recognition (OCR) involves transforming printed or handwritten images into machine-encoded text, supporting tasks such as digitizing pages from books, scene photographs, receipts, or notes. Within OCR, Handwritten Text Recognition (HTR) is focused specifically on transcribing handwritten content. HTR remains a challenging task due to the vast diversity in individual writing styles, inconsistent lighting, varying stroke width, cursive scripts, and artifacts such as smudges or noise.

Despite significant progress in handwritten text recognition (HTR) for languages such as English and Chinese, Bengali remains notably underserved. This gap is especially critical given the widespread use of handwritten Bengali in education, administration, and historical records. Developing effective HTR systems for Bengali presents unique challenges: the script contains visually complex ligatures, numerous diacritics, and allographic variations, all of which are compounded by highly diverse individual handwriting styles. These linguistic and visual complexities, combined with a scarcity of large, high-quality annotated datasets, make Bengali HTR a particularly demanding low-resource OCR task.

Existing approaches to text recognition predominantly employ encoder-decoder architectures, wherein the encoder is responsible for extracting visual features from images while the decoder generates the corresponding transcriptions. Early methods (Shi et al., 2017) commonly utilized CNNs as encoders and RNNs, with CTC (Graves et al., 2006), as decoders. With the introduction of transformers (Vaswani et al., 2017), attention-based mechanisms have been utilized to gain high performance in OCR tasks (Wang et al., 2019; Sheng et al., 2019; Zhang et al., 2020; Li et al., 2023; Fujitake, 2024).

In the context of Bengali, existing research on text recognition is relatively limited. Recent studies have explored architectures based on Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). However, the majority of these works (Azad et al., 2020; Hossain et al., 2021; Safir et al., 2021; Chaudhury et al., 2022) evaluate their methods on specific datasets, resulting in limited generalizability of their results. Some recent efforts (Hossain et al., 2022) have considered evaluation on unseen datasets, thereby addressing issues of generalization to a certain extent. Nonetheless, approaches based on transformer architectures have not yet been explored for HTR tasks. For text detection, models such as BN-DRISHTI (Jubaer et al., 2023) attain satisfactory results, so the focus has largely shifted to the recognition stage. In paral-

*Equal contribution. Author order is randomly determined.
[1] https://cognistorm.ai/hcr
[2] https://github.com/mahmudulyeamim/GraDeT-HTR

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 696–706

lel, recent work on tokenizer design (Basher et al., 2023) has demonstrated that grapheme-based tokenization can improve recognition performance by addressing Bengali's intricate ligatures, conjuncts, numerous diacritics, and allographic variations.

Motivated by the strong performance of decoder-only transformer models in English and Chinese OCR tasks (Fujitake, 2024), as well as the potential significance of tokenizer selection, we propose a Bengali Handwritten Text Recognition system that leverages a decoder-only transformer architecture in conjunction with a grapheme-based tokenizer. The core contributions of this paper are summarized below:

- We develop an end-to-end Bengali HTR pipeline that integrates both text detection and recognition for full-page images.
- We study the effects of replacing the default tokenizer in a transformer-based model with a grapheme-based tokenizer tailored for Bengali—a direction that, to the best of our knowledge, has not been previously explored—and empirically show improvements in recognition performance.
- We demonstrate that strong HTR performance can be achieved without relying on large-scale pretrained language models, offering a practical path for developing this demanding application in low-resource settings.

2 Related Work

This section reviews prior research in three key areas relevant to our work: general optical character recognition (OCR) and handwritten text recognition (HTR), Bengali handwritten text recognition, and the impact of tokenization strategies in OCR for complex scripts.

Optical Character Recognition. Text recognition has evolved from traditional connectionist temporal classification (CTC)−based models to more sophisticated sequence-to-sequence (seq2seq) approaches. Early CTC frameworks, such as CRNN (Shi et al., 2017), combined CNN-based visual feature extraction with recurrent networks for temporal modeling. Subsequent methods expanded on this by incorporating attention modules, alternative visual encoders, and advanced normalization techniques to address irregular text shapes and challenging image conditions (Gao et al., 2019; Shi et al., 2018).

The introduction of transformer architectures (Vaswani et al., 2017) led to significant advances in seq2seq modeling. Transformers, with their self-attention mechanisms and positional encodings, handle variations in character scale, spatial arrangements, and bidirectional decoding more effectively (Zhang et al., 2020; Lee et al., 2020).

A more recent direction involves integrating language models (LMs) into recognition pipelines to enhance contextual understanding. TrOCR (Li et al., 2023), for example, leverages pretrained LMs as decoders using masked language modeling (MLM) objectives, while models like MaskOCR (Lyu et al., 2022) and ABINet (Fang et al., 2021) explore more sophisticated strategies for linguistic pre-training. DTrOCR (Fujitake, 2024) advances this line of work by omitting the encoder entirely and relying on an autoregressively pretrained decoder (GPT-2) (Radford et al., 2019).

In the domain of handwritten text recognition (HTR), these trends are paralleled by augmenting traditional RNN+CTC architectures with attention and language modeling components (Memon et al., 2020; Michael et al., 2019).

Bengali Handwritten Text Recognition. Most existing research on Bengali handwritten text recognition has focused on constrained tasks, such as recognizing isolated characters or digits (Sufian et al., 2022). A few works have extended this to word-level recognition using CRNN-based models trained with the CTC loss (Hossain et al., 2022; Basher et al., 2023). However, despite the growing prominence of transformer-based architectures in other languages, such methods remain largely unexplored for Bengali HTR.

Impact of Tokenization in OCR. Bengali script comprises approximately 13,000 graphemes (Alam et al., 2021)—substantially more than English, which has around 250. This makes the consideration of graphemes in tokenization a particularly important design decision in Bengali OCR. Basher et al. (2023) have shown empirically that grapheme-level tokenization yields improved performance in HTR compared to conventional character-level tokenization for Bengali.

3 Methodology

The system pipeline consists of two primary components: a text detection module, which segments full-page images into individual word images, and a text recognition module, which adopts a decoder-

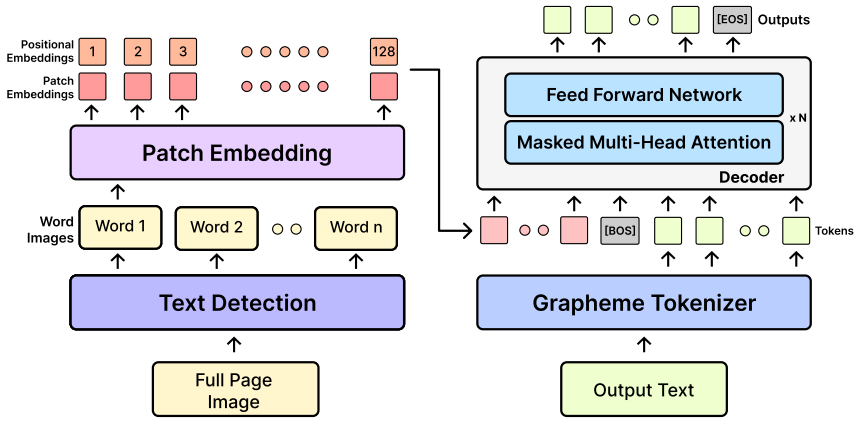


Figure 1: Architecture of the system pipeline. The pipeline consists of a text detection module and a text recognition module, which includes a patch embedding layer, a decoder-only transformer, and a grapheme-based tokenizer.

only architecture, beginning with a patch embedding layer followed by a decoder-only transformer. To enhance recognition accuracy for Bengali script, a grapheme-based tokenizer is integrated into the decoder. The overall architecture of the system is illustrated in Figure 1.

3.1 Text Detection

The task of recognizing handwritten text from full-page images typically requires large annotated datasets, which are often unavailable for Bengali. Consequently, most HTR methods operate at the line or word level. Given the constraints of our data setting, we adopt a word-level recognition approach.

The system begins by applying a text detection module to segment the full-page image into individual word images. For this, we employ BN-DRISHTI (Jubaer et al., 2023), a publicly available text detection model based on the YOLO framework (Redmon et al., 2016).

BN-DRISHTI follows a two-step line segmentation process. In the first step, line regions are identified directly from the full-page image. However, due to the curvilinear nature of Bengali handwriting, these preliminary detections often include spurious boundaries or lines above or below the actual lines. To correct for this, Hough line and Affine transformations are applied to normalize skew and straighten the text baselines. Preprocessing operations such as binarization and morphological filtering are then employed to suppress noise and refine the line boundaries. In the second step, the refined line images are re-segmented, yielding more accurate line-level outputs. Finally, individual word images are segmented from the refined text lines and forwarded to the recognition module.[3]

3.2 Text Recognition

We adopt a decoder-only text recognition module, in contrast to the encoder-decoder architecture commonly used in OCR. This module builds upon the open-source implementation of DTrOCR[4] (Fujitake, 2024), which we adapt using a grapheme-based tokenizer tailored for Bengali handwriting.

3.2.1 Patch Embedding

Transformers cannot directly process raw image inputs. Following the approach of ViT (Dosovitskiy et al., 2021), each input image of size $(3, H_0, W_0)$ is first resized to $(3, H, W)$ and then divided into non-overlapping patches of size (p_h, p_w), resulting in $N = (H/p_h) \times (W/p_w)$ patches. Each patch is linearly projected into a D-dimensional embedding. Positional encodings $P \in \mathbb{R}^{N \times D}$ are then added to retain spatial information. The resulting sequence is passed to the decoder module for autoregressive token prediction.

3.2.2 Decoder

The decoder module generates text directly from the embedded image sequence, bypassing the encoder used in traditional encoder–decoder architectures to extract visual features from raw images. This design eliminates the need for cross-attention and reduces both model parameters and computational cost. Our decoder module uses the architecture of Generative Pretrained Transformers (GPT) (Radford et al., 2018, 2019) to recognize

[3] Since we employ the exact model without any modification, we refer the interested readers to the BN-DRISHTI paper (Jubaer et al., 2023) for detailed methodology and evaluation results of the detection model.

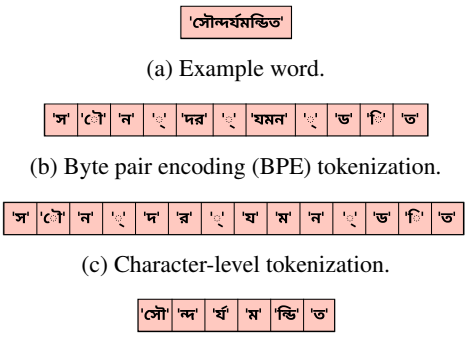
[4] https://github.com/arvindrajan92/DTrOCR

handwritten text. Specifically, we adopt a decoder-only architecture that consists solely of standard Transformer decoder blocks (Vaswani et al., 2017). In contrast to models such as TrOCR (Li et al., 2023) and DTrOCR (Fujitake, 2024), we do not rely on pretrained Bengali language models. Instead, our decoder is trained from scratch, making it suitable for low-resource settings.

After the embedded image sequence, a `[BOS]` token is appended and the resulting sequence is passed into the decoder. Tokens are then generated autoregressively until a `[EOS]` token is produced. The final output from the decoder is projected onto the vocabulary space and passed through a softmax layer for token prediction.

3.2.3 Tokenizer

A *grapheme* is the smallest visually distinguishable unit in a writing system. Transformer-based OCR models (Li et al., 2023; Fujitake, 2024) typically rely on default tokenizers such as Byte Pair Encoding (BPE) (Sennrich et al., 2016), which segments text into subwords or characters based on frequency statistics (see Figure 2b). These subword units often span multiple characters and are visually unstable in handwritten text: even slight variations in a single character can distort the entire token.



(a) Example word.

(b) Byte pair encoding (BPE) tokenization.

(c) Character-level tokenization.

(d) Grapheme-based tokenization.

Figure 2: Comparison of tokenization strategies for Bengali HTR using the example word shown in (a).

This challenge is amplified in Bengali, where writing styles vary widely and many graphemes are rendered with ligatures or conjuncts. Subword or character-level tokenization becomes unreliable without access to large, diverse training data. Furthermore, the visual order of Bengali text often diverges from its logical sequence, complicating alignment (see Figure 2c). Prior work (Basher et al., 2023) has shown that character-level tokenization fails to capture these structures effectively.

To address these challenges, we adopt a grapheme-based tokenizer that enforces a one-to-one mapping between visually coherent units and model output tokens (see Figure 2d). Specifically, we integrate BnGraphemizer (Basher et al., 2023), a trie-based grapheme tokenizer designed for Bengali, into our decoder-only transformer pipeline.

3.3 Training

The training process consists of two stages: pre-training on large-scale synthetic data and fine-tuning on real-world, human-annotated samples.

First, we employ a synthetic data generation tool[5] adapted from the VRD framework (Lauar and Laurent, 2024) to construct diverse datasets (as detailed in Section 4.1) for pre-training. Pre-training itself is conducted in two phases, first on line-level images and then on word-level images. In the initial phase, the training on synthetic images containing single lines of handwritten text helps the model learn high-level structure. In the latter phase, the model is further trained on word-level images. Line-level training is particularly useful in early stages, as text detection models may group multiple words into a single detection region, especially in cursive or compact writing styles.

Second, we fine-tune the pretrained model on human-annotated Bengali handwriting datasets (as detailed in Section 4.1), as previous works (Baek et al., 2021; Bautista and Atienza, 2022) have shown that synthetic-only dataset is generally insufficient for generalization to real-world inputs—an issue heightened in Bengali's complex script.

4 Experiments

In this section, we detail the experimental setup, describe the evaluation metrics, and present the results of our study.

4.1 Experimental Setup

We outline the experimental setup here, covering the details of training data and evaluation benchmarks, training configurations, and implementation details used in both the pre-training and fine-tuning stages.

4.1.1 Training Data and Benchmarks

For pre-training, we generate synthetic datasets comprising 4.5 million line-level samples and 7

[5] https://github.com/tahsinchoudhury/GlyphScribe

Model	Tokenizer	Pretrained LLM	#Parameters	BN-HTRd		Bongabdo	
				CER(%)	WER(%)	CER(%)	WER(%)
Line_BPE_Pretrained	BPE	✓	162M	38.34	51.83	65.04	77.65
Line_BPE_Random	BPE	✗	162M	26.17	39.22	46.91	63.85
Line_BnG_Random	BnGraphemizer	✗	87M	29.58	42.89	50.69	68.02
Word_BPE_Pretrained	BPE	✓	162M	7.57	16.30	11.61	26.99
Word_BPE_Random	BPE	✗	162M	6.68	15.05	10.09	25.39
Word_BnG_Random	BnGraphemizer	✗	87M	**6.19**	**14.20**	**8.68**	**23.56**

Table 1: Model configuration details and recognition performance (CER, WER) on the BN-HTRd and Bongabdo datasets. Model names follow the format *[Input granularity]_[Tokenizer]_[Decoder initialization]*. Here, *Line* level models take entire handwritten line images as input and predict the full line text, while *Word* level models take cropped word images as input and predict the word text. *BPE* and *BnG* denote the Byte-Pair Encoding and BnGraphemizer tokenizers, respectively. *Pretrained*/*Random* indicates whether the decoder was initialized from a pretrained Bengali language model or with random weights. Among all variants, **Word_BnG_Random** (our proposed word-level model with grapheme tokenizer, trained from scratch) achieves the lowest CER and WER.

million word-level samples. The text content is sourced from Bengali Wikipedia, the Bangla-NMT dataset (Hasan et al., 2020), and a publicly available Bengali dictionary (Kamal, 2018). These sources provide a wide range of vocabulary and writing styles. To enhance realism, our synthetic data generator introduces artifacts such as handwritten-style fonts, wavy or bent lines, Gaussian blur, and partial character fragments that mimic cropping effects. Representative examples are illustrated in Appendix A.

To fine-tune and evaluate our model, we use six human-annotated datasets: BN-HTRd (Rahman et al., 2023; Jubaer et al., 2023), BanglaWriting (Mridha et al., 2021), BanglaLekha-Isolated (Biswas et al., 2017), IIIT-Indic (Jindal et al., 2021), ICDAR 2024 (IIIT Hyderabad and CVIT Lab, 2024), and Bongabdo (Islam et al., 2023).

Dataset	# Words
BN-HTRd	104,854
BanglaWriting	21,234
BanglaLekha-Isolated (Digits only)	19,748
IIIT-Indic	113,075
ICDAR 2024	79,663
Total	**338,574**

Table 2: Summary of human-labeled datasets used to fine-tune the word-level models.

We consider two model variants: line-level models, which take entire handwritten line images as input and predict the full line text, and word-level models, which take cropped word images as input and predict the text for that word. For fine-tuning, the line models use 13,912 real handwritten line images from the BN-HTRd dataset (Rahman et al., 2023). In contrast, the word models are fine-tuned on a larger collection of approximately 340,000 (see Table 2 for details) real handwritten word images aggregated from BN-HTRd (Rahman et al., 2023), BanglaWriting, BanglaLekha-Isolated, IIIT-Indic, and ICDAR 2024. For evaluation, we benchmark on 805 full-page images from the "automatic annotation" split of the extended BN-HTRd dataset (Jubaer et al., 2023) and 111 images from the Bongabdo dataset.

4.1.2 Implementation Details

We implement our decoder using the GPT-2 architecture from HuggingFace (Wolf et al., 2020), consisting of 12 transformer layers with a hidden size of 768 and 12 self-attention heads. The maximum sequence length is set to 256 tokens. To better suit the properties of Bengali handwritten text, we replace the default byte-pair encoding tokenizer with the BnGraphemizer tokenizer.

Each input image is resized to a fixed resolution of 32×128 pixels and divided into non-overlapping patches of size 4×8, yielding 128 image tokens per word. The model is pretrained using a batch size of 32 with the Adam optimizer and a learning rate of 1×10^{-4}. During fine-tuning, the same batch size and optimizer are used, but the learning rate is reduced to 5×10^{-6}.

All training and inference experiments are conducted using PyTorch on a single NVIDIA GeForce RTX 3080 GPU (see Appendix C for more details).

4.2 Evaluation Metrics

We evaluate our handwriting recognition system using two widely adopted metrics for handwritten text

System	BN-HTRd		Bongabdo	
	CER (%)	WER (%)	CER (%)	WER (%)
Gemini 2.5 Flash[6]	19.39	32.49	37.42	56.39
Imagetotext.info[7]	20.90	37.10	14.36	31.14
Ours	**6.19**	**14.20**	**8.68**	**23.56**

Table 3: Comparison of cloud-based OCR systems and our proposed model on Bengali HTR.

Filtered	BN-HTRd		Bongabdo	
	CER (%)	WER (%)	CER (%)	WER (%)
No	6.19	14.20	8.68	23.56
Yes	**4.58**	**12.59**	**8.02**	**22.95**

Table 4: Comparison of recognition performance with and without filtering images where the text detection module failed to segment words accurately.

recognition (HTR): Character Error Rate (CER) and Word Error Rate (WER). CER measures the proportion of character-level errors—substitutions, deletions, and insertions—required to transform the predicted sequence into the ground truth, while WER computes the same at the word level. Formal definitions of both metrics are provided in Appendix B.

4.3 Experimental Results

We now discuss the results of our model across ablations, comparisons with existing OCR systems and prior HTR models, an analysis of its sensitivity to text detection quality and inference speed.

Model Configuration Analysis. We compare multiple model configurations to evaluate the effects of input granularity, decoder initialization, and tokenization. Specifically, we study line-level vs. word-level recognition, pretrained Bengali language model vs. randomly initialized decoders, and subword-level Byte Pair Encoding (BPE) vs. the grapheme-level BnGraphemizer tokenizer. While BPE is used with the pretrained LLM, BnGraphemizer is evaluated only with randomly initialized decoders, as no pretrained Bengali LLM exists for this tokenizer.

The line-level models are pretrained on synthetic line images, while word-level models undergo additional pretraining on synthetic word images. Line-level models are fine-tuned on handwritten line images, while word-level models are fine-tuned on a larger set of handwritten word images, as discussed in Section 4.1.

As shown in Table 1, word-level models outperform line-level ones, which can be attributed to the abundance of word-level training data and the more nuanced nature of line-level data. Our proposed model—Word_BnG_Random—achieves the lowest CER and WER without using a pretrained Bengali LLM. The grapheme-based tokenizer further improves accuracy while reducing model size from 162M to 87M, demonstrating both effectiveness and parameter efficiency.

Comparison with Existing OCR Systems. We compare our proposed model with two existing OCR systems: Gemini 2.5 Flash and Imagetotext.info. As shown in Table 3, our model achieves the lowest CER and WER, outperforming both systems by a large margin. Gemini 2.5 Flash is selected as a representative vision-language model (see Appendix D for prompt details) due to its strong performance on Bengali handwritten text, substantially surpassing models such as ChatGPT and Claude on this task. We exclude other Google Vision API models, as Gemini consistently outperforms them. Imagetotext.info is included for its support for Bengali handwriting and its accessible API, which enables automated benchmarking. Other OCR services were excluded based on lack of Bengali handwriting support, absence of APIs, or poor performance in preliminary evaluations.

Dependency on Text Detection Model. To assess the sensitivity of our system to the quality of word segmentation, we manually inspected the BN-HTRd and Bongabdo datasets and identified a small number of samples where the text detection model failed to accurately segment words. After filtering out these cases, our text recognition model demonstrated improved performance, as shown in Table 4. These findings suggest that the end-to-end pipeline is affected by errors in text detection, and that further gains in recognition accuracy may be achieved by incorporating a more precise word segmentation module.

Comparison with Prior Models. Table 5 presents cross-dataset evaluation results comparing our proposed model with prior Bengali HTR approaches, including LILA-BOTI (Hossain et al., 2022), CRNN-CT, and CRNN-BnG (Basher et al., 2023). We adopt cross-dataset evaluation to ensure consistency with prior work, which report performance under the same setting. This approach

[6]Although Gemini 2.5 Pro yields slightly better results, we use Flash due to strict API limits in the Pro free tier.

[7]https://www.imagetotext.info

Datasets	Model	CER (%)	WER (%)
BW (train) BN-HTRd (test)	LILA-BOTI*	25.92	51.62
	CRNN-CT*	19.94	44.10
	CRNN-BnG*	30.53	48.41
	Ours	**18.04**	**33.20**
BN-HTRd (train) BW (test)	LILA-BOTI*	15.54	36.78
	CRNN-CT*	**11.12**	29.43
	CRNN-BnG*	15.80	29.18
	Ours	12.66	**24.79**

Table 5: Cross-dataset evaluation on BW (BanglaWriting) and BN-HTRd datasets. The asterisk (*) indicates results reported from (Basher et al., 2023).

provides a fair basis for comparison and reflects the models' ability to generalize across different handwriting corpora. When fine-tuned on BW (BanglaWriting) (Mridha et al., 2021) and evaluated on BN-HTRd (Rahman et al., 2023), our model achieves the lowest CER and WER. In the reverse setting, it obtains the best WER, while the CER is slightly higher.

Inference Speed. We conducted experiments on the Bongabdo dataset to compare the inference speeds of different models, as reported in Table 6. Each experiment was repeated five times to ensure statistical robustness, and the average inference speed was reported as the final result. This set of experiments was performed on the aforementioned devices, which were also used for model training and other experiments. As expected, our proposed model with 87M parameters achieves a higher inference speed than the 162M parameter variant. This demonstrates the practicality of our model for deployment in low-resource computational settings while delivering faster results to end users.

Model	#Params	#Words	Time	Speed
Word_BPE_Random	162M	17,217	2074.32s	8.30 words/s
Word_BnG_Random	87M	17,217	**1729.59s**	**9.95** words/s

Table 6: Inference time on the Bongabdo dataset.

5 System Description

We deploy our Bengali handwriting recognition system as a Flask-based REST API, enabling external access through a lightweight and modular interface. The backend supports asynchronous task execution using Celery, with Redis serving as the message broker. To ensure reproducibility and ease of deployment, the entire system is containerized with Docker, and all components are orchestrated via Docker Compose. The inference engine runs on an NVIDIA GeForce RTX 3080 GPU.

The system supports a variety of input formats, including individual image files (PNG, JPG, JPEG) as well as multi-page PDF documents. Once uploaded, documents are processed through the detection and recognition pipeline described in Sections 3.1 and 3.2, respectively, where full-page handwritten images are segmented into word-level regions and transcribed using a decoder-only transformer model. For each image or PDF page, the recognized text is displayed in an editable text area, allowing users to make corrections as needed. In the case of PDFs, the system provides per-page navigation to support multi-page review. Users can export the extracted text in either plain text (`.txt`) or Microsoft Word (`.docx`) format.

The web interface features a document upload panel on the left and a text output display area on the right. Snapshots of the user interface, along with representative use cases, are provided in Appendix E.

6 Conclusions

We present a resource-efficient Bengali handwritten text recognition system that operates at the word level without relying on a pretrained large language model as the decoder. Our approach leverages a decoder-only transformer combined with a grapheme-based tokenizer tailored for Bangla script. Experimental results demonstrate that our method outperforms existing OCR systems and prior Bengali HTR models, achieving state-of-the-art accuracy despite being trained on limited annotated data.

7 Limitations and Future Work

Our system has two main limitations: its training data mainly consists of handwritten text on white backgrounds, limiting adaptability to more varied real-world scenarios; and the scarcity of large-scale pretrained GPT-2 models for Bengali limits the breadth of the experiments. Future work can include expanding the synthetic dataset to cover diverse backgrounds and noise artifacts, pre-training auto-regressive language models on extensive Bengali text, experimenting with other auto-regressive language models (such as Llama), and refining text detection to reduce segmentation errors. To support future research in Bengali HTR, we publicly release our complete pipeline under the MIT license.

Ethical Considerations

Our system does not retain any user-submitted documents or images. All uploads are processed temporarily and permanently deleted after a short duration to ensure user privacy. The datasets used for training and evaluation are publicly available and intended for research purposes. No personal or sensitive content was included in the model development process.

We promote responsible use of the system in archival, educational, and administrative contexts. Its intended applications include the digitization of historical or administrative documents, support for literacy development, and automation of handwritten workflows in institutional settings. It is not designed for surveillance, profiling, or other harmful purposes.

Acknowledgements

The authors acknowledge the support provided by the Bangladesh Bureau of Educational Information and Statistics (BANBEIS) and the University of Dhaka during the course of this research.

References

S. Alam, T. Reasat, A. S. Sushmit, S. M. Siddique, F. Rahman, M. Hasan, and A. I. Humayun. 2021. A large multi-target dataset of common bengali handwritten graphemes. In *Document Analysis and Recognition – ICDAR 2021*, pages 383–398. Springer International Publishing.

M. A. Azad, H. S. Singha, and M. M. H. Nahid. 2020. Bangla handwritten character recognition using deep convolutional autoencoder neural network. In *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, pages 295–300.

Jeonghun Baek, Yusuke Matsui, and Kiyoharu Aizawa. 2021. What if we only use real datasets for scene text recognition? toward scene text recognition with fewer labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3113–3122.

Mohammad Jahid Ibna Basher, Mohammad Raghib Noor, Sadia Afroze, and Ikbal Ahmed. 2023. Bngraphemizer: A grapheme-based tokenizer for bengali handwritten text recognition. In *2023 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering*, Sree Chitra Thirunal College of Engineering, Thiruvananthapuram, Kerala, INDIA.

Darwin Bautista and Rowel Atienza. 2022. Scene text recognition with permuted autoregressive sequence models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 178–196.

Mithun Biswas, Rafiqul Islam, Gautam Kumar Shom, Md. Shopon, Nabeel Mohammed, Sifat Momen, and Anowarul Abedin. 2017. Banglalekha-isolated: A multi-purpose comprehensive dataset of handwritten bangla isolated characters. *Data in Brief*, 12:103–107.

A. Chaudhury, P. S. Mukherjee, S. Das, C. Biswas, and U. Bhattacharya. 2022. A deep ocr for degraded bangla documents. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 21(5).

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 *Proceedings of the International Conference on Learning Representations (ICLR)*.

Shancheng Fang, Hongtao Xie, Yuxin Wang, Zhendong Mao, and Yongdong Zhang. 2021. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7098–7107.

Masato Fujitake. 2024. Dtrocr: Decoder-only transformer for optical character recognition. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 8025–8035.

Y. Gao, Y. Chen, J. Wang, M. Tang, and H. Lu. 2019. Reading scene text with fully convolutional sequence modeling. *Neurocomputing*, 339:161–170.

A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural network. In *Proceedings of the 23rd international conference on Machine learning*.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M. Sohel Rahman, and Rifat Shahriyar. 2020. Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2612–2623, Online. Association for Computational Linguistics.

M. I. Hossain, M. Rakib, S. Mollah, F. Rahman, and N. Mohammed. 2022. Lila-boti: Leveraging isolated letter accumulations by ordering teacher insights for bangla handwriting recognition. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 1770–177.

M. T. Hossain, M. W. Hasan, and A. K. Das. 2021. Bangla handwritten word recognition system using convolutional neural network. In *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–8.

IIIT Hyderabad and CVIT Lab. 2024. Icdar 2024 handwritten document recognition dataset. https://ilocr.iiit.ac.in/icdar_2024_hwd/.

Md Majedul Islam, Avishek Das, Ibna Kowsar, AKM Shahariar Azad Rabby, Nazmul Hasan, and Fuad Rahman. 2023. Towards full-page offline bangla handwritten text recognition using image-to-sequence architecture. In *2023 IEEE International Conference on Big Data (Big Data)*, pages 1061–1067. IEEE.

Milan Jindal, Pratyush Kumar, and C.V. Jawahar. 2021. iiit-indic-hw-words: A dataset for indic handwritten text recognition. *arXiv preprint arXiv:2105.04020*.

Sheikh Mohammad Jubaer, Nazifa Tabassum, Md Ataur Rahman, and Mohammad Khairul Islam. 2023. Bn-drishti: Bangla document recognition through instance-level segmentation of handwritten text images. *arXiv preprint arXiv:2306.09351*.

M. Kamal. 2018. Minhaskamal/bengalidictionary: A large collection of bengali words. https://github.com/MinhasKamal/BengaliDictionary.

Filipe Lauar and Valentin Laurent. 2024. Spanish trocr: Leveraging transfer learning for language adaptation. *Preprint*, arXiv:2407.06950.

J. Lee, S. Park, J. Baek, S.J. Oh, S. Kim, and H. Lee. 2020. On recognizing texts of arbitrary shapes with 2d self-attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 546–547.

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2023. Trocr: Transformer-based optical character recognition with pre-trained models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13094–13102.

Hengyuan Lyu, Chengquan Zhang, Shanshan Liu, Meina Qiao, Yangliu Xu, Liang Wu, Kun Yao, Junyu Han, Errui Ding, and Jingdong Wang. 2022. Maskocr: Text recognition with masked encoder-decoder pretraining. *arXiv preprint arXiv:2206.00311*.

J. Memon, M. Sami, R. A. Khan, and M. Uddin. 2020. Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). *IEEE Access*, 8:142642–142668.

J. Michael, R. Labahn, T. Grüning, and J. Zöllner. 2019. Evaluating sequence-to-sequence models for handwritten text recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1286–1293. IEEE.

Md Forhad Mridha, Abu Quwsar Ohi, Md Ameer Ali, Md I Emon, and Md Mohsin Kabir. 2021. Banglawriting: A multi-purpose offline bangla handwriting dataset. *Data in Brief*, 34:106633.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI*. Technical report.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI. Technical report.

Md Ataur Rahman, Nazifa Tabassum, Manash Paul, Rajib Pal, and Mohammad Khairul Islam. 2023. Bn-htrd: A benchmark dataset for document level offline bangla handwritten text recognition (htr) and line segmentation. In *Computer Vision and Image Analysis for Industry 4.0*, pages 1–16. CRC Press.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

F. B. Safir, A. Q. Ohi, M. F. Mridha, M. M. Monowar, and M. A. Hamid. 2021. End-to-end optical character recognition for bengali handwritten words. In *2021 National Computing Colleges Conference (NCCC)*, pages 1–7.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

F. Sheng, Z. Chen, and B. Xu. 2019. Nrtr: A no-recurrence sequence-to-sequence model for scene text recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 781–786. IEEE.

B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai. 2018. Aster: An attentional scene text recognizer with flexible rectification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2035–2048.

Baoguang Shi, Xiang Bai, and Cong Yao. 2017. 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.

A. Sufian, A. Ghosh, A. Naskar, F. Sultana, J. Sil, and M. H. Rahman. 2022. Bdnet: Bengali handwritten numeral digit recognition based on densely connected convolutional neural networks. *Journal of King Saud University - Computer and Information Sciences*, 34(6, Part A):2610–2620.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

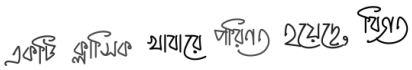
S. Wang, Y. Wang, X. Qin, Q. Zhao, and Z. Tang. 2019. Scene text recognition via gated cascade attention. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1018–1023. IEEE.

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, and 3 others. 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.

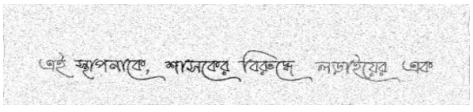
Jiaxing Zhang, Canjie Luo, Lianwen Jin, Tianwei Wang, Ziyan Li, and Weiying Zhou. 2020. Sahan: Scale-aware hierarchical attention network for scene text recognition. *Pattern Recognition Letters*, 136:205–211.

A Synthetic Images

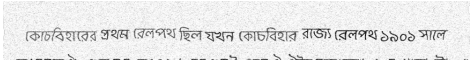
To make synthetic images resemble real handwritten samples, we introduce artifacts such as wavy and bent text, Gaussian blur, and partial character fragments simulating real-world cropping effects during data generation. Representative examples are illustrated in Figure 3.



(a) Bent line with a clear background.



(b) Line image with Gaussian blur applied.



(c) Wavy line with partial character fragments.

(d) Word with clear background.

Figure 3: Representative examples generated by the synthetic image generator.

B Definitions of the Evaluation Metrics

We provide the definitions of Character Error Rate (CER) and Word Error Rate (WER) below.

CER. CER measures the proportion of character-level errors—substitutions (S), deletions (D), and insertions (I)—required to transform the predicted text sequence (hypothesis) into the reference text (ground truth), normalized by the total number of characters in the reference (N). Formally,

$$\text{CER} = \frac{S + D + I}{N}$$

WER. WER measures the proportion of word-level errors—substitutions (S), deletions (D), and insertions (I)—needed to convert the predicted word sequence into the reference word sequence, divided by the total number of words in the reference (N). Formally,

$$\text{WER} = \frac{S + D + I}{N}$$

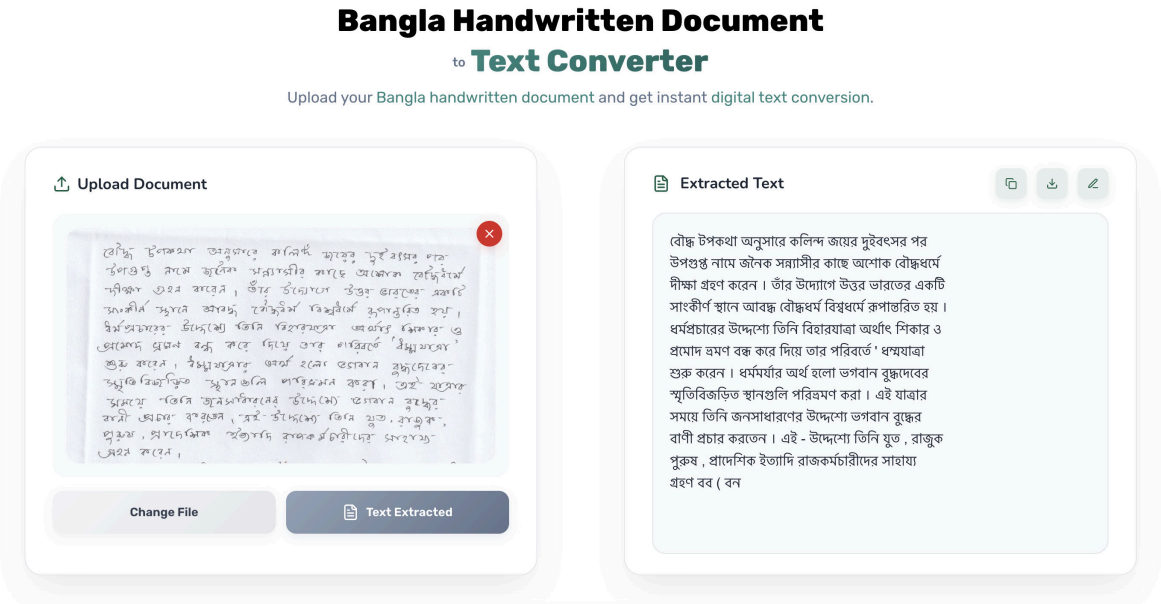
C Training Details

Our model was pretrained and fine-tuned on a single NVIDIA GeForce RTX 3080 GPU with a batch size of 32. The first-stage pretraining on 4.5M synthetic line-level images requires 13 hours over 2 epochs, while the second-stage pretraining on 7M synthetic word-level images takes 9 hours for a single epoch. The model is then fine-tuned on real handwritten datasets for 4 epochs, completing in less than 2 hours.

D LLM Evaluation

The prompt we use to transcribe an image using an external LLM is provided below.

> You are an optical character recognition (OCR) tool. Your task is to transcribe the handwritten Bengali text shown in the provided image directly and exactly, without making any corrections, translations, or modifications. Extract the text precisely as it appears, preserving original spelling, grammar, and formatting.
> Output only the text found in the image, with no additional comments, explanations, or formatting.



(a) Example of the interface after uploading an image.

(b) Example of the interface after uploading a PDF.

Figure 4: Screenshot of the Bangla Handwritten Document to Text Converter web interface, showing the document upload area on the left and the extracted text display area on the right.

E Illustrative Outputs from the System

Figure 4 depicts the system's user interface along with sample use-cases. A video demonstrating how to use the system is available on YouTube[8].

[8] https://youtu.be/ckgWBHQarxc

AutoIntent: AutoML for Text Classification

Alekseev Ilya[1,2,4], **Solomatin Roman**[1,3], **Rustamova Darina**[3], **Kuznetsov Denis**[1]

[1]Moscow Center for Advanced Studies, [2]Moscow State University, [3]ITMO University, [4]dresscode.ai.

Correspondence: ilya_alekseev_2016@list.ru

Abstract

AutoIntent is an automated machine learning tool for text classification tasks. Unlike existing solutions, AutoIntent offers end-to-end automation with embedding model selection, classifier optimization, and decision threshold tuning, all within a modular, sklearn-like interface. The framework is designed to support multi-label classification and out-of-scope detection. AutoIntent demonstrates superior performance compared to existing AutoML tools on standard intent classification datasets and enables users to balance effectiveness and resource consumption.

1 Introduction

Text classification remains a fundamental task in natural language processing, with applications ranging from intent detection in conversational systems (Weld et al., 2021) to content categorization and sentiment analysis (Taha et al., 2024). Modern NLP has been revolutionized by transformer-based embedding models (Vaswani et al., 2023; Devlin et al., 2019; Reimers and Gurevych, 2019), which provide rich contextual representations of text. However, effectively utilizing these models for classification tasks requires careful consideration of multiple components: the choice of pre-trained embedding model, the selection of appropriate classification algorithms, and the optimization of their hyperparameters.

Traditional machine learning approaches often require manual tuning of these components, which can be time-consuming and requires significant expertise. While automated machine learning (AutoML) frameworks (Baratchi et al., 2024) have emerged to automate this process, existing solutions for NLP tasks often lack comprehensive support for the full spectrum of hyperparameter optimization (for instance, choosing best embedding model) and tuning confidence thresholds for handling multi-label classification and out-of-scope (OOS) detection (Larson et al., 2019).

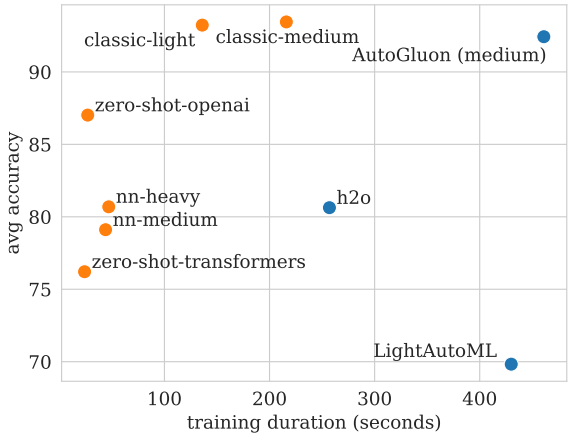


Figure 1: Average accuracy (banking77 (Casanueva et al., 2020), massive (FitzGerald et al., 2022), minds14 (Gerz et al., 2021), hwu64 (Liu et al., 2019)) and training duration (on minds14) of AutoIntent presets (**orange**) and baseline AutoML tools (**blue**).

This paper introduces AutoIntent[1], a novel AutoML framework specifically designed for intent classification tasks. The framework offers a sklearn-like interface (Pedregosa et al., 2018) for ease of use and supports even multi-label classification and OOS detection, bridging the gap between AutoML and conversation systems.

2 Background

Automated machine learning, by definition, is a tool for automating routines like data splitting for validation, hyperparameter tuning, model ensembling, and model selection. AutoML is highly relevant in scenarios where machine learning tasks need to be solved by non-experts and in conjunction with no-code ML, which is sometimes called «ML as a service» (Bisong, 2019; Barga et al., 2015; Liberty et al., 2020; LeDell and Poirier, 2020; Carney

[1]https://github.com/DeepPavlov/AutoIntent

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 707–716

	H2O	LightAutoML	AutoGluon	FEDOT	**AutoIntent (ours)**
Approach	TAML & Word2Vec	TAML & emb.	Encoder fine-tuning	TAML & TF-IDF	**Embeddings**
Scarce train data	×	Has small data modes	×	Adaptable	**Adapted for small datasets**
Custom search space	Flexible API	HP presets	✓	HP presets	**Presets & customizble configs**
Experiments tracking	H2O Flow integration	×	×	×	**W&B*, tensorboard*, codecarbon***.
Embedding prompting	×	×	×	×	✓
OOS detection	×	×	×	×	✓
Multi-label	×	✓	×	×	✓

Table 1: Comparison of NLP functionality in AutoML frameworks: H2O (LeDell and Poirier, 2020), LightAutoML (Vakhrushev et al., 2022), AutoGluon (Tang et al., 2024), FEDOT (Nikitin et al., 2021) and ours AutoIntent. HP stands for hyperparameters, TAML stands for tabular AutoML. *Weights and Biases (Biewald, 2020), Tensorboard (Abadi et al., 2015), CodeCarbon (Courty et al., 2024).

et al., 2020; Acito, 2023). The applications include sentiment analysis, robotics, healthcare, business analysis (Yuan et al., 2024; Salehin et al., 2024).

Tabular AutoML focuses on feature engineering, feature selection and model ensembling (Feurer et al., 2022; Vakhrushev et al., 2022; Erickson et al., 2020; LeDell and Poirier, 2020; Nikitin et al., 2021). Usually, they employ classical machine learning methods like GBMs (Chen and Guestrin, 2016; Prokhorenkova et al., 2018; Ke et al., 2017) and linear models, fast hyperparameter tuning methods with budget-aware strategies (Akiba et al., 2019), and ensembling strategies like stacking, blending and voting. Such frameworks sometimes can supersede exploratory data analysis and extensive research with just a running preset training recipe. It is not rare to see AutoML frameworks winning machine learning contests, but the open source solutions often are not transferable to production-ready systems as the resulting pipeline is an ensemble of a numerous amount of models without clear guides for deployment.

Neural architecture search can be viewed as an automation in the field of deep learning (Salehin et al., 2024). It emphasizes finding optimal computational graph using approaches like cell-based and hierarchical search spaces (Zoph et al., 2018; Real et al., 2019) or using scaling laws (Tan and Le, 2020). It cannot be treated as full-fledged AutoML, as it is designed to address only the model selection problem. Though, it can be a part of an AutoML pipeline (Jin et al., 2023).

AutoML tools in the NLP domain primarily stand out from other AutoML by native support of text inputs (Tang et al., 2024; Vakhrushev et al., 2022). This is especially important for use by non-experts, as it removes the requirement of manual tokenization and vectorization. Though, some tabular AutoML frameworks provide auxiliary tools for text feature extraction (LeDell and Poirier, 2020). The next peculiarity of text AutoML frameworks is their usage of transformer-based backbones, which makes sense, as this is the state-of-the-art in the field of NLP. Note that NLP AutoML primarily focuses on simple tasks like classification and regression, ignoring text generation and named entity recognition, for instance.

In AutoML frameworks, the model selection can be implemented in three ways. The first and most straightforward is to use hyperparameter tuning tools like Optuna (Akiba et al., 2019) and genetic algorithms (Feurer et al., 2022) with preset search spaces. Usually, these presets differ in how much time and computational resources they require to reach acceptable quality. The variety of presets is provided to cover all possible use cases and hardware settings. Another option is not to tune hyperparameters but use some generic hyperparameters that reach balance between the final quality and the generalization across different tasks. These hyperparameters can be obtained empirically (Tang et al., 2024). The compromise between freezing hyperparameters and tuning all of them is the meta-learning (Desai et al., 2022; Feurer et al., 2022; Wang, 2021; Tian et al., 2022; Huisman et al., 2021), where metamodel takes a dataset as input and predicts hyperparameters.

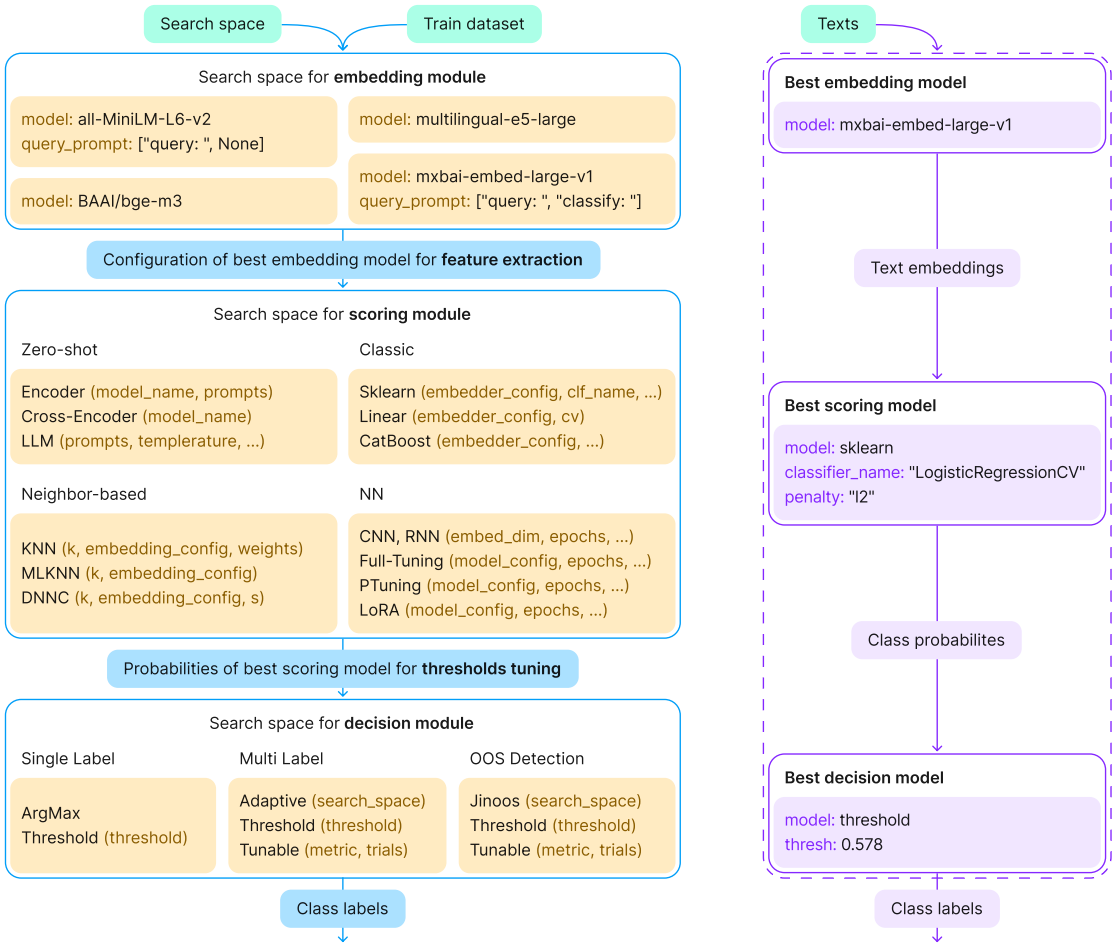


Figure 2: **(Left)** AutoIntent's three levels of hyperparameter optimization: at the module level, the embedding, scoring, and decision models are optimized sequentially; at the model level, each classification approach is tested against each other to select the best one; at the instance level, hyperparameters for each model is tuned individually with optuna samplers. **(Right)** Inference pipeline as a result of AutoIntent's hyperparameter optimization.

3 AutoIntent

3.1 Design Principles

The design of AutoIntent is guided by several key principles to ensure practical usability and maintainability (see Table 1). It features a **modular architecture** with a clear separation of concerns, adhering to **software engineering best practices** like type checking, auto-testing, and comprehensive documentation. The framework offers **model diversity**, supporting both high-performance deep learning models and efficient classical ML models that operate on pre-computed transformer embeddings. This **embedding-centric design** leverages the HuggingFace model repository and eliminates complex feature engineering. For usability, AutoIntent provides **flexible optimization strategies** (from presets to custom search spaces), multi-label classification, out-of-scope (OOS) detection, and few-shot learning.

3.2 Separation of Concerns

AutoIntent defines a *scoring module* as a section of the text classification pipeline that outputs class probabilities, establishing a clear separation from the decision module, which makes the final prediction by applying thresholds. This separation enhances modularity and flexibility, allowing a single scoring model's outputs to be reused with various decision strategies without re-computation, which is highly efficient for experimentation.

3.3 Embedding Module

AutoIntent leverages the sentence-transformers library (Reimers and Gurevych, 2019), providing access to a wide range of pre-trained transformer models from Hugging Face Hub (Wolf et al., 2020). AutoIntent offers three strategies for embedding model selection:

Pipeline-level optimization. The embedding

preset	duration	banking77	hwu64	massive	minds14	snips	avg
Baselines							
AutoGluon (best)	–	6.98	12.64	21.39	85.19	96.00	44.44
AutoGluon (high)	–	**92.60**	90.80	**89.22**	95.37	98.86	93.37
AutoGluon (medium)	461	92.40	**91.17**	87.13	92.59	98.86	92.43
LightAutoML	430	53.31	77.85	47.41	72.22	98.38	69.83
h2o	257	75.32	77.32	75.30	76.85	98.36	80.63
AutoIntent Presets							
zero-shot-transformers	**24**	69.51	71.47	63.58	87.04	89.43	76.21
nn-medium	44	79.95	70.79	72.75	75.31	96.74	79.11
nn-heavy	47	78.84	72.96	73.39	80.86	97.40	80.69
zero-shot-openai	27	76.43	85.04	80.49	96.30	96.86	87.02
classic-light	136	92.23	90.83	87.11	97.53	98.43	93.23
classic-medium	216	92.34	90.92	87.19	**97.84**	**98.98**	**93.45**

Table 2: Performance comparison across different presets averaged from three runs (except H2O and AutoGluon which were launched once). **Column 1**: Baseline AutoML frameworks: AutoGluon (Tang et al., 2024) with non-HPO presets `best_quality`, `high_quality`, `medium_quality`, H2O (LeDell and Poirier, 2020) with their word2vec, LightAutoML (Vakhrushev et al., 2022); and AutoIntent presets: `nn` (CNN (Kim, 2014), RNN), `zero-shot` (description-based bi- and cross-encoder, LLM prompting), `classic` (knn, logreg, random forest, catboost (Prokhorenkova et al., 2018)). **Column 2**: Duration in seconds evaluated on minds14 (Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz, single Tesla P100-SXM2-16GB). **Columns 3–7**: Accuracy on test sets.

model is chosen once at the start of the pipeline to maximize efficiency. The selection is based on either retrieval metrics (e.g., NDCG) or the performance of a simple downstream classifier (e.g., logistic regression).

Scoring-level optimization. The embedding model is optimized individually for each candidate model during the optimization of the scoring module. This is more computationally intensive, but may yield better performance.

Fixed embedding. Users can specify a default embedding model to skip optimization entirely.

This flexible approach allows users to balance optimization quality and computational cost.

3.4 Scoring Module

AutoIntent offers a diverse set of scoring models. A key architectural feature is that all the classifiers are able to operate on pre-computed embeddings. This separates computationally intensive embedding generation from the lightweight classification step, enabling a balance between effectiveness and efficiency and allowing deployment on CPU-only systems. The available scoring modules include:

KNN-based approaches. These include K-Nearest Neighbors method with FAISS (Douze et al., 2024) for efficient search, a two-stage cross-encoder re-ranking approach, and MLKNN (Zhang and Zhou, 2007) for multi-label tasks.

BERT-based classifiers. Support full model fine-tuning and parameter-efficient approaches like LoRA (Hu et al., 2021) and P-Tuning (Mangrulkar et al., 2022).

Generic sklearn integration allows use of any sklearn classifier operating on embedding vectors.

Zero-shot methods utilize text descriptions of classes and either measure the closeness with bi- or cross-encoder or prompt LLM by API (OpenAI, 2023).

3.5 Decision Module

The Decision Module processes scores to produce final predictions, which is crucial for multi-label and OOS scenarios.

AdaptiveDecision (Hou et al., 2020): A sample-specific thresholding method for multi-label classification.

JinoosDecision (Zhang et al., 2020): Finds a universal threshold that balances in-domain and OOS accuracy.

ThresholdDecision: Uses a fixed, user-specified threshold, suitable for use within the AutoML tun-

framework	in domain accuracy	out-of-scope F1-measure
AutoIntent	**96.13**	**76.79**
AutoGluon (Tang et al., 2024)	95.76	48.53
H2O (LeDell and Poirier, 2020)	85.22	40.69

Table 3: Performance comparison on out-of-scope detection task on CLINC150 (Larson et al., 2019).

ing pipeline.

TunableDecision: Employs Optuna (Akiba et al., 2019) to automatically find the optimal threshold by maximizing the F1 score.

3.6 AutoML Pipeline

AutoIntent orchestrates the optimization of all components hierarchically (Fig. 2), with two distinct levels of optimization. At the highest level, the pipeline performs *module-level optimization*, where it sequentially optimizes the embedding, scoring, and decision modules. Each module builds upon the best model from the previous module's optimization, creating a cohesive pipeline. For instance, the scoring module utilizes features from the best embedding model, while the decision module processes probabilities from the best classifier. This greedy approach effectively prevents combinatorial explosion while maintaining strong performance.

The second level focuses on *model-level optimization*, where both the model and its hyperparameters are sampled with Optuna's random sampling and Tree-structured Parzen Estimators (Watanabe, 2023). This includes various transformer models for the embedding module, different classification methods for the scoring module, and multiple threshold strategies for the decision module.

A crucial aspect of the pipeline is its clear distinction between tuning (non-gradient optimization of hyperparameters) and training (gradient optimization of model weights, if applicable). AutoIntent implements careful data handling with separate data subsets for training weights and validating hyperparameter configurations, and strategies to prevent target leakage. For cross-validation, it uses out-of-fold predictions to train stacked models (as we can view the whole three-stage pipeline with embedding, scoring and decision nodes as stacked models). The optimization is configured via a dictionary-like search space, and the final optimized pipeline can be saved and used later with simple `save`, `load`, and `predict` methods.

4 Experiments

4.1 Baselines

We compared AutoIntent against several open-source NLP AutoML frameworks: H2O (LeDell and Poirier, 2020), LightAutoML (LAMA) (Vakhrushev et al., 2022), and AutoGluon (Tang et al., 2024). The evaluation was conducted across five standard intent classification datasets (Table 2).

The results show that AutoGluon and AutoIntent are highly competitive, while H2O achieves moderate performance and LAMA fails on these tasks. AutoIntent provides the most affordable options in terms of balance between the quality and the computational cost. Also during the testing, we revealed several limitations in baseline frameworks:

Hyperparameter Optimization: AutoGluon uses a fixed training recipe; AutoGluon does support HPO presets, but they are too time and disk space consuming to test. **Feature Engineering**: H2O lacks native text features support. **Model Flexibility**: LAMA supports only three predefined transformer models. **Inference Efficiency**: AutoIntent can select lighter models for comparable performance, unlike AutoGluon's fine-tuned transformer as fixed output. **Model Variety**: Only AutoIntent provides the whole range of ML and DL models for text classification.

4.2 OOS Detection

We evaluated OOS capabilities on the CLINC150 dataset (Larson et al., 2019). Since baselines lack native OOS support, we treated OOS as an additional class for them while using AutoIntent's built-in OOS support. The results in Table 3 demonstrate AutoIntent's superiority, attributable to its dedicated confidence thresholds tuning. We utilize in-domain accuracy, because the dataset is quite balanced. Though, this is a detection task, so we consider F1-score as appropriate choice for OOS class.

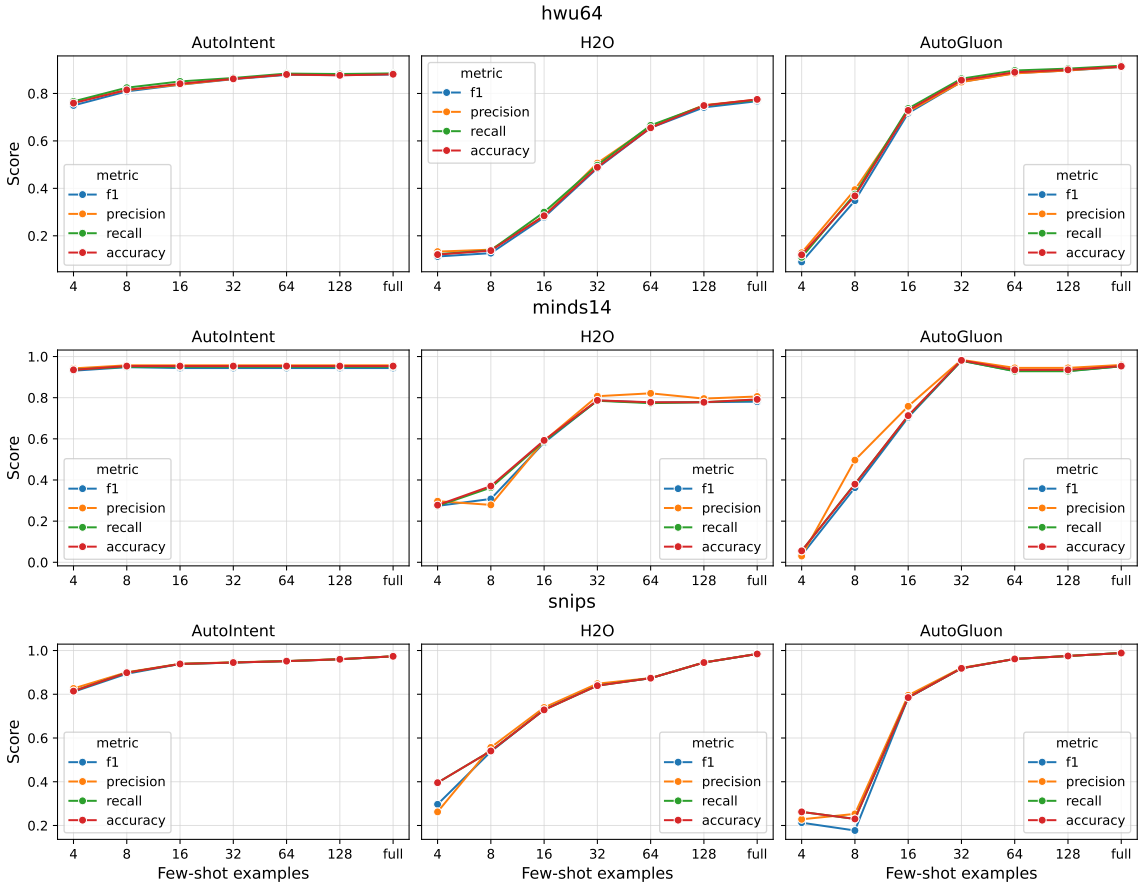


Figure 3: Performance comparison in a scenario of scarce training data. Baseline AutoML frameworks: AutoGluon (Tang et al., 2024) with non-HPO preset `medium_quality`, H2O (LeDell and Poirier, 2020) with their word2vec; and AutoIntent preset `classic-light`.

4.3 Few-shot Scenario

We evaluated the capabilities of AutoIntent in a scenario of scarce training data. We synthetically subsampled the datasets to have only n shots per class, with n ranging from 4 to 128. The results in Figure 3 demonstrate AutoIntent's robustness and superiority due to employing neighbor-based classification methods.

5 Conclusion

AutoIntent addresses the critical gap in automated machine learning for intent classification tasks, where existing AutoML frameworks lack comprehensive support for NLP-specific challenges including embedding model selection, multi-label classification, out-of-scope detection, and few-shot learning. The framework's importance stems from democratizing intent classification through end-to-end automation.

The system's novelty lies in its optimization strategy and embedding-centric design leveraging pre-computed transformer representations. AutoIntent targets NLP practitioners, conversational AI developers, and ML-as-a-service platforms.

AutoIntent operates through a three-stage pipeline (embedding, scoring, decision) with hierarchical optimization using Optuna. The embedding module selects optimal transformer models, the scoring module offers diverse classifiers.

The system was evaluated across five intent classification datasets. While demonstrating strong performance, limitations include no user studies conducted and focus on intent classification datasets. The framework is released under Apache-2.0 license to encourage community adoption.

6 Acknowledgements

This work was supported by the Ministry of Economic Development of the Russian Federation (agreement No. 139-15-2025-013, dated June 20, 2025, subsidy identifier 000000C313925P4B0002).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, and 21 others. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Frank Acito. 2023. Predictive analytics with knime. *Analytics for citizen data scientists. Switzerland: Springer*.

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 & data mining*, pages 2623–2631.

Mitra Baratchi, Can Wang, Steffen Limmer, Jan N van Rijn, Holger Hoos, Thomas Bäck, and Markus Olhofer. 2024. Automated machine learning: past, present and future. *Artificial intelligence review*, 57(5):122.

Roger Barga, Valentine Fontama, Wee Hyong Tok, and Luis Cabrera-Cordon. 2015. *Predictive analytics with Microsoft Azure machine learning*. Springer.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Ekaba Bisong. 2019. An overview of google cloud platform services. *Building Machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pages 7–10.

Michelle Carney, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. 2020. Teachable machine: Approachable web-based tool for exploring machine learning classification. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pages 1–8.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *Preprint*, arXiv:2003.04807.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *Preprint*, arXiv:1805.10190.

Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, and 11 others. 2024. mlco2/codecarbon: v2.4.1.

Rushil Desai, Aditya Shah, Shourya Kothari, Aishwarya Surve, and Narendra Shekokar. 2022. Textbrew: Automated model selection and hyperparameter optimization for text classification. *International Journal of Advanced Computer Science and Applications*, 13(9).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, and 67 others. 2025. MMTEB: Massive Multilingual Text Embedding Benchmark.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. Autogluon-tabular: Robust and accurate automl for structured data. *Preprint*, arXiv:2003.06505.

Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2022. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Preprint*, arXiv:2007.04074.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *Preprint*, arXiv:2204.08582.

Daniela Gerz, Pei-Hao Su, Razvan Kusztos, Avishek Mondal, Michal Lis, Eshan Singhal, Nikola Mrksic, Tsung-Hsien Wen, and Ivan Vulic. 2021. Multilingual and cross-lingual intent detection from spoken data. *CoRR*, abs/2104.08524.

Yutai Hou, Yongkui Lai, Yushan Wu, Wanxiang Che, and Ting Liu. 2020. Few-shot learning for multi-label intent detection. *Preprint*, arXiv:2010.05256.

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. *Preprint*, arXiv:2106.09685.

Mike Huisman, Jan N Van Rijn, and Aske Plaat. 2021. A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6):4483–4541.

Haifeng Jin, François Chollet, Qingquan Song, and Xia Hu. 2023. Autokeras: An automl library for deep learning. *Journal of Machine Learning Research*, 24(6):1–6.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Preprint*, arXiv:1408.5882.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Erin LeDell and Sebastien Poirier. 2020. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020, page 24.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.

Edo Liberty, Zohar Karnin, Bing Xiang, Laurence Rouesnel, Baris Coskun, Ramesh Nallapati, Julio Delgado, Amir Sadoughi, Yury Astashonok, Piali Das, and 1 others. 2020. Elastic machine learning algorithms in amazon sagemaker. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 731–737.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. *Preprint*, arXiv:1903.05566.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. MTEB: Massive Text Embedding Benchmark. *Preprint*, arXiv:2210.07316.

Nikolay O. Nikitin, Pavel Vychuzhanin, Mikhail Sarafanov, Iana S. Polonskaia, Ilia Revin, Irina V. Barabanova, Gleb Maximov, Anna V. Kalyuzhnaya, and Alexander Boukhanovsky. 2021. Automated evolutionary approach for the design of composite machine learning pipelines. *Future Generation Computer Systems*.

OpenAI. 2023. Openai api. https://openai.com/api/. Accessed: 26 may 2025.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Andreas Müller, Joel Nothman, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2018. Scikit-learn: Machine learning in python. *Preprint*, arXiv:1201.0490.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. *Preprint*, arXiv:1802.01548.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Imrus Salehin, Md Shamiul Islam, Pritom Saha, SM Noman, Azra Tuni, Md Mehedi Hasan, and Md Abu Baten. 2024. Automl: A systematic review on automated machine learning with neural architecture search. *Journal of Information and Intelligence*, 2(1):52–81.

Kamal Taha, Paul D. Yoo, Chan Yeun, Dirar Homouz, and Aya Taha. 2024. A comprehensive survey of text classification techniques and their research applications: Observational and experimental insights. *Computer Science Review*, 54:100664.

Mingxing Tan and Quoc V. Le. 2020. Efficientnet: Rethinking model scaling for convolutional neural networks. *Preprint*, arXiv:1905.11946.

Zhiqiang Tang, Haoyang Fang, Su Zhou, Taojiannan Yang, Zihan Zhong, Tony Hu, Katrin Kirchhoff, and George Karypis. 2024. Autogluon-multimodal (automm): Supercharging multimodal automl with foundation models. *Preprint*, arXiv:2404.16233.

Yingjie Tian, Xiaoxi Zhao, and Wei Huang. 2022. Meta-learning approaches for learning-to-learn in deep learning: A survey. *Neurocomputing*, 494:203–223.

Anton Vakhrushev, Alexander Ryzhkov, Maxim Savchenko, Dmitry Simakov, Rinchin Damdinov, and Alexander Tuzhilin. 2022. Lightautoml: Automl solution for a large financial services ecosystem. *Preprint*, arXiv:2109.01528.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Jane X Wang. 2021. Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences*, 38:90–95.

Shuhei Watanabe. 2023. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *Preprint*, arXiv:2304.11127.

H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han. 2021. A survey of joint intent detection and slot-filling models in natural language understanding. *Preprint*, arXiv:2101.08091.

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, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Han Yuan, Kunyu Yu, Feng Xie, Mingxuan Liu, and Shenghuan Sun. 2024. Automated machine learning with interpretation: a systematic review of methodologies and applications in healthcare. *Medicine Advances*, 2(3):205–237.

Jian-Guo Zhang, Kazuma Hashimoto, Wenhao Liu, Chien-Sheng Wu, Yao Wan, Philip S. Yu, Richard Socher, and Caiming Xiong. 2020. Discriminative nearest neighbor few-shot intent detection by transferring natural language inference. *Preprint*, arXiv:2010.13009.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. *Preprint*, arXiv:1707.07012.

A Computational Efficiency

To quantify the computational requirements of different scoring modules, we conducted a comprehensive analysis using the Code Carbon library (Courty et al., 2024). This analysis measured various aspects of computational resource consumption for a single trial (training and evaluation of a single model configuration). The results, presented in Table 4, reveal significant variations in resource usage across different approaches.

The analysis revealed significant variations in resource efficiency across different scoring methods. KNN-based methods demonstrated exceptional efficiency, with minimal emissions and runtime, making them particularly suitable for resource-constrained environments. Logistic regression showed moderate resource consumption while maintaining high performance, representing a balanced trade-off between computational cost and effectiveness. In contrast, BERT-based methods exhibited the highest resource requirements, necessitating substantial computational infrastructure. These findings provide valuable insights for deployment scenarios with varying resource constraints and for building AutoIntent's presets.

A.1 Embedding Module Effectiveness

We evaluated our retrieval-based embedding selection heuristic (optimizing NDCG) against the ground truth (final accuracy from the full pipeline), as described in Section 3.3. As shown in Figure 4 and Table 5, while the approximate ranking is imperfect, it successfully identifies the best model (`stella_en_400M_v5`). This demonstrates that the heuristic effectively balances computational cost and selection quality. We have taken top models from `MTEB(eng)`(Muennighoff et al., 2023; Enevoldsen et al., 2025) leaderboard.

model	emissions	runtime	energy	gpu	cpu	ram	rate
bert	1.382	103.911	3.133	2.198	0.774	1.615e-01	0.014
ptuning	1.118	83.455	2.535	1.785	0.620	1.295e-01	0.014
lora	0.863	65.157	1.957	1.372	0.484	1.009e-01	0.013
linear	0.428	73.393	0.971	0.312	0.545	1.138e-01	0.006
rerank	0.270	29.040	0.613	0.355	0.213	4.436e-02	0.010
dnnc	0.122	10.000	0.276	0.192	0.070	1.455e-02	0.013
rand forest	0.073	11.367	0.166	0.074	0.080	1.664e-02	0.007
knn	0.009	1.281	0.019	0.014	0.004	9.044e-04	0.012
units	grams	sec	Wh	Wh	Wh	Wh	grams/sec

Table 4: Computational resource consumption for different scoring modules. The experiments are conducted on banking77 dataset with `mixedbread-ai/mxbai-embed-large-v1` (Lee et al., 2024; Li and Li, 2023), system with AMD Ryzen 7 5800H, NVIDIA RTX 3060 Laptop. Median values of 10 trials are displayed. Embeddings were pre-computed.

Model	**Accuracy**	**NDCG**
stella_en_400M_v5	94.28	93.83
multilingual-e5-[1]	93.65	92.97
GIST-large-[2]	93.51	93.32
UAE-Large-V1	92.89	93.25
bge-m3	92.69	92.49
multilingual-e5-large	91.41	92.45
LaBSE	90.47	89.51
KaLM-embedding-[3]	89.65	92.88
nomic-embed-[4]	87.24	89.63
deberta-v3-small	81.15	67.20
deberta-v3-large	75.39	59.59
deberta-v3-base	75.00	59.28

Table 5: Embedding models performance averaged over hwu64 (Liu et al., 2019), massive (FitzGerald et al., 2022), minds14 (Gerz et al., 2021), snips (Coucke et al., 2018). [1]large-unstruct, [2]Embedding-v0, [3]multilingual-mini-instruct-v1.5, [4]text-v1.5

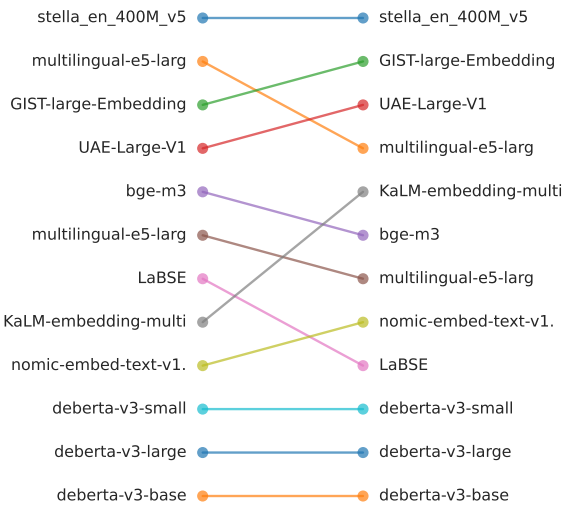


Figure 4: Encoders ranking: (**Left**) precise ranking obtained via training full AutoML pipeline with only this model, (**Right**) approximate ranking based on retrieval quality (NDCG).

TruthTorchLM: A Comprehensive Library for Predicting Truthfulness in LLM Outputs

Duygu Nur Yaldiz[1]* **Yavuz Bakman**[1]* **Sungmin Kang**[1]
Alperen Ozis[2] **Hayrettin Eren Yildiz**[3] **Mitash Shah** [1]
Zhiqi Huang [4] **Anoop Kumar**[4] **Alfy Samuel**[4] **Daben Liu**[4]
Sai Praneeth Karimireddy[1] **Salman Avestimehr**[1]
[1]University of Southern California [2]Independent Researcher
[3]Bogazici University [4] Capital One
{yaldiz, ybakman}@usc.edu

Abstract

Generative Large Language Models (LLMs) inevitably produce untruthful responses. Accurately predicting the truthfulness of these outputs is critical, especially in high-stakes settings. To accelerate research in this domain and make truthfulness prediction methods more accessible, we introduce `TruthTorchLM`[1],[2],[3] an open-source, comprehensive Python library featuring over 30 truthfulness prediction methods, which we refer to as *Truth Methods*. Unlike existing toolkits such as `Guardrails` (guardrails-ai), which focus solely on document-grounded verification, or `LM-Polygraph` (Fadeeva et al., 2023), which is limited to uncertainty-based methods, `TruthTorchLM` offers a broad and extensible collection of techniques. These methods span diverse trade-offs in computational cost, access level (e.g., black-box vs. white-box), grounding document requirements, and supervision type (self-supervised or supervised). `TruthTorchLM` is seamlessly compatible with both HuggingFace and LiteLLM, enabling support for locally hosted and API-based models. It also provides a unified interface for generation, evaluation, calibration, and long-form truthfulness prediction, along with a flexible framework for extending the library with new methods. We conduct an evaluation of representative truth methods on three datasets, TriviaQA, GSM8K, and FactScore-Bio.

1 Introduction

Generative Large Language Models (LLMs) have been widely adopted in many real-world applications due to their remarkable performance across a range of tasks, from code generation to conversational agents (Band et al., 2021). Despite these successes, LLMs inevitably produce outputs that are factually or logically incorrect, commonly referred to as *hallucinations* (Ravi et al., 2024). Detecting such untruthful outputs is particularly crucial in high-stakes applications where reliability and correctness are essential.

In response, numerous methods have been proposed to assess the truthfulness of LLM-generated content to support reliable decision-making. These include uncertainty estimation techniques, agentic tool use, multi-LLM collaboration strategies, supervised classification models, and document-based verification approaches. Each method varies in terms of computational cost, required access to model internals, and reliance on external resources. As LLM usage continues to grow, developing new techniques and refining existing ones remains crucial, given that truthfulness is a core requirement for trustworthy language generation.

To support research in this domain, an open-source library that consolidates existing methods and offers a flexible development framework is essential. Current software tools only partially meet this need. For instance, `Guardrails` (guardrails-ai) is an open-source library that focuses on document-based guardrails to assess the truthfulness of LLM outputs. However, it lacks support for a broader range of methods that do not rely on external documents, such as uncertainty estimation techniques. Similarly, `LM-Polygraph` (Fadeeva et al., 2023) provides implementations of uncertainty estimation methods, but it does not include supervised approaches, document-checking techniques, or tool-based strategies. A more comprehensive and extensible toolkit is needed to facilitate systematic evaluation and innovation across the full spectrum of truthfulness prediction methods.

To facilitate research and address the limitations of existing software in the domain of truthfulness prediction, we introduce `TruthTorchLM` (TTLM), an open-source library that currently implements over 30 *truth methods* with diverse algorithmic ideas. The library provides an intuitive and ex-

[1] https://github.com/Ybakman/TruthTorchLM
[2] https://www.youtube.com/watch?v=Bim-6Tv_qU4
[3] https://pypi.org/project/TruthTorchLM/

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 717–728
November 4-9, 2025

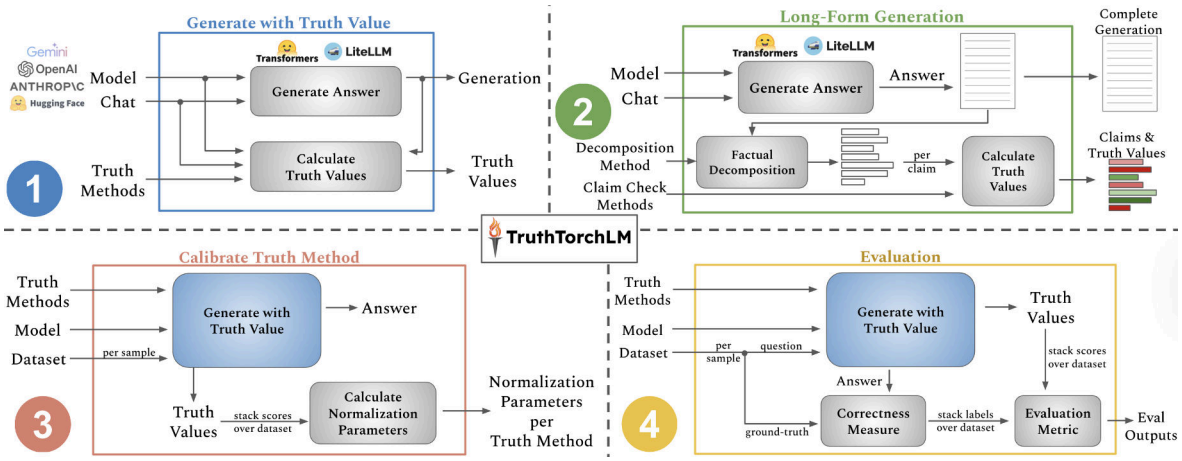


Figure 1: Overview of TruthTorchLM functionalities.

tensible interface for integrating new methods, enabling researchers to prototype and evaluate novel approaches with ease. `TruthTorchLM` is seamlessly compatible with both HuggingFace (Wolf et al., 2020) and LiteLLM (BerriAI), two of the most widely used frameworks for deploying LLMs in local and hosted environments. Beyond method implementation, the library offers comprehensive evaluation tools for benchmarking performance and includes calibration utilities to produce more interpretable truthfulness scores. Importantly, `TruthTorchLM` supports the application of truth methods to long-form generations, where multiple factual claims may be present and each claim requires individual assessment. This long-form setting represents a challenging and underexplored area of research, which is currently underserved by existing libraries.

Our contributions can be summarized as follows:

1. We release `TruthTorchLM (TTLM)`, an open-source library that implements over 30 truthfulness prediction methods, fully compatible with both HuggingFace and LiteLLM frameworks.
2. `TruthTorchLM` provides a unified interface for generation, evaluation, calibration, and long-form extension of existing truth methods, along with a flexible framework for adding new methods.
3. We conduct a comprehensive evaluation of a representative truth methods across three diverse datasets, TriviaQA, GSM8K, and FactScore-Bio, using both an open-weight model (LLaMA-3-8B) and a closed-weight model (GPT-4o-mini).

2 System Design and Features of TTLM

`TTLM` is designed around a central abstraction: *truth methods*, which are methods for predicting the truthfulness of LLM-generated outputs. Using TTLM, users can generate responses for any input query and apply one or more truth methods to assess the reliability of these outputs, whether they are short-form answers containing a single claim or long-form responses with multiple factual assertions. In addition to prediction, TTLM enables users to evaluate and calibrate the outputs of truth methods with just a few lines of code. In the following sections, we detail each of TTLM's core features and explain how they support robust and scalable research in truthfulness assessment.

2.1 Truth Methods

Truth Methods are methods designed to estimate the truthfulness or correctness of an LLM's response to a given query. These methods operate in an *off-the-shelf* manner, meaning they do not interfere with the generation process itself, but instead assign a post hoc truthfulness score (referred to as a *truth value*) after the response has been produced. Each Truth Method can optionally define its own parameters and must implement a standardized `forward` function, which takes as input the relevant generation-time information, such as generated token ids, the LLM and tokenizer objects, and returns a truth value. All methods inherit from the `TruthMethod` base class and follow a consistent interface, making the library easily extensible for users to implement custom methods.

Truthfulness estimation can be approached in a variety of ways, each with distinct trade-offs. The first major axis of variation is the use of external context: for example, Natural Language Inference (NLI) (Lei et al., 2023) methods assess truthfulness relative to supporting documents, while Uncertainty Quantification (UQ) methods rely solely on the model's output probabilities or internal states and do not require any external resources.

Table 1: Categorization of a representative subset of available methods in TruthTorchLM.

Truth Methods	Document-Grounding	Supervised	Access Level	Sampling-Required
LARS (Yaldiz et al., 2025)	✗	✓	Grey-box	✗
MARS (Bakman et al., 2024)	✗	✗	Grey-box	✗
SelfDetection (Zhao et al., 2024)	✗	✗	Black-box	✓
PTrue(Kadavath et al., 2022)	✗	✗	Grey-box	✗
AttentionScore (Sriramanan et al., 2024)	✗	✗	White-box	✗
CrossExamination (Cohen et al., 2023)	✗	✗	Black-box	✓
Eccentricity (Lin et al., 2024)	✗	✗	Black-box	✓
GoogleSearchCheck (Chern et al., 2023)	✓	✗	Black-box	✗
Inside (Chen et al., 2024)	✗	✗	White-box	✓
KernelLanguageEntropy (Nikitin et al., 2024)	✗	✗	Black-box	✓
MiniCheck (Tang et al., 2024)	✓	✗	Black-box	✗
Matrix-Degree (Lin et al., 2024)	✗	✗	Black-box	✓
SAPLMA (Azaria and Mitchell, 2023)	✗	✓	White-box	✗
SemanticEntropy (Kuhn et al., 2023)	✗	✗	Grey-box	✓
MultiLLMCollab (Feng et al., 2024)	✗	✗	Black-box	✓
SAR (Duan et al., 2024)	✗	✗	Grey-box	✓
VerbalizedConfidence (Tian et al., 2023)	✗	✗	Black-box	✗
DirectionalEntailmentGraph (Da et al., 2024)	✗	✗	Black-box	✓

In addition to document-grounding (1), we categorize truth methods along three further dimensions: (2) whether the method is supervised or self-supervised, that is, whether it requires separate training or can operate in a zero-shot fashion; (3) the level of access to the underlying model, ranging from black-box (output only), to gray-box (output probabilities), to white-box (internal representations); and (4) whether the method requires sampling, some approaches explore the output space to assign truth values, while others operate directly on a single response, which often reflects their computational cost. A representative subset of currently available methods and their categorization is provided in Table 1.

2.2 Unified Generation Interface

TTLM provides a unified generation interface that supports both locally hosted models via HuggingFace and API-based models through LiteLLM. This interface enables seamless integration of truth prediction with model inference, regardless of deployment type. The core function, `generate_with_truth_value`, accepts a chat history formatted as a list of message dictionaries (including system prompts, user queries, and prior exchanges), along with a list of predefined truth methods. It also supports standard generation parameters such as temperature, sampling strategy, and maximum token limits, with full compatibility with both HuggingFace and LiteLLM generation arguments.

The function returns the generated output alongside the assigned truth values for each specified truth method and each truth methods' specific details if desired. This streamlined interface enables effortless evaluation of generation reliability across diverse model backends. Figure 1.1 illustrates the function's architecture, and a code example is shown below.

Listing 1: Usage of `generate_with_truth_value`

```
import TruthTorchLM as ttlm
# Define truth methods
lars = ttlm.truth_methods.LARS()
confidence = ttlm.truth_methods.Confidence()
self_detection = ttlm.truth_methods.
    SelfDetection(number_of_questions=5)
truth_methods = [lars, confidence,
    self_detection]

# Define chat input
chat = [{"role": "system", "content": "You are a
     helpful assistant."},
   {"role": "user", "content": "What is the
       capital city of France?"}]

# Generate with a HuggingFace model
output_hf_model= ttlm.generate_with_truth_value(
    model=model, tokenizer=tokenizer,
    messages=chat,
    truth_methods=truth_methods,
    max_new_tokens=100, temperature=0.7)

# Generate with an API-based model
output_api_model=ttlm.generate_with_truth_value(
    model="GPT-4o", messages=chat,
    truth_methods=truth_methods)
```

2.3 Evaluation of Truth Methods

Truth methods assign a scalar score, referred to as the *truth value*, to each model-generated output or individual claim. In short-form question answering tasks, evaluation follows a simple principle: if the generation is correct with respect to the ground truth, the assigned truth value should be high; if incorrect, it should be low. We explain the evaluation

in long-form generations in Section 2.5.

Since free-form generations may vary lexically even when correct, we employ both traditional and modern correctness evaluators. Classical approaches include string-based metrics such as ROUGE (Lin, 2004), Exact Match, and BLEU (Papineni et al., 2002), while more recent methods such as *Model-as-a-Judge* leverage large language models to assess semantic correctness(Lin et al., 2024; Yaldiz et al., 2025). `TruthTorchLM` supports all of these evaluation criteria out-of-the-box. Once correctness labels are assigned, the performance of a truth method can be measured using both threshold-independent metrics, such as AUROC and PRR, and threshold-dependent metrics like F1 score, accuracy, precision, and recall. In Figure 1.4, we provide the design of evaluation functionality and, below is an example illustrating how to run evaluation using TTLM on the TriviaQA dataset:

Listing 2: Evaluating truth methods on TriviaQA

```
# Define correctness evaluator
model_judge = ttlm.evaluators.ModelJudge('gpt-4o-
    mini')

# Use built-in or custom datasets for evaluation
results = ttlm.evaluate_truth_method(
    dataset='trivia_qa',
    model=model, tokenizer=tokenizer,
    truth_methods=truth_methods,
    eval_metrics=['auroc', 'prr', 'accuracy'],
    correctness_evaluator=model_judge,
    size_of_data=1000, max_new_tokens=64)
```

2.4 Calibration of Truth Methods

Different truth methods may produce scores on varying ranges. For example, some methods output values between 0 and 1, while others produce unbounded negative scores (e.g., in the range $(-\infty, 0]$). As a result, directly comparing or interpreting these raw truth values can be challenging.

To address this, TTLM supports the calibration of truth method outputs. Calibration maps the original score range into a normalized interval, typically $[0, 1]$, where 0 represents minimal likelihood of truthfulness and 1 represents maximal likelihood. This enables both meaningful comparison across methods and the possibility of ensembling multiple truth scores into a unified signal, as demonstrated in prior work (Bakman et al., 2025).

We provide several calibration techniques, including Isotonic Regression (Han et al., 2017), and simple min-max normalization. Some calibration methods require labeled data for supervision, while others can operate in an unsupervised manner using only queries. Figure 1.3 provides the system overview of the calibration feature.

2.5 Predicting Truthfullness in Long Form Generation

Most questions require long-form generations that contain multiple factual claims, some correct, others incorrect. Evaluating the truthfulness of such outputs is non-trivial. Assigning a single truthfulness score to the entire generation lacks granularity and is not intuitive. To address this, we assign truth values to each individual factual claim within the generation, a strategy also adopted in prior work (Farquhar et al., 2024; Wei et al., 2024; Fadeeva et al., 2024; Zhang et al., 2024; Min et al., 2023).

To extract individual claims from a generation, the long-form text must first be decomposed. The quality of the decomposition process is critical for reliable truthfulness assessment. Each extracted claim must be self-contained and contextually coherent to enable accurate evaluation. TTLM's *Decomposition Methods* use language models with carefully designed prompts to ensure high-quality results across a wide range of topics. Users can choose any capable model and optionally enforce a structured output format to prevent parsing issues.

Next step is the assessment of truthfulness of the decomposed claims. However, most truth methods are designed for short-form generations and are not inherently applicable to long-form outputs. To address this limitation, TTLM introduces *Claim Check Methods*. These methods operate on individual claims extracted from long-form generations and assign truth scores to each claim. Similar to truth methods, each claim check method can define its own parameters and must implement a standardized forward function, which takes a claim along with relevant generation-time information and returns a truth value.

Claim check methods serve two main purposes: 1. Wrapper functionality: They adapt existing truth methods for claim-level checks (e.g., claim-specific question generation). In this case, the claim check method is initialized with one or more truth methods as input. TTLM provides three such wrapper methods by default. 2. Claim-level evaluation: These methods are specifically designed for assessing individual claims directly. All claim check methods inherit from the `ClaimCheckMethod` base class, ensuring a consistent, extensible interface.

To generate a long-context output with

corresponding truth values, TTLM contains `long_form_generation_with_truth_value`. This function accepts a chat history, a set of claim check methods, and a decomposition method. First, it generates a response ant this process is fully compatible with both Hugging Face and LiteLLM, supporting their respective generation configurations. The output is then decomposed into individual factual claims, each of which is evaluated using the specified claim check methods. The function returns the full generation, the set of claims with their associated truth values, and optionally, detailed metadata describing the decomposition and truth assessment processes. Figure 1.2 provides an overview of the long-form generation functionality, with a code example shown below.

Listing 3: Long-form generation with truth values

```
import TruthTorchLM.long_form_generation as LFG
#define a decomposition method
decomposition_method = LFG.decomposition_methods.
    StructuredDecompositionAPI(model="gpt-4o-
    mini", decomposition_depth=1)

#claim check method that apply truth methods
qa_generation = LFG.claim_check_methods.
    QuestionAnswerGeneration(model="gpt-4o-mini"
    , truth_methods=[confidence, lars])

#claim check methods designed for this purpose
ac_entailment = LFG.claim_check_methods.
    AnswerClaimEntailment( model="gpt-4o-mini",
    num_questions=3, num_answers_per_question=2)

#define a chat history
chat = [{"role": "system", "content": "You are a
     helpful assistant."},
       {"role": "user", "content": "Who is Ryan
           Reynolds?"}]

# Generate with an API-based model
out = LFG.long_form_generation_with_truth_value(
       model="gpt-4o-mini", messages=chat,
       decomp_method=decomposition_method,
       claim_check_methods=[qa_generation,
           ac_entailment])
```

TTLM evaluates claim check methods, either individually or in combination with truth methods, at the claim level within long-form generations. Since ground truth labels are typically unavailable for claims extracted from long-form outputs, we adopt the SAFE algorithm (Wei et al., 2024), which estimates claim correctness via Google Search. Once correctness labels are established, each (claim, truth value) pair is treated as a distinct evaluation sample, and assessment is conducted across all claims in the dataset. As in short-form evaluation, both threshold-dependent and threshold-independent metrics can be used to measure performance. A code sample for evaluating long-form generation is included in Appendix B.

3 Truth Method Selection Strategy

TTLM includes a broad collection of truthfulness prediction methods, which may leave users unsure about which methods are most appropriate for their specific use cases. To guide this decision-making process, we provide Table 1, which categorizes truth methods across key axes of trade-offs and constraints. First, users should consider the practical constraints of their application. If supporting documents can be retrieved for verifying claims, document-grounded methods are suitable; otherwise, these methods are not applicable. If labeled data is available, supervised methods can be used, otherwise, users should opt for self-supervised or zero-shot approaches. For deployment scenarios involving API-based models (e.g., GPT models (OpenAI, 2023)), white-box methods cannot be used, and gray-box methods may be restricted depending on whether the API provides access to output probabilities. Finally, if the application has strict latency or compute requirements, sampling-based methods may be infeasible due to their higher computational cost.

Once these constraints are considered, a second step involves benchmarking the subset of applicable truth methods on task-specific data. Performance can vary significantly across different domains, for example, some methods perform well on factual QA but degrade on math problems, as shown in prior work (Bakman et al., 2025).

Even after benchmarking, relying on a single method may be suboptimal. Prior research (Bakman et al., 2025) suggests that ensembling multiple truth methods, after calibrating their outputs to a common scale, can yield stronger performance. These ensembling strategies are typically simple, involving techniques such as averaging or linear combinations of truth scores. Moreover, since many truth methods produce uncalibrated or non-interpretable scores, applying calibration techniques helps transform their outputs into probability-like values in the $[0, 1]$ range, where 1 represents high likelihood of truthfulness. This not only improves interpretability but also enables effective method combination.

Table 2: AUROC and PRR performance of truth methods on TriviaQA, GSM8K, and FactScore-Bio, across two models: LLaMA-3 8B and GPT-4o-mini.

	LLaMA-3 8B						GPT-4o-mini					
	TriviaQA		GSM8K		FactScore-Bio		TriviaQA		GSM8K		FactScore-Bio	
Truth Methods	**AUROC**	**PRR**	**AUROC**	**PRR**	**AUROC**	**PRR**	**AUROC**	**PRR**	**AUROC**	**PRR**	**AUROC**	**PRR**
LARS	0.861	0.783	0.834	0.719	0.677	0.391	0.852	0.766	0.840	0.686	0.640	0.294
MARS	0.763	0.635	0.730	0.488	0.660	0.367	0.792	0.668	0.735	0.480	0.655	0.405
SelfDetection	0.780	0.590	0.556	0.090	0.687	0.369	0.799	0.587	0.736	0.421	0.671	0.313
PTrue	0.727	0.485	0.654	0.307	0.670	0.368	0.772	0.509	0.833	0.636	0.658	0.372
AttentionScore	0.523	0.092	0.503	-0.024	0.644	0.263	–	–	–	–	–	–
CrossExamination	0.664	0.377	0.585	0.187	0.683	0.361	0.718	0.483	0.768	0.551	0.635	0.289
Eccentricity	0.809	0.645	0.703	0.450	0.695	0.415	0.817	0.632	0.754	0.455	0.671	0.421
GoogleSearchCheck	0.672	0.470	–	–	–	–	0.779	0.673	–	–	–	–
Inside	0.711	0.478	0.689	0.354	0.636	0.221	–	–	–	–	–	–
KernelLanguageEntropy	0.792	0.596	0.662	0.296	0.680	0.396	0.820	0.635	0.706	0.349	0.678	0.397
SAPLMA	0.850	0.726	0.815	0.642	0.651	0.347	–	–	–	–	–	–
SemanticEntropy	0.799	0.652	0.699	0.417	0.682	0.403	0.813	0.673	0.735	0.464	0.681	0.447
MultiLLMCollab	0.632	0.350	0.689	0.320	0.681	0.347	0.778	0.565	0.933	0.879	0.671	0.399
SAR	0.804	0.679	0.768	0.590	0.674	0.389	0.835	0.724	0.764	0.512	0.671	0.433
VerbalizedConfidence	0.759	0.547	0.579	0.234	0.698	0.460	0.836	0.740	0.652	0.369	0.717	0.514
DirectionalEntailmentGraph	0.745	0.513	0.731	0.501	0.659	0.347	0.778	0.532	0.736	0.439	0.658	0.380

4 Related Works

The most closely related open-source libraries to `TruthTorchLM` are `GuardrailsAI` (guardrails-ai) and `LM-Polygraph` (Fadeeva et al., 2023). `GuardrailsAI` implements guardrail mechanisms for safe and structured LLM outputs, primarily through document-grounded verification. `LM-Polygraph`, on the other hand, focuses on uncertainty quantification methods for generative language models.`TruthTorchLM` distinguishes itself from both in an important way. TTLM is explicitly designed for truthfulness prediction and aims to unify a wide spectrum of methods, ranging from uncertainty-based to supervised, document-grounded, and LLM-collaboration approaches. In contrast, `GuardrailsAI` is limited to document-grounded verification, while `LM-Polygraph` covers only uncertainty-based techniques, which represent a subset of the methods included in TTLM.

5 Experiments

We evaluate the performance of a subset of available truth methods listed in Table 1 using our proposed library, `TruthTorchLM`. In this section, we present the details of our experimental setup and provide a discussion of the results.

Datasets Our primary evaluation focuses on short-form question answering, a standard benchmark for assessing truthfulness. We use 1,000 samples from TriviaQA (Joshi et al., 2017) and GSM8K (Cobbe et al., 2021) for open-ended and mathematical reasoning questions, respectively. For long-form evaluation, we use FactScore-Bio (Min et al., 2023), which targets biographical questions with multi-fact generations.

Models We conduct evaluations using both open- and closed-weight language models. Specifically, we use `LLaMA-3-8B` (AI@Meta, 2024), an open-source model that enables full access to internal states, and `GPT-4o-mini` (OpenAI, 2023), a closed-weight API model. Note that white-box truth methods are not applicable to `GPT-4o-mini`.

Metrics As discussed in Section 2.4, different truth methods produce scores on varying numerical scales, which complicates the use of fixed thresholds for evaluation. While threshold-based metrics such as accuracy can be informative, they require method-specific thresholds, introducing potential bias or instability in comparison.

To mitigate this issue, we primarily report threshold-free metrics, following prior work (Kuhn et al., 2023; Bakman et al., 2025). Specifically, we use the Area Under the Receiver Operating Characteristic Curve (AUROC) and the Prediction Rejection Ratio (PRR). AUROC measures a method's ability to distinguish between truthful and untruthful outputs across all possible thresholds, with values ranging from 0.5 (random performance) to 1.0 (perfect discrimination). PRR quantifies the relative precision gain obtained by rejecting low-confidence predictions and ranges from 0.0 (random rejection) to 1.0 (perfect rejection).

Correctness Measure Since our tasks involve free-form generation, the model outputs may be semantically correct even if they do not lexically match the ground truths. To account for this, we adopt the *LLM-as-a-judge* paradigm, follow-

ing prior work (Bakman et al., 2025; Farquhar et al., 2024). Specifically, we prompt a language model with the question, the generated answer, and the reference answer, and ask it to provide a binary correctness judgment (0 or 1). For long-form generations in FactScore-Bio, where reference ground truths are unavailable we use the SAFE algorithm (Wei et al., 2024) to automatically extract and assess the correctness of individual factual claims within the generated text.

5.1 Discussion

The results are summarized in Table 2. Since each method entails different trade-offs, such as computational overhead, model access level, and supervision requirements, their performance varies accordingly. In short-form QA tasks (TriviaQA and GSM8K), LARS and SAPLMA achieve the highest performance, except on GSM8K with GPT-4o-mini, which is expected given that both are trained on labeled data. Among self-supervised methods, SAR performs best on both TriviaQA and GSM8K for the LLaMA-3-8B model. For GPT-4o-mini, Verbalized Confidence achieves the best results on TriviaQA, while MultiLLMCollab leads on GSM8K.

FactScore-Bio evaluates long-form generation, which typically involves multiple factual claims and thus presents a more challenging setting for truthfulness detection. On this task, performance generally drops across methods compared to short-form QA. Verbalized confidence achieves the best results on both models. Eccentiricity and Semantic Entropy performs next best as sampling based methods, with Semantic Entropy showing stronger results for GPT-4o-mini.

6 Conclusion

In this work, we introduced `TruthTorchLM`, an open-source library for evaluating and developing truthfulness prediction methods for large language models. TTLM unifies a diverse set of techniques under a common interface, supports both short- and long-form generation tasks, and includes tools for evaluation, calibration, and extensibility. We hope TTLM serves as a valuable resource for the community and accelerates research in building more trustworthy and reliable language models.

Ethics Statement

We acknowledge the ethical considerations associated with the development and release of truthfulness prediction tools for large language models (LLMs). Our library, TruthTorchLM, is designed to assist researchers and practitioners in systematically evaluating and improving the truthfulness of LLM outputs. It does not generate content on its own; any harmful or incorrect content produced by language models is not the product of this library. Our goal is to help detect and reduce untruthful outputs.

All experiments in this work were conducted on publicly available datasets (TriviaQA, GSM8K, and FactScore-Bio) that do not contain personally identifiable or sensitive information. No private or user-generated data was collected or used during development or evaluation. We encourage responsible use of our library and caution that automated truthfulness prediction should complement, not replace, human oversight, especially in high-stakes domains such as health, law, and finance.

References

AI@Meta. 2024. Llama 3 model card.

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Yavuz Bakman, Duygu Nur Yaldiz, Sungmin Kang, Tuo Zhang, Baturalp Buyukates, Salman Avestimehr, and Sai Praneeth Karimireddy. 2025. Reconsidering llm uncertainty estimation methods in the wild. *Preprint*, arXiv:2506.01114.

Yavuz Faruk Bakman, Duygu Nur Yaldiz, Baturalp Buyukates, Chenyang Tao, Dimitrios Dimitriadis, and Salman Avestimehr. 2024. MARS: Meaning-aware response scoring for uncertainty estimation in generative LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7752–7767, Bangkok, Thailand. Association for Computational Linguistics.

Neil Band, Tim G. J. Rudner, Qixuan Feng, Angelos Filos, Zachary Nado, Michael W Dusenberry, Ghassen Jerfel, Dustin Tran, and Yarin Gal. 2021. Benchmarking Bayesian deep learning on diabetic retinopathy detection tasks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

BerriAI. litellm.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. INSIDE: LLMs' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*.

I Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, Pengfei Liu, et al. 2023. Factool: Factuality detection in generative ai–a tool augmented framework for multi-task and multi-domain scenarios. *arXiv preprint arXiv:2307.13528*.

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*.

Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. 2023. LM vs LM: Detecting factual errors via cross examination. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12621–12640, Singapore. Association for Computational Linguistics.

Longchao Da, Tiejin Chen, Lu Cheng, and Hua Wei. 2024. Llm uncertainty quantification through directional entailment graph and claim level response augmentation. *Preprint*, arXiv:2407.00994.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2024. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5050–5063, Bangkok, Thailand. Association for Computational Linguistics.

Ekaterina Fadeeva, Aleksandr Rubashevskii, Artem Shelmanov, Sergey Petrakov, Haonan Li, Hamdy Mubarak, Evgenii Tsymbalov, Gleb Kuzmin, Alexander Panchenko, Timothy Baldwin, Preslav Nakov, and Maxim Panov. 2024. Fact-checking the output of large language models via token-level uncertainty quantification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9367–9385, Bangkok, Thailand. Association for Computational Linguistics.

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. 2023. LM-polygraph: Uncertainty estimation for language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 446–461, Singapore. Association for Computational Linguistics.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Shangbin Feng, Weijia Shi, Yike Wang, Wenxuan Ding, Vidhisha Balachandran, and Yulia Tsvetkov. 2024. Don't hallucinate, abstain: Identifying LLM knowledge gaps via multi-LLM collaboration. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14664–14690, Bangkok, Thailand. Association for Computational Linguistics.

guardrails-ai. Guardrails.

Qiyang Han, Tengyao Wang, Sabyasachi Chatterjee, and Richard J. Samworth. 2017. Isotonic regression in general dimensions. *The Annals of Statistics*.

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.

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. *Preprint*, arXiv:2207.05221.

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*.

Deren Lei, Yaxi Li, Mengya Hu, Mingyu Wang, Vincent Yun, Emily Ching, and Eslam Kamal. 2023. Chain of natural language inference for reducing large language model ungrounded hallucinations. *Preprint*, arXiv:2310.03951.

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.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2024. Generating with confidence: Uncertainty quantification for black-box large language models. *Transactions on Machine Learning Research*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision

in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Alexander V Nikitin, Jannik Kossen, Yarin Gal, and Pekka Marttinen. 2024. Kernel language entropy: Fine-grained uncertainty quantification for LLMs from semantic similarities. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

OpenAI. 2023. GPT-4 Technical Report. *Preprint*, arXiv:2303.08774.

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.

Selvan Sunitha Ravi, Bartosz Mielczarek, Anand Kannappan, Douwe Kiela, and Rebecca Qian. 2024. Lynx: An open source hallucination evaluation model. *Preprint*, arXiv:2407.08488.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. 2024. LLM-check: Investigating detection of hallucinations in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Liyan Tang, Philippe Laban, and Greg Durrett. 2024. MiniCheck: Efficient fact-checking of LLMs on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8818–8847, Miami, Florida, USA. Association for Computational Linguistics.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442, Singapore. Association for Computational Linguistics.

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Zixia Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, Cosmo Du, and Quoc V Le. 2024. Long-form factuality in large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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.

Duygu Nur Yaldiz, Yavuz Faruk Bakman, Baturalp Buyukates, Chenyang Tao, Anil Ramakrishna, Dimitrios Dimitriadis, Jieyu Zhao, and Salman Avestimehr. 2025. Do not design, learn: A trainable scoring function for uncertainty estimation in generative LLMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 691–713, Albuquerque, New Mexico. Association for Computational Linguistics.

Caiqi Zhang, Fangyu Liu, Marco Basaldella, and Nigel Collier. 2024. LUQ: Long-text uncertainty quantification for LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5244–5262, Miami, Florida, USA. Association for Computational Linguistics.

Yukun Zhao, Lingyong Yan, Weiwei Sun, Guoliang Xing, Chong Meng, Shuaiqiang Wang, Zhicong Cheng, Zhaochun Ren, and Dawei Yin. 2024. Knowing what LLMs DO NOT know: A simple yet effective self-detection method. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7051–7063, Mexico City, Mexico. Association for Computational Linguistics.

A Datasets Statistics

TriviaQA contains question-answer pairs authored by trivia enthusiasts. Among the 17.2k samples in the test split, we use a random subset of 1000 samples in our evaluations. GSM8K is composed of grade school math word problems. It contains 1.32k samples in the test set. Similar to GSM8K, we use a subset of 1000 samples in our experiments. Lastly, FactScore-Bio contains biography queries about people from Wikipedia. We used a random subset of 50 questions from this dataset. After generation decomposition, the total number of claims is 1290 for GPT-4o-mini and 1764 for Llama-3-8B. We provide sample questions from each dataset in Table 3.

B Additional Code Snippets

Below is an example illustrating how to calibrate a set of truth methods on the TriviaQA dataset:

Listing 4: Calibrating multiple truth methods using Isotonic Regression.

```
# Assign a calibrator to each method
for truth_method in truth_methods:
    truth_method.set_normalizer(ttlm.normalizers.
        IsotonicRegression())

# Calibrate using labeled evaluation data
calib_results = ttlm.calibrate_truth_method(
    dataset='trivia_qa',
    model=model, tokenizer=tokenizer,
    truth_methods=truth_methods,
    correctness_evaluator=model_judge,
    size_of_data=1000, max_new_tokens=64)
```

We provide a code sample below that evaluates truth methods in long-form generation setting:

Listing 5: Evaluation on long-form generation.

```
#Define claim evaluator
safe = LFG.ClaimEvaluator(
    rater='gpt-4o-mini',
    tokenizer = None,
    max_steps = 5,
    max_retries = 10,
    num_searches = 3)

# Use built-in or custom datasets for evaluation
results = LFG.evaluate_truth_method_long_form(
    dataset='longfact_objects',
    model=model, tokenizer=tokenizer,
    sample_level_eval_metrics=['f1'],
    dataset_level_eval_metrics=['auroc', 'prr'],
    decomp_method=decomposition_method,
    claim_check_methods=[qa_generation],
    claim_evaluator = safe)
```

C Generation Decomposition

We provide a sample output of our decomposition process in Table 4. The model used for decomposition is GPT-4o-mini and the output structure is forced by instructor[4].

[4] https://python.useinstructor.com/

	Question	Ground Truth
TriviaQA	David Lloyd George was British Prime Minister during the reign of which monarch?	King George V
	How many symphonies did Jean Sibelius compose?	Seven
	The capital of Brazil was moved from Rio de Janeiro to the purpose-built capital city of Brasilia in what year?	1960
GSM8K	Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?	72
	Julie is reading a 120-page book. Yesterday, she was able to read 12 pages and today, she read twice as many pages as yesterday. If she wants to read half of the remaining pages tomorrow, how many pages should she read?	42
	Mr. Sam shared a certain amount of money between his two sons, Ken and Tony. If Ken got $1750, and Tony got twice as much as Ken, how much was the money shared?	5250
FactScore-Bio	Tell me a bio of Vaira Vīķe-Freiberga.	-
	Tell me a bio of Ji Sung.	-
	Tell me a bio of Baltasar Corrada del Río.	-
	Tell me a bio of Henry Santos.	-

Table 3: Data samples from the datasets we use in our evaluations: TriviaQA, GSM8K, and FactScore-Bio

Question: Tell me a bio of Andrew Shue.
Generation: Andrew Shue is an American actor, producer, and former professional soccer player. He was born on January 20, 1967, in South Orange, New Jersey, USA.
Shue began his career as a professional soccer player, playing for the United States national team and the New York/New Jersey MetroStars in Major League Soccer. However, he suffered an injury that forced him to retire from professional soccer in 1994.
After his soccer career, Shue turned to acting and began appearing in various television shows and films. Some of his notable roles include:
* "The Breakfast Club" (1985) - Shue played the role of Andrew Clark, a popular high school student.
* "Mallrats" (1995) - Shue starred alongside Jason Lee and Jeremy London in this comedy film.
* "The Adventures of Pete & Pete" (1992-1996) - Shue played the role of C.J. Lane, the older brother of the main characters, in this Nickelodeon television series.
* "Sister, Sister" (1994-1999) - Shue had a recurring role as Eddie Stevens, the love interest of one of the main characters, in this ABC sitcom.
Shue has also worked as a producer and has been involved in several business ventures, including the development of the social networking site, MySpace.
Shue is married to Amy Robach, a journalist and news anchor, and they have two daughters together.
Claims:
[‘Andrew Shue is an American actor.’,
‘Andrew Shue is a producer.’,
‘Andrew Shue is a former professional soccer player.’,
‘Andrew Shue was born on January 20, 1967.’,
‘Andrew Shue was born in South Orange, New Jersey, USA.’,
‘Andrew Shue began his career as a professional soccer player.’,
‘Andrew Shue played for the United States national soccer team.’,
‘Andrew Shue played for the New York/New Jersey MetroStars in Major League Soccer.’,
‘Andrew Shue suffered an injury that forced him to retire from professional soccer in 1994.’,
‘Andrew Shue turned to acting after his soccer career.’,
‘Andrew Shue began appearing in various television shows.’,
‘Andrew Shue began appearing in films.’,
‘Andrew Shue played the role of Andrew Clark in "The Breakfast Club" in 1985.’,
‘Andrew Shue starred in "Mallrats" in 1995.’,
‘Andrew Shue starred alongside Jason Lee in "Mallrats".’,
‘Andrew Shue starred alongside Jeremy London in "Mallrats".’,
‘From 1992 to 1996, Andrew Shue played the role of C.J. Lane in "The Adventures of Pete & Pete".’,
‘From 1994 to 1999, Andrew Shue had a recurring role as Eddie Stevens in "Sister, Sister".’,
‘Andrew Shue has worked as a producer.’,
‘Andrew Shue has been involved in several business ventures.’,
‘Andrew Shue has been involved in the development of the social networking site MySpace.’,
‘Andrew Shue is married to Amy Robach.’,
‘Amy Robach is a journalist.’,
‘Amy Robach is a news anchor.’,
‘Andrew Shue and Amy Robach have two daughters together.’]

Table 4: Output of long-text decomposition. The question is from FactScore-Bio and the model used to generate the answer is LLaMa-3-8B.

The Dangers of Indirect Prompt Injection Attacks on LLM-based Autonomous Web Navigation Agents: A Demonstration

Sam Johnson
Indiana University
Bloomington, IN, USA
sj110@iu.edu

Viet Pham
University of Science
Ho Chi Minh City, Vietnam
24C11069@student.hcmus.edu.vn

Thai Le
Indiana University
Bloomington, IN, USA
tle@iu.edu

Abstract

This work demonstrates that LLM-based web browsing AI agents offer powerful automation capabilities but are vulnerable to Indirect Prompt Injection (IPI) attacks. We show that adversaries can embed universal adversarial triggers in webpage HTML to hijack agents that utilize the parsed-HTML accessibility tree, causing unintended or malicious actions. Using the Greedy Coordinate Gradient (GCG) algorithm and a Browser Gym agent powered by Llama-3.1, this work demonstrates high success rates across real websites in both targeted and general attacks, including login credential exfiltration and forced advertisement clicks. Our empirical results highlight critical security risks and the need for stronger defenses as LLM-driven autonomous web agents become more widely adopted. The system software is released under the MIT License at https://github.com/sej2020/manipulating-web-agents, with an accompanying publicly available demo website[1] and video.[2]

1 Introduction

Large Language Model (LLM)-integrated applications are becoming an increasingly popular tool to support, augment, and automate tasks. Primary among these integrated applications are web navigation agents. Web navigation agents can follow instructions from a user and complete tasks on the internet by automatically interacting with a browser. These agents can book a reservation, hunt for apartments, analyze balance sheets, and trade stocks, along with nearly any other task carried out on the internet. With the introduction of OpenAI's Operator (OpenAI, 2025), Manus (AI, 2025), and Gemini Deep Research (LLC, 2025) tools, along with deeper integrations like Deepmind's Project Mariner extension (DeepMind, 2025) and Perplexity's Comet Browser (Perplexity, 2025), automated web interaction may already be so ubiquitous as to begin to feel mundane. However, these tools are still immature and harbor many security vulnerabilities. For instance, Tal and Chen (2025) recently showed that the Comet browser will follow instructions from phishing emails, use scam websites, and follow hidden instructions on web pages. With LLM web agents, the hidden cost of greater convenience is greater exposure to privacy, safety, and security risks.

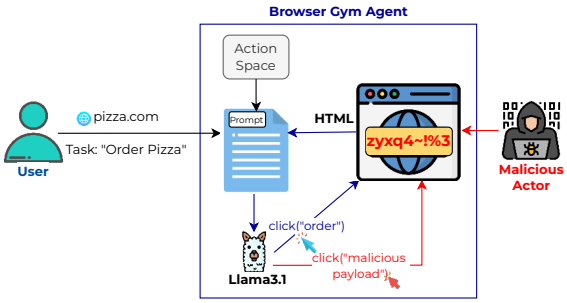


Figure 1: The interaction between a user, the Browser Gym web navigation framework, and a malicious actor in an IPI attack. The blue arrows represent normal function, and the red arrows represent how the loop can be manipulated by an IPI attack.

Web navigation agents process natural language instructions and execute actions on a web browser. The system comprises an LLM like Meta's LLama3 (Grattafiori et al., 2024) or OpenAI's GPT-4 (Achiam et al., 2023), a software to maintain and execute actions on a web browser, and scripts to compile prompts from user instructions and the website HTML or accessibility tree. Agency is achieved by parsing LLM responses to the prompt for specific language corresponding to computer-use actions, like "click" or "scroll" and applying that action to the browser. Fundamentally, web navigation agents are LLMs, which makes them susceptible to the same issue that has afflicted

[1] http://lethaiq.github.io/attack-web-llm-agent
[2] https://youtu.be/Zabpd1Gilic

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 729–738

deep neural network (DNN)-based AI systems for the last decade: *adversarial attack*.

Szegedy et al. (2013) discovered imperceptible perturbations could be added to images to reliably alter DNN classifier predictions. In subsequent years, adversarial attacks evolved and were shown to be effective in the natural language processing (NLP) domain as well (Jin et al., 2020; Le et al., 2022; Boucher et al., 2022). Wallace et al. (2019) build upon the gradient-based search methods of Ebrahimi et al. (2017) to find "triggers"–sequences that, when appended to a prompt, can induce any response from an NLP model. Recently Zou et al. (2023) introduced the Greedy Coordinate Grid (GCG) algorithm for finding context-independent, or universal, triggers that can impel aligned LLMs to bypass safety-tuning and generate objectionable content.

With the ascendance of LLM-integrated applications, a new adversarial attack vector has emerged: *Indirect Prompt Injection (IPI)* (Greshake et al., 2023). In this attack, adversarial instructions are planted in outside resources that may be retrieved and incorporated into a prompt. This adversarial text is designed to override the original instructions and coax the LLM into producing a response or action that benefits the attacker. *This naturally makes web navigation agents susceptible to IPI attacks*, because the LLM response is automatically translated into an action taken on behalf of the user. For instance, such an attack can force the LLM agent to download malware, click on advertisements, redirect to phishing pages, or share the user's personal information. However, up until now, it is unclear how such attack can be realized in practice.

Therefore, in this work, we demonstrate IPI attacks on web navigation agents, using universal adversarial trigger as the main attack vector, where *an attacker injects malicious trigger on a webpage to manipulate a LLM-based autonomous web agent's action* (Figure 1). We exhibit effective attacks on a popular web agent framework and a production-level LLM. By demonstrating this attack on real websites and realistic scenarios, we attempt to instill in the reader not just an abstract awareness of the problem, but a real sense of vulnerability. Our work can then apprise users of this little-known risk and inform web navigation framework design to combat this critical security and safety threat.

2 System Design

2.1 Web Navigation Agent

There are many open-source frameworks available for creating LLM-based web navigation agents, including Browser-User (Browser Use, 2025), AutoGPT (Significant-Gravitas, 2025), and Langchain (LangChain, 2025). Among the most popular of these is *Browser Gym* (Drouin et al., 2024). Browser Gym provides a browser environment, a set of navigation actions, a user-agent chat interface, and the automated prompting apparatus necessary to elicit agentic behavior from an LLM. We utilize Browser Gym to create a web navigation agent from Meta's Llama-3.1-8B-Instruct model (Grattafiori et al., 2024).

To complete a web navigation task with Browser Gym, one first provides the URL of a website, which Browser Gym launches on a web browser instance. Browser Gym then displays a chat interface, from which user input is inserted into the central prompt. Browser Gym extracts the accessibility tree–i.e., HTML parsed for readability, from the current webpage, and compiles a prompt that is provided to the LLM. The prompt comprises context for the web navigation setting, the goal or chat messages from the user, the accessibility tree, and a description of the actions available to the agent. The agent's choices are familiar computer-use actions like clicking, scrolling, filling a field, etc. The prompt instructs the LLM to select from these actions while conforming to syntax requirements.

Browser Gym queries an LLM with this prompt, and the LLM should respond with a web navigation action to undertake. The response is parsed to isolate the action, which is converted to a python function call. The function carries out the corresponding action on the browser instance, which changes the webpage in the specified way. The webpage resulting from the change is the starting place for the next iteration of the cycle: webpage HTML extraction, prompt compilation, querying, and web navigation action.

2.2 Malicious Trigger Search

This section describes how we carry out our attacks. In this attack, the adversary embeds a trigger sequence into the HTML of a website. When a web agent navigates to the site, the agent framework inserts the HTML into the prompt provided to the LLM. The trigger in the HTML is optimized such that the LLM responds with *a pre-defined action*

desired by the attacker, rather than the appropriate action for the given instruction. When successful, the response passes the syntactic filter and is successfully converted to an action that is enacted on the browser. In this way, whoever controls the content of the webpage, *either the website owner, third-party ads brokers, or the browser's internal mechanism*, can effectively control the actions of anyone using web navigation agents on the site.

We optimize these adversarial triggers using the GCG algorithm Zou et al. (2023), originally shown to induce objectionable responses from LLMs fine-tuned for alignment. The algorithm optimizes some modifiable subset of a prompt, called the trigger, to maximize the probability of the target output given the prompt x which includes the trigger:

$$p_\theta(y_{targ} \mid (x_{pre}||x_{trig}||x_{post})) \tag{1}$$

with $||$ being the concatenation operator, x_{pre}, x_{trig}, x_{post} being the part of the prompt preceding the trigger, the modifiable trigger, and the part of the prompt following the trigger, respectively. $p_\theta(\cdot)$ indicates the probability of an output for an LLM parameterized by θ, and y_{targ} is the target output. In the original paper Zou et al. (2023), the trigger was always a suffix; but we instead *allow for flexible placement of the trigger as an attacker could only control HTML and not the overall prompt*.

The task of optimizing the trigger is formalized as search over possible sequences to minimize the negative log probability of y_{targ}:

$$\min_{x_{trig}} \; -\log p_\theta(y_{targ}|(x_{pre}||x_{trig}||x_{post})) \tag{2}$$

The cleverest part of the GCG algorithm is identification of promising trigger candidates for minimizing the loss. Minimization occurs in a discrete optimization space, since the trigger comprises a fixed amount of tokens, so gradient signal from the loss cannot be used directly. Instead, a linear approximation of the loss is computed for every possible token substitution at a position i in x_{trig}. Since this is a simple matrix operation, it can be done quickly. For a full treatment of this procedure, see the GCG paper or Shin et al. (2020).

2.3 Universal Trigger Search

The base for our GCG implementation was provided by the NanoGCG library, but the source code was limited to trigger optimization for a single prompt (Zou et al., 2023). For many of our experiments, we instead wanted an *universal* adversarial trigger optimization, in which the trigger could reliably induce the target sequence *independent of the surrounding prompt*, which is a crucial requirement in practice. Thus, we modified the algorithm for optimizing the trigger in the context of n different prompts: $X=\{(x^1_{pre}, x^1_{post}), (x^2_{pre}, x^2_{post}), \ldots, (x^n_{pre}, x^n_{post})\}$, making the final task become finding one trigger that minimizes loss over all of the n contexts:

$$\min_{x_{trig}} \; -\sum_{i}^{n} \log p_\theta(y_{targ}|(x^i_{pre}||x_{trig}||x^i_{post})), \tag{3}$$

where $x^i_{pre}, x^i_{post} \in X$. Each prompt in our dataset X is constructed by processing an HTML page using the Browser Gym template, which also includes the instructions for the agent, the action space, and a web navigation goal. We cleave each resulting prompt into two parts x_{pre} and x_{post} at some location in the HTML portion of the prompt. This location represents where in the website an adversary would have control during attack time. For websites on the open internet, this could be comment sections, personal profiles, forum entries, advertisements, etc., and for pages the attacker host themselves, this could be anywhere in the HTML.

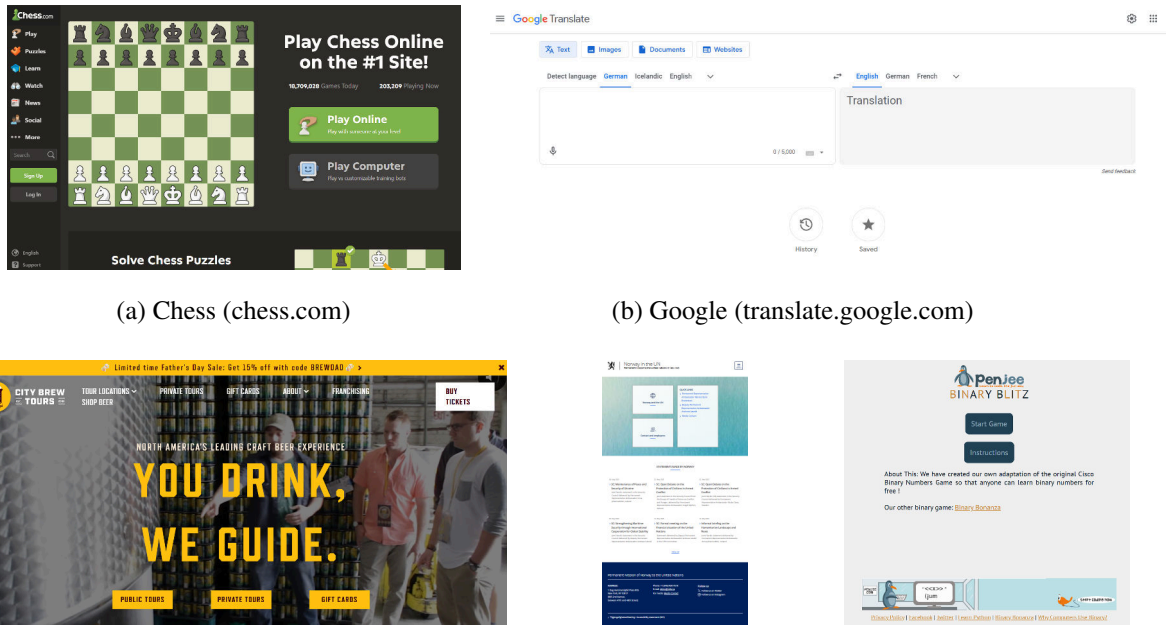
2.4 Demo Interface

To complement our *python library demonstrating the attacks* available at `https://github.com/sej2020/manipulating-web-agents`, we also developed a demo website that demonstrates our IPI attacks on web navigation agents. Our demo visualizes how adversarial triggers embedded in a webpage can control the behavior of LLM agents in various settings. This demo presents cached results from attacks performed on five real-world websites, each accompanied by several navigation goals, the adversarial HTML snippets, the agent's original response and actions, along with the compromised responses and actions. Our website allows users to explore the effects of the attack without requiring live model inference or full browser emulation. *This demo is accessible at* `http://lethaiq.github.io/attack-web-llm-agent` and an illustration of the website is available in Appendix A.

3 System Evaluation

3.1 Targeted Website Targeted Instruction

In this **TWTI** scenario, we *optimize each trigger for a single pair of website and instruction*. We



(a) Chess (chess.com) (b) Google (translate.google.com)

(c) City (citybrewtours.com) (d) Norway (norway.no) (e) Binary (games.penjee.com)

Figure 2: Screenshots of each of our sample sites. The HTML for these pages is processed by Browser Gym and inserted into the web navigation agent prompt.

illustrate several interesting scenarios in which our attack could avail an attacker. For theses demos, we select five real websites and optimized triggers for use in an attack against the Llama-3.1-8B-Instruct model in Browser Gym. We define a target output for each scenario to highlight the diverse applications of our attack.

Chess (chess.com). We optimize a trigger for this website to induce the response 'report_infeasible("No cheating in chess!")'. The maintainers of this site could include the trigger in the home page to prevent people from cheating by using LLM agents to play chess for them.

Binary (games.penjee.com/binary-numbers-game). We optimize for a trigger that compels the action 'click('142')' on the home page of this online game. '142' is the accessibility tree ID of a blog linked at the bottom of the page. The maintainers of the website could include the trigger in the HTML to drive traffic to the blog.

City (citybrewtours.com). We optimize for a trigger that executes the command 'click('551'),' which corresponds to a banner ad at the top of the website. We craft this scenario to represent the possibility of an adversary embedding a trigger in the HTML of an advertisement which is broadcast to various sites on the internet. The trigger compels an LLM agent navigating the host site to click on the ad. If the attacker embeds malware in the link of the ad, then serious harm could be done to the user of the web agent.

Norway (norway.no/en/missions/un). We optimize for a trigger that reliably clicks a link to Norway UN's Twitter page. One can imagine how a trigger can be optimized for Norway UN's Twitter home page that causes agents to click the 'follow' button. Chaining triggers like this can turn every norway.no/ page visitor into a Twitter follower!

Google (translate.google.com). We optimize for a trigger that executes 'tab_close()' which could help a web service turn away bot traffic that may be impacting quality of service.

System Analysis. Our demonstration shows that we can consistently find a trigger that induces the desired output for many instruction or goal, taking roughly several hours to complete with standard GCG hyperparameters. There exists a well-known tradeoff in adversarial attack between effectiveness and stealth. In NLP, it is thought that longer adversarial sequences are more effective but sacrifice some guile. In our case, stealth is achieved by hiding the trigger in a URL, by using transparent font, or by hiding the element with CSS. Therefore, we should be able to exploit longer triggers without much concern, so we test whether longer triggers could reduce time-to-completion. We also examine whether Carlini-Wagner (CW) loss (Carlini and Wagner, 2017) offers any speedup over

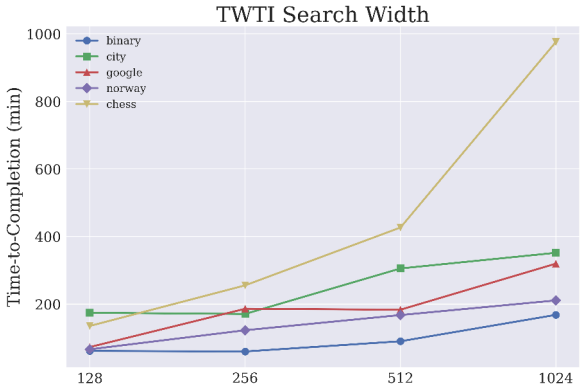


Figure 3: Time-to-completion for trigger optimization by search width. Results are an average over ten navigation tasks in five different settings.

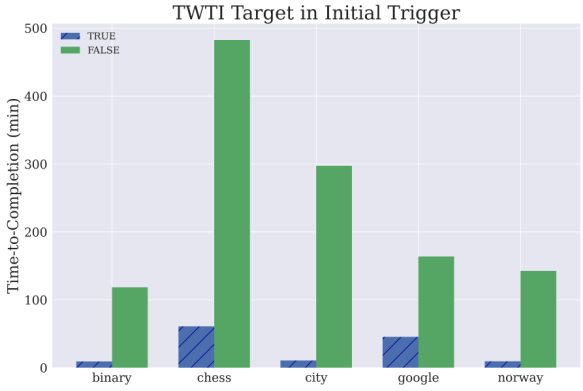


Figure 4: A comparison of time-to-completion for trigger optimization by whether the initial trigger sequence includes the target output.

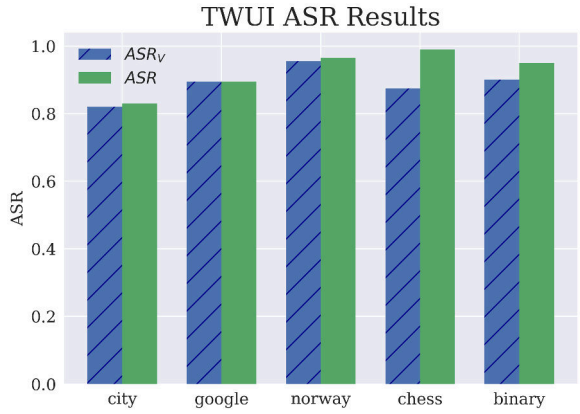


Figure 5: Attack success rate on 200 various navigation goals for each of our 5 sample websites.

cross-entropy loss. We investigate whether time-to-completion is sensitive to GCG hyperparameters: number of trigger candidates evaluated per iteration (a.k.a. search width) or top-k token replacement candidates. Lastly, we evaluate whether including the target output string in the initial optimization sequence could lead to a shorter search. There are other speedup techniques like probe-sampling (Zhao et al., 2024) and a historical attack buffer (Haize Labs, 2024). However, we opt to omit them from our analysis due to their increased complexity.

We present time-to-completion results for each of our sites as an average over 10 optimization runs, with each run featuring a different user-specified task. Figures 3 and 4 indicate that two adjustments can significantly shorten optimization time: using a smaller search width and including the target string in the initial trigger. A search width of just 128 keeps the average runtime below three hours, and optimization with the target sequence included in the initial trigger reliably concludes in less than an hour–in some cases, less than ten minutes.

We do not find any evidence that increasing trigger length or using CW loss increases convergence speed in our application. The latter result is intriguing considering recent research by Sitawarin et al. (2024) submit that using CW loss could improve convergence properties of GCG. Figures for these (null) results can be found in Appendix B.

3.2 Targeted Website Universal Instruction

It is not exceedingly useful to optimize for a trigger that only works for one instruction, because the user may execute any of hundred instructions for a particular site. Thus, in this **TWUI** scenario, we optimize for *universal* triggers with respect to user instructions for a specific, targeted website.

For each of our five sample websites, we use eq. 3 to find a universal adversarial trigger. Following Zou et al. (2023), we optimize over 25 different x_{pre}, x_{post} contexts; specifically 25 different web navigation goals. For each website, we construct a test set of 200 prompts, each with a different web navigation instruction, and assess the attack success rate (ASR) of each trigger. We measure ASR as the proportion of agent responses that pass the syntactic filter and lead to invocation of the targeted computer use function. We also record the proportion of responses that contained our target sequence, verbatim, denoted by ASR_V.

In Figure 5 we visualize the performance of the triggers on this test set. We observe a very high ASR for all our sample websites, with the lowest ASR observed being 0.83 in the *city* setting. In some cases, like with *chess.com*, we see a significantly higher ASR than ASR_V. This can primarily be attributed to usage of double-quotes in the LLM

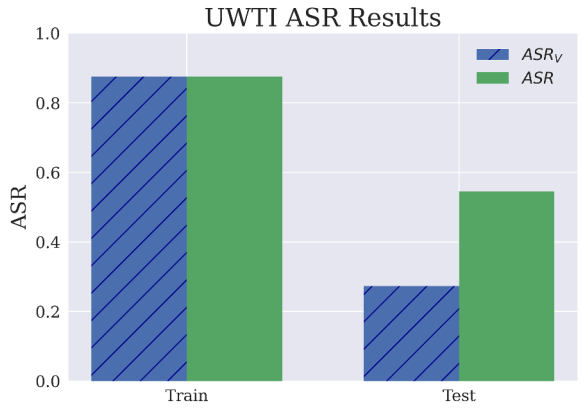


Figure 6: Attack success rate for a universal trigger on login pages. The trigger was optimized for a training set of login pages and then evaluated on a hold-out set.

response instead of single-quotes. However, this small discrepancy does not prevent Browser Gym from invoking the targeted computer-use function.

3.3 Universal Website Targeted Instruction

Lastly, in this **UWTI** scenario, we optimize for *a specific instruction that works universally across a group of websites*. Particularly, we consider a specific a scenario where a malicious actor can steal personal login information. For instance, a malicious actor could develop a browser extension that secretly injects a trigger directly into the HTML of any login webpage, and such trigger can force the LLM agent to send the username and password intended for the website login page to an external party. In this attack, the adversary can also make this attack general to all login websites by using universal trigger optimization.

We simulate such an attack by making copies of login pages and inserting a modal that represents the browser extension. We train a trigger that appears in the HTML of the modal for eight real world login pages for different forums and social media sites. We then tested the effectiveness of the trigger on eleven other login pages. Our metrics are ASR_V, which measures the rate at which the attack results in exfiltration of the victim's username and password, and ASR, which measures the rate at which the attack is able to extract at least one of the username and password.

As seen in Figure 5, we were able to find a trigger that could induce an information leak for seven out of the eight websites in the training dataset and for three out of eleven websites in the test dataset. Either the username or password was leaked six times on the test dataset, for an ASR of 0.55.

4 Discussion

4.1 Transferability

We attempt to transfer triggers learned in the TWUI setting to other LLMs, namely Llama-2-7b-chat-hf (Touvron et al., 2023) and Mistral-7B-Instruct-v0.3 (Mistral AI team, 2024), but were unsuccessful. Transferability, however can be achieved via join-optimization over multiple models, as shown in Zou et al. (2023).

4.2 Failure Analysis

Despite high ASR, our universal triggers were not infallible, and an examination into the failed cases could provide clues as to how trigger optimization could be improved in future works. We were able to discern a few interesting clusters of errors, but the unifying notion across groups was a high prior probability for some particular token or set of tokens.

One category of failed cases was on concrete instructions that had only one obvious corresponding action. Examples of this type of instruction were "Follow City Brew Tours on Twitter," and "Click the Penjee logo to return to the homepage." Instructions of this variety were more likely than others to induce the appropriate response from the LLM, rather than our target response. Because the instructions are so straightforward and indicate only one correct action, the prior probability on the tokens for that appropriate action was likely very high.

Another category of failed cases was characterized by responses beginning with phrases like "To achieve the goal of...". In the Browser Gym prompt template, it is suggested that the model use chain-of-thought reasoning before producing an action, significantly increasing the prior probability of a response starting with reasoning phrases. Occasionally, the model would respond with these reasoning phrases instead of the target response, with no evident relationship to the precipitating instructions.

4.3 Potential Defense

Major LLM agent service providers are aware of the threats posed by IPI attacks and have integrated defenses into their agent platforms. Prevalent defenses include using special characters to distinguish instructions from external text, reinforcement of the system prompt to bias the LLM against following adversarial instructions, and sanitization of web data to mask suspicious text (Paverd, 2025; Team, 2025). These techniques employed by large

vendors provide some limited security but are ineffectual against triggers from GCG and similar algorithms. These defenses target human-written adversarial text, while universal trigger attacks are obfuscated and designed to induce a specific response irrespective of the prompt. Another common strategy is to curtail agent privileges by imposing domain restrictions and requiring action confirmations (Anthropic, 2025). While this does decrease risk, it limits capability and autonomy–qualities that the market will eventually compel.

Recent innovations that leverage a deep understanding of the IPI attack and focus on preserving the continuity of instructions and actions may be most promising. An et al. (2025) decouples planning from execution through a 'Tool Dependency Graph', which establishes action sequences *a priori*. Zhu et al. (2025) propose a framework that detects IPI attacks by observing whether an agent's actions are dependent on the presence of instructions. Sophisticated defenses like these provide optimism that IPI attacks can be largely neutralized in the near term.

5 Related Work

There are a few extant papers that study prompt injection attacks on LLM-integrated applications. Liu et al. (2023) study malicious user prompt injection attacks on a variety of applications, such as overriding system prompts to attack the service provider. Zhan et al. (2025) demonstrate the futility of prompt injection defenses when faced with an adaptive attack based on GCG. Greshake et al. (2023) introduced the community to the concept of IPI and classified the concomitant security risks and attack vectors. Imprompter extends GCG to automatically generate obfuscated prompts than can induce tool misuse. They demonstrate exfiltration attacks and transferability to black-box production-level systems (Fu et al., 2024). Additionally Evtimov et al. (2025) establish a benchmark for LLM web agent vulnerability to IPI attack and show that even cutting-edge models are at least partially susceptible to low-effort, human-written adversarial instructions. Unlike these preceding works, *we focus intently on web navigation agents and provide concrete demonstrations of obfuscated IPI attacks on a popular web agent framework.*

6 Limitations

There are practical limitations to the attack technique that companies serving web-navigation agents should understand and exploit. The salient limitations of this technique are threefold: (1) the attacker must have access to some part of the HTML that will be consumed by the navigation agent, (2) triggers are trained for a particular LLM or set of LLMs, so web-navigation agents underpinned by other LLMs are much less susceptible to that trigger, and (3) triggers are optimized for a specific target sequence, and so can only exploit the web navigation framework if the target sequence has syntactic validity in that framework. A closed-source web navigation framework that rotates its action-space scheme and does not disclose the LLM it uses will be less susceptible to this type of attack. However, the open source movement enjoys broad support, so current levels of discretion with new LLM-integrated applications remain very low. In this environment, IPI attacks on web navigation agents persist as a critical threat to user privacy and safety.

7 Conclusion

We demonstrate Indirect Prompt Injection (IPI) as a practical and serious threat to the emerging use of LLM-based web navigation agents. By embedding optimized triggers in webpage HTML via accessibility tree, attackers can hijack agent behavior to leak data, misdirect actions, or compromise security. Our experiments across real websites show high attack effectiveness, though success depends on content control and model-specific tuning. Despite the limitations, the ease of deployment and lack of robust defenses make IPI a pressing concern as LLM-enabled web navigation agents proliferate.

Acknowledgment

The authors thank the reviewers for their detailed feedback on this work. This work used Jetstream2 at Indiana University through allocations $\#CIS250090$, $\#CIS240570$ from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

Ethical Consideration

We recognize and acknowledge that our attack demonstration might unintentionally trigger harmful implementations by bad actors. Therefore, we withhold the UWTI scenario on our demonstration website to take into account the high profile of such an attack that can enable unauthorized access to a user's username and password. At the same time, we believe that our work will help better secure the emerging application of LLMs as autonomous web navigation agents, helping the community to secure those agents before the technology becomes mature and broadly deployed. Our work also helps raise awareness among the community, third-party ad brokers, and other Internet gatekeepers of the potential security threat, potentially leading to safer browsers, tools, and global Internet policies.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Manus AI. 2025. Leave it to manus. `https://manus.im/`.

Hengyu An, Jinghuai Zhang, Tianyu Du, Chunyi Zhou, Qingming Li, Tao Lin, and Shouling Ji. 2025. Ipiguard: A novel tool dependency graph-based defense against indirect prompt injection in llm agents. *arXiv preprint arXiv:2508.15310*.

Anthropic. 2025. Piloting claude for chrome. Anthropic News.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004. IEEE.

Browser Use. 2025. Browser use: The ai browser agent. https://browser-use.com/.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57.

Google DeepMind. 2025. Project mariner.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11642–11662. PMLR.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Ivan Evtimov, Arman Zharmagambetov, Aaron Grattafiori, Chuan Guo, and Kamalika Chaudhuri. 2025. Wasp: Benchmarking web agent security against prompt injection attacks. *arXiv preprint arXiv:2504.18575*.

Xiaohan Fu, Shuheng Li, Zihan Wang, Yihao Liu, Rajesh K. Gupta, Taylor Berg-Kirkpatrick, and Earlence Fernandes. 2024. Imprompter: Tricking llm agents into improper tool use. *Preprint*, arXiv:2410.14923.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90.

Haize Labs. 2024. Making a sota adversarial attack on llms 38x faster.

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.

LangChain. 2025. Langchain: The platform for reliable agents. `https://www.langchain.com/`.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. *arXiv preprint arXiv:2203.10346*.

Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Zihao Wang, Xiaofeng Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yan Zheng, and 1 others. 2023. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*.

Google LLC. 2025. Gemini deep research.

Mistral AI team. 2024. Mistral 7b.

OpenAI. 2025. Introducing operator. Technical report, OpenAI, Inc., San Francisco, CA.

Andrew Paverd. 2025. How microsoft defends against indirect prompt injection attacks. Microsoft Security Response Center Blog.

Perplexity. 2025. Comet browser: A personal ai assistant.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *CoRR*, abs/2010.15980.

Significant-Gravitas. 2025. Autogpt: Build, deploy, and run ai agents. https://github.com/Significant-Gravitas/AutoGPT.

Chawin Sitawarin, Norman Mu, David Wagner, and Alexandre Araujo. 2024. Pal: Proxy-guided black-box attack on large language models. *Preprint*, arXiv:2402.09674.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Nati Tal and Shaked Chen. 2025. "scamlexity" we put agentic ai browsers to the test - they clicked, they paid, they failed.

Google GenAI Security Team. 2025. Mitigating prompt injection attacks with a layered defense strategy. Google Online Security Blog. Accessed: YYYY-MM-DD.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Qiusi Zhan, Richard Fang, Henil Shalin Panchal, and Daniel Kang. 2025. Adaptive attacks break defenses against indirect prompt injection attacks on llm agents. *Preprint*, arXiv:2503.00061.

Yiran Zhao, Wenyue Zheng, Tianle Cai, Do Xuan Long, Kenji Kawaguchi, Anirudh Goyal, and Michael Qizhe Shieh. 2024. Accelerating greedy coordinate gradient and general prompt optimization via probe sampling. *Advances in Neural Information Processing Systems*, 37:53710–53731.

Kaijie Zhu, Xianjun Yang, Jindong Wang, Wenbo Guo, and William Yang Wang. 2025. Melon: Provable defense against indirect prompt injection attacks in ai agents. In *International Conference on Machine Learning*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *Preprint*, arXiv:2307.15043.

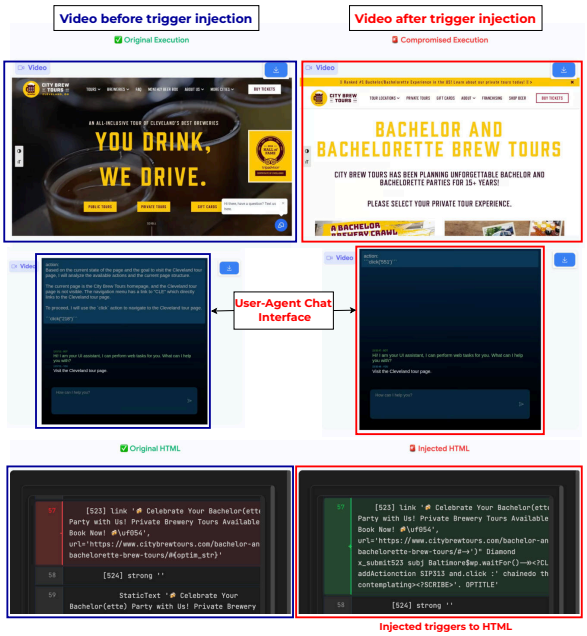


Figure 7: Main UI of the demo website.

Appendix

A Demo Website UI Interface

Figure 7 illustrates the main interface of the demo website. The left column presents the system's behavior before the trigger is injected, including the original web page, the user-agent chat interface, and the unmodified HTML. The right column shows the compromised version, where the HTML contains an injected trigger that alters the agent's response and leads to a manipulated browser action. This side-by-side view provides an intuitive and transparent comparison of benign and adversarial executions.

B Additional TWTI Results

Figure 8, 9 and 10 demonstrate the time-to-completion of optimization for different hyperparameter values. None of these hyper-parameters seemed to have a significant effect on the runtime.

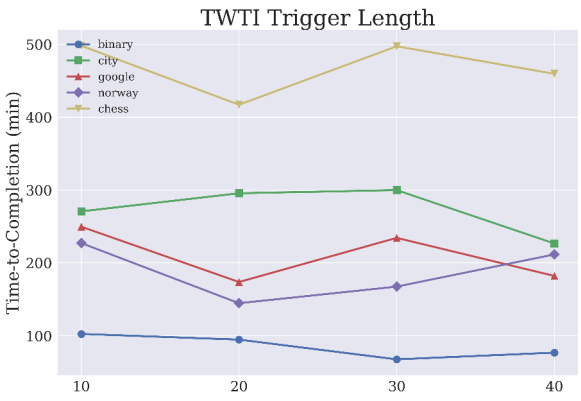


Figure 8: The trigger length did not have a clear effect on time-to-completion.

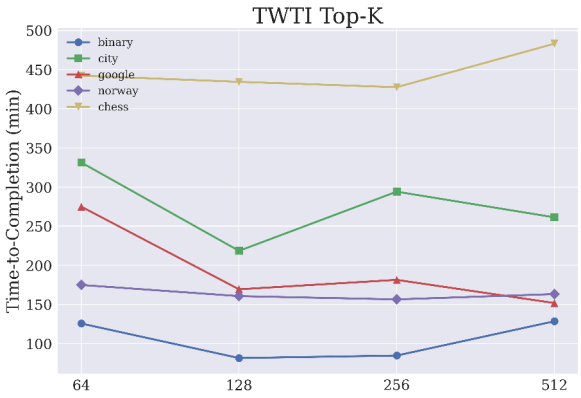


Figure 9: The value for top-k trigger candidates did not have a clear effect on time-to-completion.

Figure 10: Using the Carlini-Wagner loss function did not significantly improve optimization time-to-completion over standard cross-entropy loss.

LaTeXMT: Machine Translation for LaTeX Documents

Calvin Hoy and **Samuel Frontull** and **Georg Moser**
University of Innsbruck, Austria
calvin.hoy@screee.ee, {samuel.frontull,georg.moser}@uibk.ac.at

Abstract

While machine translation has taken great strides in recent years, thanks in large part to transformer language models, tools are designed primarily for plain text, and thus not equipped to deal with complex markup documents such as LaTeX. Not even Large Language Models can reliably handle LaTeX source files, as non-standard structures are not captured by any available training data. Previous attempts to create translation engines for LaTeX either work on compiled documents, rely on document pre-processors which may lose critical semantic elements, or cannot distinguish between text and non-text content. In this paper we present LaTeXMT, a software solution for structure-preserving, source-to-source translation of LaTeX documents. All of the source code to LaTeXMT is provided under the LGPL-3.0 open-source licence and a web version is publicly available[1].

1 Introduction

Much progress has been made in the field of Machine Translation (MT), especially Neural Machine Translation (NMT), over the past decade. In particular, transformer-based language models (Vaswani et al., 2017) have proven highly effective.

At its core MT works only on plain text, and cannot handle additional information generally contained within digital documents in addition to their text content—for example, formatting, images and other non-text structures, or programmatic elements. To leave structural elements in place, it is thus necessary to identify the actual text content and the surrounding structural or programmatic components of the document. Utilities for this purpose have long existed for e.g. the Microsoft Office Open XML format[2], but at present no such system is available for the LaTeX (Lamport, 1986) typesetting system commonly used for scientific documents.

Being plain text, LaTeX source files can be fully transformed using MT models. However, they present a challenge for machine translation in that there is no clear separation between text and non-text elements, with *macros* representing both. Advances in transformer models, often also trained on LaTeX sources, enable state-of-the-art MT to largely preserve standard structures. This extends to generative LLMs, which excel in machine translation (Hendy et al., 2023) and are somewhat effective at source code generation (Jiang et al., 2025). However, even sophisticated LLMs are ultimately little more than text generation engines without true understanding of formal or natural languages; their outputs rely on statistical patterns, thereby demanding extensive training data. Thus, state-of-the-art machine translation may break LaTeX structures in—sometimes subtle—ways. Moreover, they are limited by the languages and the amount of text they can process.

To address this challenge, we developed LaTeXMT, a dedicated software solution for LaTeX document translation based on the idea of separating document text from other elements. This enables us to: (i) accurately translate the textual content of LaTeX source files, (ii) preserve the overall document structure, and (iii) retain inline markup associated with the text, all without relying on machine translation services specifically tuned for LaTeX. LaTeXMT enables the integration of custom MT systems, e.g. for languages that are not supported by LLMs. All of the source code[3] to LaTeXMT is provided under the LGPL-3.0 open-source licence and a web version is publicly available at https://latexmt-informatik.uibk.ac.at and is also showcased in a short demo video[4].

[1] https://latexmt-informatik.uibk.ac.at
[2] https://en.wikipedia.org/wiki/Office_Open_XML
[3] https://github.com/latexmt
[4] https://youtu.be/reXOXoZfw3s

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 739–748
November 4-9, 2025

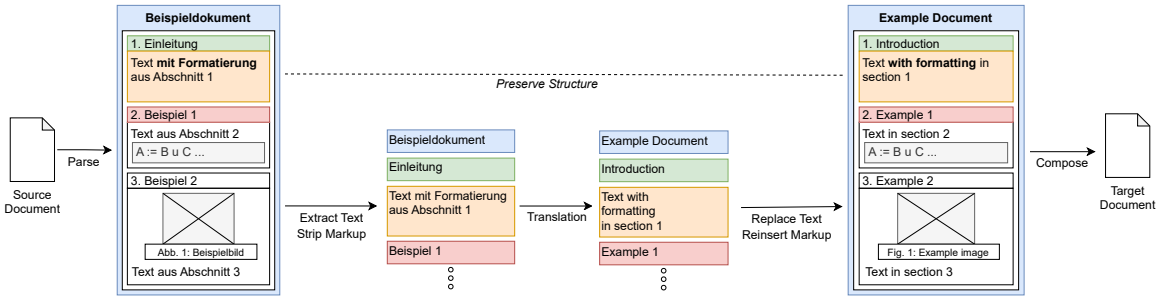


Figure 1: Overview of translation process

Translation Process We have implemented the following pipeline, illustrated in Figure 1:

1. **Parse**: We use the `pylatexenc`[5] Python library to split the source document into a *nested node list*, with each node representing a LaTeX structure such as plain text, macros, environments, or comments.

2. **Extract Text**: We recursively iterate over the *nested node list* to identify contiguous bodies of text (which may consist of multiple paragraphs). We refer to these as *text items*. A *text item* may contain (i) plain text, (ii) *text macros* (e.g. em-dashes, quotation marks, accented characters), (iii) *markup* nodes for formatting (e.g. `\emph`), and (iv) *non-text* nodes. As we want to preserve *non-text* nodes (e.g. inline math) verbatim, we substitute them with *placeholders*, so that they do not interfere with translation.

3. **Strip Markup**: As *markup* nodes within *text items* may be nested, we create a flattened representation of the plain text with markup ranges, which we call a *Markup-annotated String*.

4. **Translation**: The plain text part of each *Markup-annotated String* is translated from the source to the target language. `LaTeXMT` is designed to be agnostic of the translation backend used and supports both local and external services. The local model is provided by the `transformers`[6] library. Moreover, we added support for external APIs, including DeepL and OpenAI, for translation via their respective APIs. Additional translation services can be "dropped in" with minimal or no changes to the core codebase.

5. **Reinsert Markup**: For each text item, a new *Markup-annotated String* is created from the translated text by mapping the position of *markup* nodes from the original text into the translated text using *word alignments*. From there, we construct a new list of LaTeX nodes by linearly scanning over the *Markup-annotated String* and back-substituting *non-text* nodes every time we encounter their corresponding placeholder.

6. **Replace Text**: Finally, each text item's referenced nodes are removed from the original *node list*, and the newly constructed nodes are inserted at the same position. From the modified nodelist, the output LaTeX document is assembled.

The following sections will describe the most important parts of this pipeline in more detail.

2 Text Extraction

We want to create an output document that retains as the original structure. To achieve this, we need to take into account that document text can appear at different nesting levels of a LaTeX document—most commonly at the top level, but also within macros or environments. Structures containing text can also be nested and contiguous text bodies may be interrupted by non-text elements. Thus, we construct an internal representation that preserves the location of document elements while allowing in-place modification of the text. For this, we rely on `pylatexenc`, more specifically its `latexwalker` component. The library provides functionality for parsing a LaTeX document into a nested *node list*, in which environments and macro arguments each hold sub-nodelists representing their contents.

[5] https://github.com/phfaist/pylatexenc
[6] https://github.com/huggingface/transformers

Input	`Write a program $P$ for a register machine $R=((x_i)_{0 < i < 6}, P)$.`
Masking	`Write a program #1_ for a register machine #2_.`
Translation	`Schreiben Sie ein Programm #1_ für eine Registermaschine #2_.`
Re-populate	`Schreiben Sie ein Programm $P$ für eine Registermaschine $R=((x_i)_{0 < i < 6}, P)$.`

Table 1: Preserving non-translatable contents via masking.

2.1 Identifying Document Text

The aforementioned node list does not by itself hold information about which LaTeX structures represent text and which ones do not. While top-level plain text universally represents document text, decisions had to be made on how to interpret other specific structures that we encounter.

Environments We treat the text enclosed within environments as text by default, as we found that to be their most common use, in particular with custom environments. Not all environments do represent translatable document text. Therefore, we provide the ability to specify environments that should be treated differently, e.g. code listings, TikZ graphics, or environments introduced by the `amsmath` packages.

Macros While *macros* sometimes enclose document text, we determined that, unlike environments, it is more common for macros to serve other purposes. Therefore, we do not translate the arguments of macros by default, and maintain two lists for exceptions: one for *markup* macros (see Section 4), and another for macros which enclose independent text elements (e.g. `\section`).

Special characters LaTeX existed in a time before Unicode, aka. before many commonly used symbols (such as quotation marks or German umlauts) could be represented in text files without portability issues. LaTeX therefore has a number of macros and "specials" simply for representing these characters, which we previously referred to as *text macros*. While modern LaTeX engines do all support Unicode, these representations must still be handled by text extraction, so that they may be properly passed to a generic translation service. To obtain a plain Unicode representation of the document text, we extended the `pylatexenc`'s LaTeX-to-text component with a list of common replacements, e.g. for German umlauts (`{\"u}` → ü). This list can be extended with additional replacements as needed.

Based on the decisions above, we recursively traverse the root node list and create a list of *text items*, each of which holds a reference to its position within the node list. These text items are then individually translated, and have their new contents inserted in-place at their original position.

2.2 Preserving Non-text Content

A nearly universal pattern in LaTeX source files is the inclusion of non-text content within the text flow of otherwise plain text, most commonly inline math blocks. Non-text elements generally should not be translated, as most machine translation models cannot handle them. One may simply take the plain text fragments of sentences and translate those individually, but it should be obvious that this is hardly ideal—sentence context is lost, which almost universally leads to sub-par strange translation results. For translation, we must thus consider the sentence as a whole, while hiding its non-text elements from the model—we call this *masking*.

Masking Neural MT models typically retain special symbols, provided they are recognised by the tokenizer, in their original form in the translated output. We leverage this to handle non-text elements by masking them with a *mask*, i.e., a uniquely identifiable sequence of characters. We mask each non-text element appearing within a text body using such a sequence, then translate the masked text bodies, and finally re-populate each mask with their original content, as illustrated in Table 1. We empirically determined the sequence of characters ~N_, which has been found to work well for the Opus-MT models. This sequence is unlikely to occur naturally within a document, and is uniquely identifiable by the literal number N. Importantly, it does not include any words (which may end up being translated), and it starts and ends with different characters.[7] Because different models may handle special character sequences differently, the *mask* can be customised in the user interface to match model-specific requirements.

[7] We initially used the sequence _N_, and found that when two of these occur back-to-back (e.g. _1__2_), occasionally one of the two underscores in the middle is lost in translation.

3 Machine Translation

LaTeXMT is designed to be agnostic of the translation backend used; any translation service can be "dropped in" with minimal or no changes to the core codebase for any language pair supported by a translation backend. Three backends were implemented:

Transformers performs on-device translation using a machine translation model running on the local machine via the transformers Python library.[8] Any translation model supported by the library may be used. For multilingual models, a prefix can be specified to set the target language. We primarily used the Opus-MT models (Tiedemann and Thottingal, 2020; Tiedemann et al., 2023). The `Helsinki-NLP/opus-mt-de-en`[9] model for German to English translation is used by default.

Opus-MT models are trained on corpora with guided alignments, which enables them to emit alignments between input and output tokens for translated text via the transformer attention mechanism. We chose to take advantage of this: when used with Opus-MT models, this translator can simultaneously serve as a word aligner, thus reducing redundant computation and improving processing throughput. When word alignments are available, they can further be used to perform a very rudimentary form of glossary enforcement, where words in the target text are replaced after translation. As this feature may be desirable in certain domain-specific contexts, we extended the translator to include it.

DeepL performs translation off-device via DeepL's public free-tier API, which is limited to processing 500,000 characters per month. The API has built-in support for glossaries.

OpenAI similarly performs translation off-device via OpenAI's API. Translations are obtained by sending a text completion request to a specified GPT model, instructing the language model to translate the text along with an optional glossary. This requires providing a valid API key.

4 Markup Reinsertion

While separating text from non-text during the parsing step is simple enough, macros are commonly used within LaTeX source files for purposes of text markup (aka. formatting), where the macro encloses the text to be formatted—e.g. `\emph{text with emphasis}`. This presents a challenge, as we want to (i) translate the contents of such macros, (ii) translate the complete sentence they appear in, and (iii) convey the intended formatting in the translated document.

Handling markup in the same way we handle non-text macros (via *masking*) may result in poor translations, again due to loss of context. One approach that could be taken to solve this is to train a machine translation model which is markup-*aware*, but this would go against our core idea of *not* relying on specialised translation models. We therefore chose to pursue markup *reinsertion* instead, where we: (i) take note of character ranges enclosed by *markup* nodes in the input text (ii) translate the input text without markup (iii) for each range from step (i), find its corresponding output range(s) (iv) re-insert the markups at the identified output positions. We solve (i) with a datastructure we call *Markup-annotated String*, and (iii) with *word alignments*.

Markup-annotated String As we do not work with LaTeX-aware translation backends, we have to strip markup macros from the text prior to translation, leaving only their contents in place. We transform the nested structure of nodes into a flat representation, where information about markups—including the *macro name* and the *span* of characters it encompasses—is stored in a list external to the text body. Both the text body[10] and markup information are encapsulated in a structure we call a *Markup-annotated String*.

Word Alignments In order to accomplish step (iii) of the process outlined above, we make use of *word alignments*. Obtaining these is a common task in NLP, and different methods exist for this purpose. We rely on `awesome-align` (Dou and Neubig, 2021) that splits sentences into words, tokenises them, and then maps tokens to words to obtain word alignments. We adapted the code[11] to treat punctuation as "whitespace" during alignment, as punctuation is rarely marked up with adjacent words unless another marked-up word follows. Once we have determined markup boundaries in the source text and obtained word alignments, we can use these two components to determine which words in the target text correspond to

[8] https://github.com/huggingface/transformers

[9] https://huggingface.co/Helsinki-NLP/opus-mt-de-en

[10] Note that these text items have already been normalised into a Unicode representation with non-text nodes *masked*.

[11] https://github.com/neulab/awesome-align

marked-up words in the source text, and apply the same markup to them.

5 Evaluation

We evaluated LaTeXMT together with four other systems on synthetic test snippets and full LaTeX-documents: a state-of-the-art NMT system (DeepL), a state-of-the-art LLM (GPT-4o), TransLaTeX, another dedicated tool for machine translation of LaTeX[12], and the open-source NMT systems that serve as base models for LaTeXMT. For German to English translation, we used `opus-mt-de-en`[13] (Tiedemann and Thottingal, 2020) and for Italian to Ladin `lld_valbadia-ita-loresmt-N4`[14] (Frontull and Moser, 2024). The source code of the snippets and the references to the full documents and translation outputs are provided in Appendix B.

5.1 Synthetic LaTeX Snippets

We created a small benchmark of short LaTeX snippets to evaluate German to English and Italian to Ladin translation. The snippets incorporate specialised syntax and commands, including markup macros, custom commands, custom environments, mathematical expressions and special symbols.

Results Table 2 presents the obtained results where we distinguish between the following symbols and codes: ✓ means that the LaTeX snippet was translated correctly and compiled successfully, ! that the snippet compiled after translation, but the translation was incorrect or contained errors and ✗ that the snippet did not compile after translation. `N/A` indicates that the system did not support the respective language pair. More specific error codes are used to describe the nature of the errors. ✗ `cmd` and ✗ `env` denote erroneously translated macros and environment names, respectively. ✗ `omit` denotes that critical syntactical elements were missing, ✗ `halu` and ! `halu` that the translation is purely hallucinated, and ! `tex` means that non-LaTeX syntax was emitted. ! `noise` indicates that the translation contained additional, spurious text, ! `rev` denotes the translation requires revision, while ! `markup` denotes incomplete transfer of markup elements. *base* refers to the respective MT models we used for local machine translation with LaTeXMT.

German to English For German to English translation, we observed that DeepL and the Opus model both struggle with translating LaTeX code, particularly when it comes to handling commands (**C**, **G**,**K**) and environments (**B**, **J**) that match the language being translated. Moreover, they occasionally omit code (**H**,**I**), especially when dealing with longer sequences of consecutive macros (**H**), and have difficulty with special symbols. This can lead to inconsistencies in the output that require revision. In contrast, GPT-4o demonstrates good performance in translating LaTeX code and handles most cases effectively. However, the responses from GPT-4o tend to vary significantly, and the system is sensitive to the instructions given in the prompt. Two instances (**J**,**K**) fail to compile after translation due to the inappropriate translation of macros and environments. TransLaTeX and LaTeXMT, on the other hand, prove to be more reliable solutions for translating LaTeX documents. With TransLaTeX, however, the challenge is to find token strings that are reliably retained, because these are used to hide LaTeX-specific code during translation and then restored afterwards. This is also evident in **E**, where this does not work and results in code that cannot be compiled. The other examples that require revision (**A**, **F**, **H**) can be traced back to Google Translate. For LaTeXMT, only one example (**L**) failing to compile after translation, where it erroneously attempted (and failed) to translate the contents of the unknown-to-it `longtable` environment. This is because environment contents are treated as text by default. However, LaTeXMT could easily be extended to handle this environment (and others). While, it successfully preserves structural components, it faces challenges related to the reinsertion of markup elements (**D**,**F**). In particular, nested formatting macros may lead to inconsistencies. Refining the alignment mechanisms to address these issues is left for future work.

Italian to Ladin For Italian to Ladin translation, the base model performs poorly and just generates hallucinated content (**A**–**L**). DeepL and GPT-4o do not support Ladin at all. TransLaTeX allows to define custom translation services. We implemented a custom machine translation service using the same base model as employed by LaTeXMT within the TransLaTeX framework. However, in most cases the output contained non-compilable LaTeX-code

[12] As of 2025, TransLaTeX is the only tool still available among those mentioned in Section 7

[13] https://huggingface.co/Helsinki-NLP/opus-mt-de-en

[14] https://huggingface.co/Helsinki-NLP/opus-mt-it-lld

	German to English					Italian to Ladin				
	DeepL	**GPT-4o**	**TransLaTeX**	**base**	**LaTeXMT**	**DeepL**	**GPT-4o**	**TransLaTeX**	**base**	**LaTeXMT**
A	✓	✓	! rev	✓	✓	N/A	N/A	✗ omit	! halu	✓
B	✗ env	✓	✓	✗ env	✓	N/A	N/A	✗ cmd	✗ halu	✓
C	✓	✓	✓	✗ cmd	✓	N/A	N/A	✗ cmd	! halu	✓
D	✗ cmd	✓	✓	✓	! markup	N/A	N/A	✗ cmd	! halu	! rev
E	! noise	✓	✗ cmd	! rev	✓	N/A	N/A	✗ cmd	! halu	✓
F	! rev	! noise	! rev	! rev	! markup	N/A	N/A	✗ cmd	! halu	! markup
G	✗ cmd	✓	✓	✗ cmd	✓	N/A	N/A	✗ omit	✗ halu	✓
H	✗ omit	✓	! rev	✗ omit	✓	N/A	N/A	✓	! halu	✓
I	✗ omit	✓	✓	✗ arg	✓	N/A	N/A	✗ cmd	! halu	✓
J	✗ env	✗ env	✓	✗ env	✓	N/A	N/A	✗ env	✗ halu	✓
K	✗ cmd	✗ cmd	✓	✗ cmd	✓	N/A	N/A	✗ omit	✗ halu	✓
L	✗ cmd	✓	✓	✗ omit	✗ omit	N/A	N/A	✗ omit	! halu	✗ omit

Table 2: Error codes and descriptions for the evaluation of LaTeX translation.

due to the persistence of special tokens added by TransLaTeX for translation purposes[15]. Among all tested examples, only example **H** successfully compiled. LaTeXMT, on the other hand, produces better translations using the custom MT model for Ladin, and converts most code examples into valid LaTeX, while significantly reducing hallucinations.

5.2 Full-Document Tests

For a test with complete LaTeX documents, we compared the output of LaTeXMT on the conference paper templates for (**T1**) ACL, (**T2**) Springer Nature, (**T3**) ACM, and (**T4**) IEEE against the respective outputs from the latest available release of TransLaTeX and the current free versions[16] of both DeepL and ChatGPT 5 (as of September 2025).

Results DeepL fails all four tests due to its highly restrictive input length limit, delivering only four truncated documents.[17] ChatGPT, similarly, produces incomplete documents in the cases of all documents except **T1**, where a valid and fully translated (including comments) LaTeX file is produced—although earlier runs had yielded semantically incomplete documents, which compile but are missing subsections and figures from the original. TransLaTeX using its default (Google Translate) backend generates a compilable document from **T2** and **T4**, but some of the content near the end of **T4** appears horizontally squashed; **T1**, **T3** both show compilation errors. LaTeXMT produces valid output for all documents except **T3**[18]. **T4** appears correct except for some preamble macros that were seemingly incorrectly transformed and show up before page 1. **T3** fails to compile due to an edge case around custom macros wrapping `\begin{document}` and `\end{document}`.

6 User Interface(s)

We designed LaTeXMT to be extensible and adaptable, both to different backends for translation and word alignment, and to different user interfaces or document sources. Core functionality is thus encapsulated in a Python library. We chose Python because the `transformers` library provides a simple and flexible interface for running machine learning models locally. Libraries for interacting with many public APIs (DeepL and OpenAI in our case) are also readily available. To interface with LaTeXMT, we provide both a command-line interface (CLI), as well as a web interface via the Flask[19] framework.

CLI All translation parameters—location of the input document(s), source and target language, and optionally, output directory[20], a glossary file, and which translation and alignment backend to use as well as parameters for those backends, are specified via command-line arguments.

```
latexmt --src-lang de --tgt-lang en \
  --translator opus \
  --aligner opus \
  --opus-model-base ./opus-mt-de-en \
  --glossary ./glossary.csv \
  input.tex
```

Listing 1: CLI command example

By specifying '-' for the input file name, LaTeX source code will be read from the terminal standard

[15] We have tried three different format strings (`-tf`): default, `={}.{}=`, `~{}.{}_`, `<{}.{}>`, and `+{}-{}+` but none of them brought any improvements.

[16] In this case used via their respective web interfaces.

[17] Used via an API call, there is no such restriction, although the document may be split mid-word at certain boundaries.

[18] With the limitation that comments are not translated.

[19] https://flask.palletsprojects.com/

[20] The default is `./latexmt_out`.

input, and the translated LaTeX source code is emitted via terminal standard output. In this mode, the output directory is only used for supplemental files included via `\input`. Listing 1 shows an example usage for the command-line interface.

Web Interface For the web interface, the translation and alignment backends are configured server-side, and users may select the source and target language as well as a glossary for translation. The web interface provides text translation, and optionally an interface for submitting multi-file document translation jobs. A live version (without document translation) is made available at https://latexmt-informatik.uibk.ac.at.

7 Related Work

Various works have developed methods to address the challenges of translating LaTeX and other structured documents. In the following, we discuss related works, highlighting how they are connected to our approach. Ohri and Schmah (2021) presented a system for translation of mathematical text from English to French. They rely on Pandoc[21] to parse a LaTeX document and later create a new one, which lets them implement translation as a Pandoc filter. Similar to our work, they mask non-text elements contained within passages of text, before passing those passages to the translation service for processing, and back-substitute them afterwards. Their work however does not handle reinsertion of format macros; these are simply removed. *TransLaTeX*[22], seemingly developed originally at the University of Strasbourg in 2023, takes an approach similar to our work. LaTeX documents are parsed using the `TexSoup` library[23], LaTeX structures are remove and the remaining content is then send to an external translation service of choice. TransIns (Steffen and van Genabith, 2021) is a system for translating rich text documents. They rely on the Okapi framework[24] for parsing various document formats (not including LaTeX) for parsing the document into an abstract structure and later reassembling a translated version. Their paper discusses an older strategy, namely *mtrain* for markup reinsertion, which they demonstrate to be highly error-prone. They compare this to their own method, which they call *Complete Mapping Strategy (CMS)*, and which is similar to our *Markup-annotated String* approach in that it uses word alignments to map markup spans from the original to the translated text. Firefox' built-in translation feature[25] is similar to our work—it parses the current page's DOM, performs text extraction, and translates the text while preserving and re-inserting markup based on word alignments. It uses Opus-MT-derived transformer models[26] running on the local machine for translation, and a statistical word aligner, `eflomal`[27].

8 Conclusion

We developed `LaTeXMT`, a software solution which can reliably translate a wide range of LaTeX documents by separating their text content from non-text elements, producing LaTeX source files that faithfully preserve the original document structure. It maintains the relative positioning of content, supports mixed text and non-text elements (*masking*), and restores text formatting (*markup reinsertion*). In order to achieve markup reinsertion, we made use of *word alignments* obtained from the internal state of a neural machine translation model. State-of-the-art NMT models struggle to process LaTeX code effectively. While modern LLMs are better equipped to manage LaTeX, their capabilities are still limited by context size and language support. `LaTeXMT` offers a compelling solution by providing greater controllability and the ability to handle larger documents more efficiently. Through the integration of custom MT systems, users can tailor `LaTeXMT` to their specific needs, extending its applicability across a wider range of languages, also those not supported by LLMs.

Markup reinsertion based on word alignments usually yields good results, but it does not handle e.g. partial word markup, and it of course fails when word alignment does not find a correspondence between source text and its the translation. To address this, further refinement of the alignment mechanism is needed, possibly through the use of static analysis techniques.

Despite these limitations, `LaTeXMT` provides users with the necessary tools to correct errors as they arise, making it the preferred choice for handling complex LaTeX documents.

[21] https://pandoc.org/
[22] https://github.com/Ectalite/translatex
[23] https://github.com/alvinwan/texsoup
[24] https://okapiframework.org/
[25] https://github.com/mozilla/translations
[26] https://github.com/mozilla/firefox-translations-models/
[27] https://github.com/robertostling/eflomal

Acknowledgments

We would like to thank the anonymous reviewers for their work and valuable comments and suggestions, which have greatly helped to improve our presentation and Gernot Baumgartner for his invaluable support behind the scenes.

References

Zi-Yi Dou and Graham Neubig. 2021. Word Alignment by Fine-tuning Embeddings on Parallel Corpora. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Samuel Frontull and Georg Moser. 2024. Rule-Based, Neural and LLM Back-Translation: Comparative Insights from a Variant of Ladin. In *Proceedings of the The Seventh Workshop on Technologies for Machine Translation of Low-Resource Languages (LoResMT 2024)*, pages 128–138, Bangkok, Thailand. Association for Computational Linguistics.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How Good Are GPT Models at Machine Translation? A Comprehensive Evaluation. *Preprint*, arXiv:2302.09210.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2025. A survey on large language models for code generation. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.

Leslie Lamport. 1986. *LATEX: A Document Preparation System*. Addison-Wesley Publishing Company.

Aditya Ohri and Tanya Schmah. 2021. Machine Translation of Mathematical Text. *IEEE Access*, 9:38078–38086.

Jörg Steffen and Josef van Genabith. 2021. TransIns: Document Translation with Markup Reinsertion. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 28–34, Online and Punta Cana, Dominican Republic. 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.

Jörg Tiedemann, Mikko Aulamo, Daria Bakshandaeva, Michele Boggia, Stig-Arne Grönroos, Tommi Nieminen, Alessandro Raganato Yves Scherrer, Raul Vazquez, and Sami Virpioja. 2023. Democratizing neural machine translation with OPUS-MT. *Language Resources and Evaluation*, 58(2):713–755.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

A Limitations

Over the course of development, we made a number of decisions in approaching issues with no perfect solutions, with trade-offs usually being made in favour of more general solutions as opposed to more specific ones, and to avoid overcomplicating things as a whole. This ultimately left some edge cases we chose not to solve for, some of which we already mentioned, and all of which we will recount below. In this section, we describe these limitations and mention ways in which it might be approached.

Masking of word sequences Sometimes, a non-text element may act as a stand-in for not just a single noun, but several words. For instance, short equations may be used to say "X gleich Y" in German. Using the *masking* approach in this case may lead to an incorrect translation:

```
Prüfen Sie, ob 1$$A = B$$1 ist.
Prüfen Sie, ob #1_ ist.
Check whether #1_ is.
Check whether 1$$A = B$$1 is.
```

In general however, we found *masking* to be an effective strategy for preservation of non-translatable content—and while attempts could be made to solve for this particular edge cases, others most certainly exist, so we chose to accept it as a limitation.

Non-standard macros/environments It is difficult to determine ahead of LaTeX compilation whether or not the arguments to a macro are going to appear as document text. We thus cannot automatically determine which macros should have their contents translated, and rely instead on a list of well-known ones; unknown macros and their arguments are simply left untouched. Perhaps some form of static analysis could be applied, e.g. repeatedly expanding macros per their definition until only well-known macros remain, in order to track how the arguments are ultimately used.

Named Entities Any named entity that appears in the source text and which matches the source language is translated among with the text. This

simply cannot be solved without either making use of a very finely-trained machine translation model, or adjusting the input to demarcate passages of text that should be preserved verbatim (e.g. enclosing it within a non-standard macro).

Partially marked-up words Our alignment-based markup reinsertion method cannot map markup that spans only part of an input word to the corresponding part in the output. However, partial markup reinsertion is theoretically possible using token-level alignments. Still, the granularity of this approach is inherently limited by the model's tokenization, and thus alternative methods would be needed for fine-grained markup handling.

Comments In LaTeX, nothing can actually ever follow a comment within a line. One might thus consider remembering the contents of comments together with the line number they originally appeared on, then simply appending the comment to the same line after translation. However, this idea is complicated by the fact that we do not preserve individual line breaks in their original position, (as these are structurally irrelevant to the LaTeX engine), which shifts line numbers between the input and output documents.

Whitespace The text extraction functionality of `pylatexenc` simply maps all whitespace characters to a plain space, including non-breaking spaces and manual line breaks. While this behaviour could be changed with ease, this leaves the problem that there is no universal way to represent different whitespace characters in a way that will be reliably preserved by machine translation models, as most if not all perform whitespace-delimited word tokenization and do not concern themselves with different kinds of whitespace. Unlike non-text macros, whitespace also never represents any words within the text body. Thus, we cannot use masking to preserve them, as that would most certainly adversely affect translation results. Beyond paragraph splitting, we did not address with this issue, and accepted a lack of whitespace handling as a limitation. Since output documents tend to require manual review anyway, any non-breaking-spaces or manual newlines can be added back in manually after translation.

Code Listings Presently, the LaTeX parser we use represents the contents of code listings simply as a single text node. This could be extended to run another parser, based on the language specified for the code listing (or a "one-size-fits-most" parser if no language is specified), over the contents of code listings, breaking those up into comment and non-comment nodes, then adding the contents of comments to the list of text items to be translated.

B Test Data

This section includes the code snippets and descriptions of the larger test files used for the evaluation discussed in Section 5. The translation outputs of the different models are not included here, but can be found in the `latexmt/test-cases`[28] repository.

Synthetic LaTeX Snippets

A. LaTeX snippet with mathematical expressions

```
Konstruieren Sie eine TM $M$ über
  dem Alphabet $\Sigma = \{\ma, \mb
  \}$, welche die Sprache
\[
  L = \{x \mid x \text{ enthält den
  String $\ma \mb \ma$}\}
\]
akzeptiert.
```

```
Costruisci una Macchina di Turing
  $M$ sull'alfabeto $\Sigma = \{\ma,
   \mb\}$, tale che $M$ accetti la
  lingua
\[
  L = \{x \mid x \text{ contiene la
  secuenza $\ma \mb \ma$}\}
\]
```

B. Custom environment

```
\begin{hinweis}
  Das ist ein \emph{Hinweis} in
  einem \textbf{custom environment}.
\end{hinweis}
```

```
\begin{nota}
  Una \emph{nota} in un \textbf{
  ambiente personalizzato}.
\end{nota}
```

C. Custom command definition

```
\newcommand{\FF}{\mathsf{false}}
\newcommand{\TT}{\mathsf{true}}

Das \textbf{Ergebnis} ist $\TT$.
```

```
\newcommand{\FF}{\mathsf{false}}
\newcommand{\TT}{\mathsf{true}}

Il \textbf{risultato} equivale a $\
  TT$.
```

D. Custom command usage

[28] https://github.com/latexmt/test-cases

```
\hinweis{Das ist ein \textit{Hinweis
  }.}
```

```
\nota{Questo serve da \textit{nota
  }.}
```

E. Special symbols

```
\textit{Kategorisieren} Sie die
  Sprache der Palindrome \"uber dem
  Alphabet $\Sigma = \{\Null,\Eins\}
  $ anhand der \emph{Chomsky-
  Hierarchie}.
```

```
\textit{Categorizza} il linguaggio
  dei palindromi sull'alfabeto $\
  Sigma = \{\zero,\uno\}$
  utilizzando la \emph{Gerarchia di
  Chomsky}.
```

F. Special symbol 2

```
Es kann gezeigt werden dass die
  Sprache \textbf{\emph{nicht} regul
  \"ar} ist.
```

```
Si pu\`o dimostrare che il
  linguaggio \`e \textbf{\emph{non}
  regolare}.
```

G. `align*` with custom macros

```
\begin{align*}
  P & \to \lw \mid \Null \mid \Eins
    \\
  P & \to \Null P \Null \mid \Eins P
    \Eins
\end{align*}
```

```
\begin{align*}
  P & \to \lw \mid \Zero \mid \Uno
  \\
  P & \to \Zero P \Zero \mid \Uno P
  \Uno
\end{align*}
```

H. Sequence of custom macros

```
Testen Sie Ihren Automaten mit den
  Zeichenreihen $\mb\mb\mb$, $\mb\mb
  \ma\mb\mb$ und $\mb\mb\ma\mb\mb\ma
  \mb\mb$.
```

```
Testate il vostro automa a stati
  finiti con le sequenze $\mb\mb\mb$
  , $\mb\mb\ma\mb\mb$ e $\mb\mb\ma\
  mb\mb\ma\mb\mb$.
```

I. Define date

```
\newtheorem*{uebungen}{Übungen}
\date{\today}
```

```
\newtheorem*{esericizi}{Esercizi}
\date{\today}
```

J. Custom environment 2

```
\begin{leer}
  nicht leer
\end{leer}
```

```
\begin{vuoto}
    non vuoto
\end{vuoto}
```

K. Macro for untranslatable content

```
\newcommand{\germanText}{\textbf{
  Dies ist deutscher Text, der nicht
   übersetzt werden kann!}}

Hier ist der deutsche Text:
\germanText
```

```
\newcommand{\italianText}{\textbf{
  Questa frase italiana non deve
  essere tradotta!}}

Ecco la frase in italiano:
\italianText
```

L. `longtable` environment

```
\begin{longtable}{|c|c|c|}
\hline
\textbf{1} & \textbf{2} & \textbf{3}
   \\
\hline
\end{longtable}
```

```
\begin{longtable}{|c|c|c|}
\hline
\textbf{1} & \textbf{2} & \textbf{3}
   \\
\hline
\end{longtable}
```

Full-Documents Tests

R1. ACL Paper Style LaTeX Template `acl_latex.tex` available at https://github.com/acl-org/acl-style-files/

R2. Springer Nature LaTeX Template, available at https://www.overleaf.com/latex/templates/springer-nature-latex-template/myxmhdsbzkyd

R3. ACM Generic Journal Manuscript LaTeX Template, available at https://www.overleaf.com/latex/templates/association-for-computing-machinery-acm-generic-journal-manuscript-template/yffvrvzbhhpt

R4. IEEE Conference LaTeX Template, available at https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn

LangVAE and LangSpace: Building and Probing for Language Model VAEs

Danilo S. Carvalho[1], **Yingji Zhang**[2], **Harriet Unsworth**[1], **André Freitas**[1,2,3]
CRUK National Biomarker Centre, University of Manchester, United Kingdom[1]
Department of Computer Science, University of Manchester, United Kingdom[2]
Idiap Research Institute, Switzerland[3]
https://github.com/neuro-symbolic-ai/{LangVAE, LangSpace} | Short Video

Abstract

We present *LangVAE*, a novel framework for modular construction of variational autoencoders (VAEs) on top of pre-trained large language models (LLMs). Such language model VAEs can encode the knowledge of their pre-trained components into more compact and semantically disentangled representations. The representations obtained in this way can be analysed with the LangVAE companion framework: *LangSpace*, which implements a collection of probing methods, such as vector traversal and interpolation, disentanglement measures, and cluster visualisations. LangVAE and LangSpace offer a flexible, efficient and scalable way of building and analysing textual representations, with simple integration for models available on the HuggingFace Hub. Additionally, we conducted a set of experiments with different encoder and decoder combinations, as well as annotated inputs, revealing a wide range of interactions across architectural families and sizes w.r.t. generalisation and disentanglement. Our findings demonstrate a promising framework for systematising the experimentation and understanding of textual representations.

1 Motivation and Purpose

Variational Autoencoders (VAEs) (Kingma et al., 2013) are of considerable importance in machine learning due to their capacity to integrate prior knowledge, quantify uncertainty, enhance generalisation, and deliver interpretability. First, the integration of prior distribution serves as an inductive bias, enabling the model to leverage existing knowledge and providing a principled way to incorporate domain expertise. In the computational linguistics domain, for example, the hierarchical syntax information can be well encoded via hyperbolic prior (Davidson et al., 2018; Cho et al., 2023). Second, their probabilistic formulation allows for explicit uncertainty quantification, providing not only point estimates but also confidence intervals over latent variables and reconstructions, which is significant in the Safety and Trustworthy AI domain, such as hallucinations of LLMs (Ji et al., 2023). Third, by enforcing a smooth and continuous latent space, VAEs promote better composition and generalisation, as they capture the underlying generative factors of the input distribution (Bonnet and Macfarlane, 2024). Fourth, the latent space can compress the knowledge into abstract-level concepts, which is similar to how humans understand the world (Barrault et al., 2024). Concurrently, the rapid pace of development of LLMs has led to substantial gains in a wide variety of NLP tasks, demonstrating remarkable knowledge representation capabilities (Kauf et al., 2023; Selby et al., 2025), but present critical challenges in interpretability and fine-grained control (Kunz and Kuhlmann, 2022; Friedman et al., 2024).

To leverage the strengths of both LMs and VAEs, Language model-based VAEs (LM-VAEs) (Bowman et al., 2015) have been proposed and widely deployed in the controlled text generation domain, such as style transfer tasks: modifying sentences with regard to markers of sentiment, formality, affirmation/negation (Bao et al., 2019; Vasilakes et al., 2022; Gu et al., 2022; Liu et al., 2023; Gu et al., 2023; Liu et al., 2024) and textual, syntactic, semantic representation learning domain (Mercatali and Freitas, 2021; Carvalho et al., 2023; Zhang et al., 2024b,c). However, despite their strategic positioning in delivering more controlled latent representations, there has been limited software infrastructure support to facilitate experimentation with LM-VAEs and in particular, scaling-up to Large Language Model configurations (LLM-VAEs).

In this work we address these issues by presenting a novel framework for modular construction of LM-VAEs on top of pre-trained LMs of different scales, called *LangVAE*, and its companion framework *LangSpace*, dedicated to latent space probing and evaluation. LangVAE introduces a novel ap-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 749–759

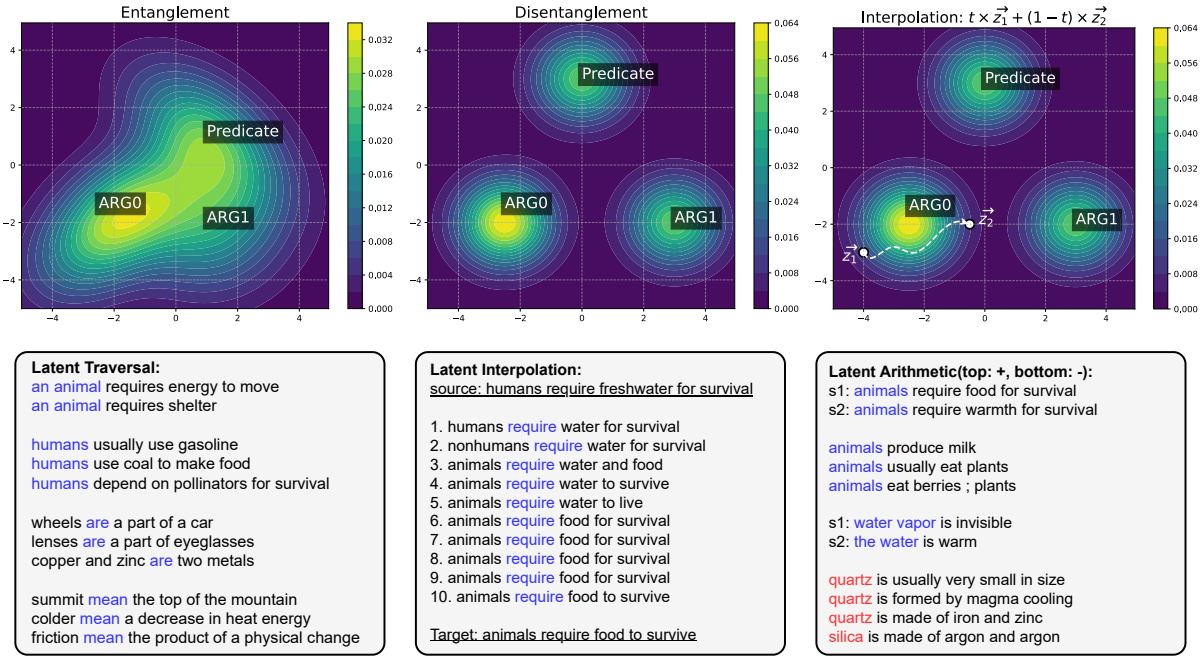


Figure 1: *LangVAE* is a flexible framework designed to support arbitrary combinations of pretrained encoders and decoders for learning latent representations under either a categorical semantic prior or a Gaussian prior. *LangSpace* facilitates comprehensive analysis of the learned latent space through automated evaluation of key properties such as disentanglement and visualization (top) and enables controlled generation by leveraging these latent properties, such as latent traversal, interpolation, and arithmetic operations (bottom).

proach for latent vector unpooling to autoregressive LMs that sharply reduces the computational and memory requirements, while incorporating compatibility to contemporary LLMs and hardware optimisations.

Finally, we conducted a set of experiments as a case study to demonstrate the frameworks' semantic representation capabilities and highlight the effects of different combinations of encoder and decoder models, in terms of generalisation and latent space disentanglement, evidencing the impact of facilitating a systematic analysis across different encoder-bottleneck-decoder combinations and parametrisations.

Both frameworks are available as python libraries in the PyPI package repository and on public source code repositories[1] [2]. A demonstration video is available at: youtu.be/DVcrdIX9CfI.

2 Language Model VAEs

A language model VAE (LM-VAE) is a variational autoencoder where both the encoder and decoder components are LMs (Bowman et al., 2015; Li et al., 2020; Tu et al., 2022; Zhang et al., 2023). It can encode the knowledge of their pre-trained components into compact latent vectors and enables guided language generation from an abstract level using said vectors. The benefits of such models also extend to interpretability (due to their better disentanglement properties), as the VAE architectural bottleneck provides a singular point for probing a model's latent space structure and its syntactic/semantic representation (Li et al., 2020; Mercatali and Freitas, 2021; Carvalho et al., 2023; Zhang et al., 2024b,c) and inferential properties (Bonnet and Macfarlane, 2024). The creation of continuous latent representation spaces, with better disentanglement and separability of syntactic/semantic properties offers a key mechanism for supporting generative control both at the level of sentences (Bao et al., 2019; Felhi et al., 2022; Zhang et al., 2024c) and natural language inferences (Yu et al., 2022).

In its most basic conceptualisation, an LM-VAE consists of: (**a**) an encoder type LLM (e.g., BERT, T5), to provide base representations for each token of the input text; (**b**) a pooling process to accumulate the input token representations; (**c**) a projection layer, to convert the base encoding to the regularised VAE latent space; (**d**) an unpooling process,

[1] https://github.com/neuro-symbolic-ai/LangVAE
[2] https://github.com/neuro-symbolic-ai/LangSpace

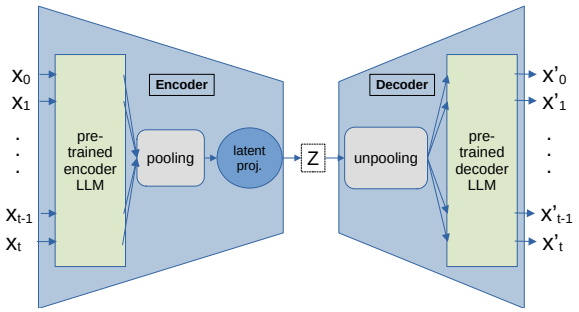


Figure 2: Diagram of fundamental LMVAE architecture.

to derive token representations from a latent vector and feed them to the decoder; and (**e**) a decoder type LLM (e.g., GPT, Llama) capable of generating tokens from a sequence of input representations. This structure is illustrated in Figure 2. On the top of this base configuration, syntactic and semantic features can be injected into the latent space, aiming to improve the localisation and control of such features via conditionalisation mechanisms, such as CVAE or clustering losses. Moreover, further architectural interventions can be integrated aiming for additional control, such as the addition of INN layers (Zhang et al., 2024a), aiming for improving the separability of semantic features.

2.1 Optimus

The pioneer LM-VAE is Optimus (Li et al., 2020), which combines a BERT encoder and a GPT-2 decoder to perform sentence encoding, using a mean pooling process, a linear projection layer (MLP), and an unpooling process consisting of two concurrent schemes for latent memory injection to the decoder:
Memory: appends a projection of the latent vector directly to each hidden layer of the decoder as a hidden memory vector for the decoder to attend.
Embedding: adds a projection of the latent vector to the decoder embedding layer at each decoding step.

Optimus is trained end-to-end, meaning that the encoder projection layer and the memory and embedding injection layers are jointly trained with the base encoder and decoder models. In this way, the pre-trained models are fine-tuned to "weld" with the projection and injection layers, facilitating convergence.

Despite its demonstrated capabilities and potential, Optimus has some limitations, in particular regarding model coupling and scalability. We next discuss our proposed approach for building LM-VAEs and its improvements over the SOTA.

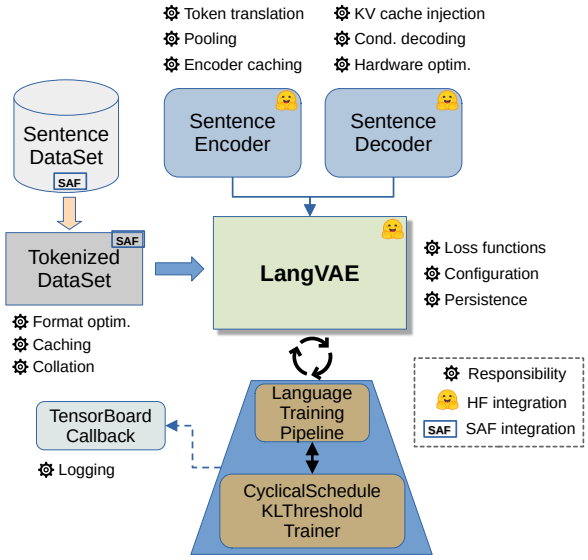


Figure 3: Overview of the LangVAE framework.

3 LangVAE: Building modular LM-VAEs

Aiming to address current LM-VAE limitations and facilitate the development of specialised models and experimentation over next-gen LLMs, we developed *LangVAE*. This is a novel framework targeted at language representation learning research, focused on the modular development of the architectural components discussed in the previous section (especially projections and unpooling processes), and having a strong integration with the python transformers library[3]. LangVAE is developed and distributed as a python library[1] under the GPLv3 License. It is built on top of the pythae library for autoencoders (Chadebec et al., 2022). Figure 3 provides an overview of LangVAE's modules and responsibilities.

3.1 Architecture

LangVAE implements the fundamental LM-VAE architecture (Figure 2) in the following ways:

Pre-trained LLM encoder: as a loader for an encoder type LLM compatible with the transformers library, via the automodel classes (AutoModel, AutoModelForTextEncoding).

Pooling process: mean pooling, last hidden state of the base encoder, or the CLS token hidden state, which is automatically selected depending on the pre-trained encoder model configuration.

Latent projection layer: a linear MLP that adjusts the input encoding size on training time.

[3] https://github.com/huggingface/transformers

Unpooling process: a variation of the memory injection scheme from Optimus, called *KV cache injection*, which does not require customisation of the pre-trained decoder code. Instead, it uses the transformers library KV caching mechanism for guiding the decoder (detailed in the next section).

Pre-trained LLM decoder: same as the encoder, but relying on the transformers *AutoModelForCausalLM* class for model parametrisation regarding tokenizer configuration and hardware optimisations (e.g., flash attention and multi-GPU distribution).

In addition, LangVAE provides the following functionalities:

Data conversion: TokenizedDataSet classes for convenient and efficient tokenisation of text datasets, including the handling of annotations.

Training pipeline: supporting cyclical schedule KL annealing to avoid the KL vanishing problem, with beta and KL thresholds.

Training monitoring: with tensorboard logging.

3.2 KV cache injection

One of the central contributions behind LangVAE is the key-value (KV) cache injection scheme, as an alternative to Optimus' memory injection. This new scheme uses the KV caching mechanism of the Causal LM model classes within the transformers library to inject a positional projection of the latent vector. A linear projection of the latent vector $h_{cache} = W_m z$ plays the role of an additional context to guide generation, in the form of hidden KV cache entries X_t^h interleaved with those produced by the decoder, where $W_m \in \mathbb{R}^{LH \times S}$ is separated into $S \times L$ (sequence length * # layers) vectors of hidden size $H = K \times V$. Figure 4 illustrates this scheme.

There are two main advantages to this approach. Firstly, it eliminates the need to change the layout of the hidden layers to accommodate the injected memory vector. Therefore, it is compatible with any model that supports the KV caching mechanism. Lastly, in enables training the LM-VAE model with the weights for the base pretrained models frozen, greatly reducing the computational and memory requirements. Additionally, this scheme allows distributed training of the injection layer, as the projection matrices can be co-located with the respective hidden layers in training time, and size

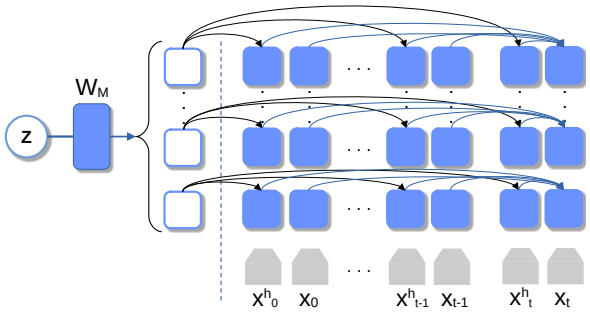


Figure 4: Illustration of the KV cache injection scheme. $W_m z$ projects hidden KV cache entries X_t^h that are attended by the decoder when predicting the next token. The hidden cache entries are interleaved with the ones produced by the decoder.

of context (number of hidden cache entries) can be adjusted.

3.3 Main advantages & Limitations

The main advantages of LangVAE can be summarised as follows:

• Modular architecture allows flexible development of different LM-VAE configurations. Flexible composition of base models and bottleneck parametrisations, loss functions, etc.

• Compatible with most state-of-the-art autoregressive models.

• Has a substantially reduced computational requirements for training, compared to the SOTA LM-VAE (Optimus), with an average parameter reduction of over 95% measured when using decoder models between 3B to 7B parameters (Section 5.1).

• Supports multi-GPU training and inference.

Its main limitations are related to the cache injection mechanism:

• Slower convergence, as there are far less parameters to adjust.

• Latent vector sizes tend to be larger, compared to Optimus, to compensate for the overall parameter reduction.

• The model size scales linearly with the maximum context size, creating a tradeoff between efficiency and context capacity.

3.4 Installation and API Examples

LangVAE can be installed directly from the PyPI package repository with: *pip install langvae*

We briefly illustrate the key components of LangVAE's API and how they are instantiated in the supplementary material (Appendix Section A.1). A full example of model training can be found in the

README file of the code repository[1] and on the supporting python notebook[4].

4 LangSpace: Simplified probing for LM-VAEs

LangSpace[2], is a companion framework to LangVAE focused on the evaluation and on the latent space probing for LM-VAEs. It provides an easy-to-use API to perform a variety of analyses on pre-trained LM-VAEs models, namely:

• Probes: vector arithmetic and interpolation, latent space traversal, disentanglement and cluster visualisation.

• Metrics: disentanglement (z-diff, z-min-var, MIG, Disentanglement, Informativeness, Completeness), interpolation (quality, smoothness).

4.1 Installation and API Examples

Like LangVAE, LangSpace can be installed from the PyPI repository with: *pip install langspace*

We briefly illustrate below the use of one of LangSpace probes: latent traversal. A full example with all the available probes can be found within the available public notebook[5], with some illustrated in Figures 6 and 7 (Appendix B).

```
# Seed sentences to traverse
sentences = [
    "animals require food to survive",
    "water vapor is invisible"
]
# Dataset importer
ds = ListImporter()(
    [sent.split() for sent in sentences]
).sentences
# Create dataset tokeniser
seeds = TokenizedDataSet(
    ds, model.decoder.tokenizer,
    model.decoder.max_len
)
# Create probe and generate report: a
# dataframe with collumns for the
# original sentence, traversed dimension,
# distance traversed and the generated
# result, respectively.
trav_report = TraversalProbe(
    model, trav_dataset,
    sample_size=10,
    dims=list(range(128))
).report()
```

[4] https://bit.ly/3FMPg5N

[5] https://bit.ly/424bjw3

5 Case study & Model availability

To demonstrate LangVAE and LangSpace capabilities for language modeling and highlight the effects of different combinations of encoder and decoder models, in terms of generalisation and disentanglement of the latent space, we conducted a set of experiments as a case study. The experiments consist of a simple explanation sentence modeling task (Zad et al., 2021; Dalvi et al., 2021) with posterior evaluation of the induced latent space. While LM-VAE models could be employed in a range of downstream tasks (e.g., sentiment analysis, style-transfer, among others), this study focuses on the semantic representation aspect and its conditional intervention. Pre-trained checkpoints for all model combinations presented in this study are available in our public HF Hub repository[6].

5.1 Experimental setup

For the pre-trained LLMs, we selected three distinct encoder models, in order of parameter size: BERT (base-cased) (Devlin et al., 2019), Flan-T5 (base) (Chung et al., 2024) and Stella (en-1.5B_v5) (Zhang et al., 2025), and four decoder models: GPT-2 (base) (Radford et al., 2019), Qwen (2.5-3B) (Team, 2024), Llama (3.2-3B) (Grattafiori et al., 2024) and Mistral (7B-v0.3) (Jiang et al., 2023). The selection considered the inclusion of different model families and sizes. For each combination, inputs without and with semantic role labeling (SRL) annotations were used (as semantic features), where the SRL annotations were passed as additional variables (one-hot encoded) to the encoder only, going through a separate pooling process (always mean pooling). The latent size (128) and maximum sentence was kept the same for all tests. All models were trained for 50 epochs, with $LR = 0.001$, $target\_kl = 2.0$, $max\_beta = 1.0$, 40 beta annealing cycles, and batch size of 50. Disentanglement measurements were obtained using LangSpace's disentanglement probe for the metrics z-diff (Higgins et al., 2017), z-min-var (Kim and Mnih, 2018) and Informativeness (Eastwood and Williams, 2018).

All experiments were performed on a computer with the following specifications: CPU: AMD EPYC 7413 24-Core, GPU: 2x NVIDIA A100-SXM4-80GB, Memory: 200GB. LangVAE allows caching of the base encoder outputs, which causes the training time to be mostly dominated by the

[6] https://huggingface.co/neuro-symbolic-ai

base decoder inference time. The shortest training time was of aprox. 1h (GPT-2) and the longest was about 4.5h (Mistral-7B). Training requirements for larger decoders scale similarly to inference, with a training run of Phi-4 (14B) also taking about 4.5h to complete. The ratios of the LangVAE trained models' size to the base LLMs was: GPT-2 = 0.547, Qwen2.5-3B = 0.024, Llama3.2-3B = 0.076, Mistral-7B = 0.037. Excluding GPT-2, this represents an over 95% parameter reduction.

5.2 Data

The same data was used for all tests: a subset of all explanatory sentences from the EntailmentBank dataset (Dalvi et al., 2021), which was loaded using the *saf-datasets*[7] library. The dataset contains 12496 sentences, from which 99% were used for training and 1% for validation[8]. Evaluation was performed on a random sample of 200 sentences including the validation set and a small portion of the training set. SRL annotation was performed using the AllenNLP[9] library with a SOTA SRL model (Shi and Lin, 2019).

5.3 Results

The results for the explanation sentence modeling task are presented in Table 1. The first observation is that the highest reconstruction performance was achieved by the smallest model combination (for SRL). While not the expected outcome, this can be explained by the constraint imposed on the latent space size and training data, causing the simpler model to better generalise the inputs.

The encoder complexity has a substantial impact on the generalisation capability of the model: even though bert-base-cased and flan-t5-base have the same encoding size (768), BERT outperforms T5 in most cases, indicating a higher level of information entanglement on T5. Stella, on the other hand, has a much larger encoding size (1536), with a larger dominating effect based on the information loss over the dimensionality reduction.

The injection of the SRL categories within the model improved reconstruction performance in all combinations except when Mistral is the decoder. This is a surprising result and indicate some particularity of Mistral's internal representations that invite further investigation.

[7] https://github.com/neuro-symbolic-ai/saf_datasets

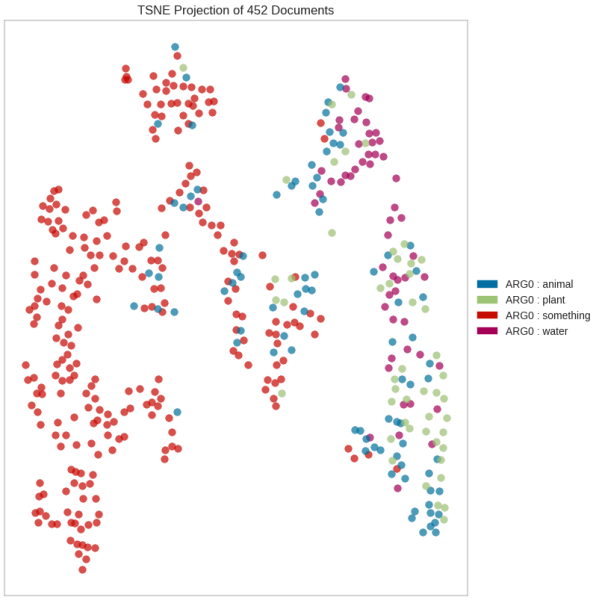
[8] The validation split here is just a means to prevent overfitting, since we are only testing the representations.

[9] https://github.com/allenai/allennlp-models

Encoder	Decoder	Annot.	Reconstr. (BLEU)	Disentanglement z-diff	z-m-var ↓	inform.
BERT	gpt-2	-	0.76	0.46	0.68	0.36
BERT	gpt-2	SRL	**0.84**	0.43	0.70	0.40
BERT	Qwen	-	0.44	0.58	0.69	0.46
BERT	Qwen	SRL	0.49	0.53	0.61	0.44
BERT	Llama	-	0.65	**0.62**	0.71	0.38
BERT	Llama	SRL	0.80	0.59	0.65	0.43
BERT	Mistral	-	0.81	0.51	**0.59**	0.43
BERT	Mistral	SRL	0.75	0.55	0.62	0.44
Flan-T5	gpt-2	-	0.11	0.50	0.62	0.35
Flan-T5	gpt-2	SRL	0.81	**0.62**	0.67	0.42
Flan-T5	Qwen	-	0.19	0.52	0.69	0.39
Flan-T5	Qwen	SRL	0.31	0.55	0.68	0.43
Flan-T5	Llama	-	0.74	0.52	0.68	**0.49**
Flan-T5	Llama	SRL	0.80	0.59	0.64	0.41
Flan-T5	Mistral	-	0.78	**0.62**	0.61	0.39
Flan-T5	Mistral	SRL	0.72	0.51	0.63	0.43
Stella	gpt-2	-	0.18	0.50	0.68	0.34
Stella	gpt-2	SRL	0.61	0.52	0.65	0.40
Stella	Qwen	-	0.15	0.48	0.69	0.44
Stella	Qwen	SRL	0.27	0.54	0.66	0.43
Stella	Llama	-	0.45	0.51	0.73	0.40
Stella	Llama	SRL	0.64	**0.62**	0.72	0.42
Stella	Mistral	-	0.57	0.54	0.72	0.46
Stella	Mistral	SRL	0.55	0.51	0.71	0.39

Table 1: Results from the explanation sentence modeling experiments. Best values for each column in bold.

Finally, the SRL categories did not induce consistent improvements on the disentanglement scores, with the exception of Llama3.2, where it led to qualitative improvements in terms of SRL cluster separability, as illustrated in Figure 5. We additionally trained a single instance of Llama3.1-8B on the Wiktionary dataset from *saf-datasets*[7] (∼1M sentences) and analysed latent space interpolation, illustrated on Table 2.



(a) No annotation

(b) SRL annotation

Figure 5: TSNE plots for the [bert-base-cased, Llama-3.2-3B] combination, without (a) and with (b) SRL annotated inputs. We can observe a better separation of the *water* and *animal* subjects on the annotated model.

Source / Target	Distance	Generate
the high seas / the continent	0.361	1. the high 2. the high 3. the sea 4. the sea 5. the sea 6. the sea 7. the sea 8. the land 9. the world 10. the world
a primary schooler / a college student	0.299	1. a primary school 2. a primary school 3. a junior school 4. a junior school 5. a high school 6. a high school 7. a high student 8. a college 9. a college 10. a student

Table 2: Latent interpolation example using a LangVAE model (BERT [base-cased] - Llama3.1 [8B]), trained on the Wiktionary dataset. Ten points connecting from the source to the target latent vectors are decoded to generate a list of interpolated sentences. We used short sentences for which there are expected intermediate concepts. We can observe a semantic progression when connecting terms for which there are intermediate senses.

6 Conclusion

In this work we presented *LangVAE*, a modular and efficient library for building language model VAEs (LM-VAEs), and its companion framework *LangSpace*, dedicated to LM-VAE latent space control, probing and evaluation. With the goal of lowering the experimental barriers in this research area, it introduces a novel approach for latent vector unpooling to autoregressive LMs that sharply reduces the computational and memory requirements for training such models, along with a flexible code architecture which is oriented towards modern LLM development.

We demonstrated the representation capabilities of LangVAE and LangSpace with a set of experiments using different encoder and decoder combinations, as well as annotated inputs, which reveal a wide range of interactions across architectural families and sizes w.r.t. generalisation and disentanglement. Such interactions point to uncovered factors regarding the models' internal representation properties and how they exchange information.

Limitations

While the evaluation aimed at covering a diverse set of pre-trained encoders and decoders, it is limited in the scope of downstream tasks and dataset diversity. Further analisys of the failure modes of each encoder-decoder combination should also be done to better understand the unexpected results and how latent space geometry can be optimised. We leave those as future work.

LangVAE is designed to work with the standard model interfaces provided by the HuggingFace *transformers* library[3], and may not be compatible with models containing specialised custom code or that do not have a KV cache implementation or use a different one (e.g., Gemma). We welcome contributions to the LangVAE and LangSpace code repositories[1,2], specially those aimed at increasing model compatibility.

Acknowledgements

This work was partially funded by the SNSF project RATIONAL (200021E_229196), the CRUK National Biomarker Centre, and supported by the Manchester Experimental Cancer Medicine Centre and the NIHR Manchester Biomedical Research Centre.

References

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6008–6019.

Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R Costa-jussà, David Dale, et al. 2024. Large concept models: Language modeling in a sentence representation space. *arXiv preprint arXiv:2412.08821*.

Clément Bonnet and Matthew V Macfarlane. 2024. Searching latent program spaces. *arXiv preprint arXiv:2411.08706*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Danilo Silva de Carvalho, Giangiacomo Mercatali, Yingji Zhang, and André Freitas. 2023. Learning disentangled representations for natural language definitions. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1371–1384.

Clément Chadebec, Louis Vincent, and Stephanie Allassonniere. 2022. Pythae: Unifying generative autoencoders in python - a benchmarking use case. In *Advances in Neural Information Processing Systems*, volume 35, pages 21575–21589. Curran Associates, Inc.

Seunghyuk Cho, Juyong Lee, and Dongwoo Kim. 2023. Hyperbolic vae via latent gaussian distributions. *Advances in Neural Information Processing Systems*, 36:569–588.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. 2018. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*.

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.

Cian Eastwood and Christopher KI Williams. 2018. A framework for the quantitative evaluation of disentangled representations. In *6th International Conference on Learning Representations*.

Ghazi Felhi, Joseph Le Roux, and Djamé Seddah. 2022. Towards unsupervised content disentanglement in sentence representations via syntactic roles. *arXiv preprint arXiv:2206.11184*.

Dan Friedman, Andrew Lampinen, Lucas Dixon, Danqi Chen, and Asma Ghandeharioun. 2024. Interpretability illusions in the generalization of simplified models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 14035–14059.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yuxuan Gu, Xiaocheng Feng, Sicheng Ma, Lingyuan Zhang, Heng Gong, and Bing Qin. 2022. A distributional lens for multi-aspect controllable text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1023–1043, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuxuan Gu, Xiaocheng Feng, Sicheng Ma, Lingyuan Zhang, Heng Gong, Weihong Zhong, and Bing Qin. 2023. Controllable text generation via probability density estimation in the latent space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12590–12616, Toronto, Canada. Association for Computational Linguistics.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

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.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Carina Kauf, Anna A Ivanova, Giulia Rambelli, Emmanuele Chersoni, Jingyuan Selena She, Zawad Chowdhury, Evelina Fedorenko, and Alessandro Lenci. 2023. Event knowledge in large language models: the gap between the impossible and the unlikely. *Cognitive Science*, 47(11):e13386.

Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2649–2658. PMLR.

Diederik P Kingma, Max Welling, et al. 2013. Auto-encoding variational bayes.

Jenny Kunz and Marco Kuhlmann. 2022. Where does linguistic information emerge in neural language models? measuring gains and contributions across layers. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4664–4676.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699, Online. Association for Computational Linguistics.

Guangyi Liu, Zeyu Feng, Yuan Gao, Zichao Yang, Xiaodan Liang, Junwei Bao, Xiaodong He, Shuguang Cui, Zhen Li, and Zhiting Hu. 2023. Composable text controls in latent space with ODEs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16543–16570, Singapore. Association for Computational Linguistics.

Yi Liu, Xiangyu Liu, Xiangrong Zhu, and Wei Hu. 2024. Multi-aspect controllable text generation with disentangled counterfactual augmentation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9231–9253, Bangkok, Thailand. Association for Computational Linguistics.

Giangiacomo Mercatali and André Freitas. 2021. Disentangling generative factors in natural language with discrete variational autoencoders. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3547–3556, Punta Cana, Dominican Republic. 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.

David Selby, Yuichiro Iwashita, Kai Spriestersbach, Mohammad Saad, Dennis Bappert, Archana Warrier, Sumantrak Mukherjee, Koichi Kise, and Sebastian Vollmer. 2025. Had enough of experts? quantitative knowledge retrieval from large language models. *Stat*, 14(2):e70054.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *ArXiv*, abs/1904.05255.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Haoqin Tu, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. 2022. Adavae: Exploring adaptive gpt-2s in variational auto-encoders for language modeling. *arXiv preprint arXiv:2205.05862*.

Jake Vasilakes, Chrysoula Zerva, Makoto Miwa, and Sophia Ananiadou. 2022. Learning disentangled representations of negation and uncertainty. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8380–8397, Dublin, Ireland. Association for Computational Linguistics.

Jialin Yu, Alexandra I Cristea, Anoushka Harit, Zhongtian Sun, Olanrewaju Tahir Aduragba, Lei Shi, and Noura Al Moubayed. 2022. Interaction: a generative xai framework for natural language inference explanations. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Samira Zad, Maryam Heidari, Parisa Hajibabaee, and Masoud Malekzadeh. 2021. A survey of deep learning methods on semantic similarity and sentence modeling. In *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0466–0472. IEEE.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. Jasper and stella: distillation of sota embedding models.

Yingji Zhang, Danilo Carvalho, and André Freitas. 2024a. Learning disentangled semantic spaces of explanations via invertible neural networks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2113–2134.

Yingji Zhang, Danilo Carvalho, Marco Valentino, Ian Pratt-Hartmann, and André Freitas. 2024b. Improving semantic control in discrete latent spaces with transformer quantized variational autoencoders. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1434–1450.

Yingji Zhang, Danilo S Carvalho, Ian Pratt-Hartmann, and André Freitas. 2023. Llamavae: Guiding large language model generation via continuous latent sentence spaces. *arXiv preprint arXiv:2312.13208*.

Yingji Zhang, Marco Valentino, Danilo Carvalho, Ian Pratt-Hartmann, and André Freitas. 2024c. Graph-induced syntactic-semantic spaces in transformer-based variational autoencoders. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 474–489.

A API examples

A.1 LangVAE

```
# Creates a GPT-2 based decoder expecting
# a latent vector of size 128, that
# generates a maximum of 32 tokens,
# distributed on any number of CUDA GPUs.
decoder = SentenceDecoder(
    "gpt2", latent_size=128,
    max_len=32, device="cuda",
    device_map="auto"
)

# Creates a BERT based encoder producing
# a latent vector of size 128, expecting
# GPT-2 tokenised inputs.
encoder = SentenceEncoder(
    "bert-base-cased", latent_size=128,
    decoder.tokenizer, device="cuda"
)

# Defines a basic VAE model configuration
model_config = VAEConfig(latent_dim=128)

# Initialise LangVAE model
model = LangVAE(
    model_config, encoder, decoder
)

# Alternatively, loads a pretrained
# checkpoint from the HF Hub.
org = "neuro-symbolic-ai"
name="eb-langvae-flan-t5-base-gpt2-l128"
model = LangVAE.load_from_hf_hub(
    f"{org}/{name}"
)
```

B Notebook examples

```python
from langspace.probe import ClusterVisualizationProbe
from langspace.probe.cluster_vis.methods import ClusterVisualizationMethod as CvM

target_roles = {"ARG0": ["animal", "water", "plant", "something"]}
labels = list(target_roles.keys())
label_map = dict()
for prefix in ["B-", "I-"]:
    for lbl in labels:
        target_roles[prefix + lbl] = target_roles[lbl]
        label_map[prefix + lbl] = lbl

cluster_viz_report = ClusterVisualizationProbe(model, dataset, sample_size=1000, target_roles=target_roles,
                                               methods=[CvM.TSNE, CvM.PCA], cluster_annotation="srl_f",
                                               annotations=annotations, plot_label_map=label_map).report()
```

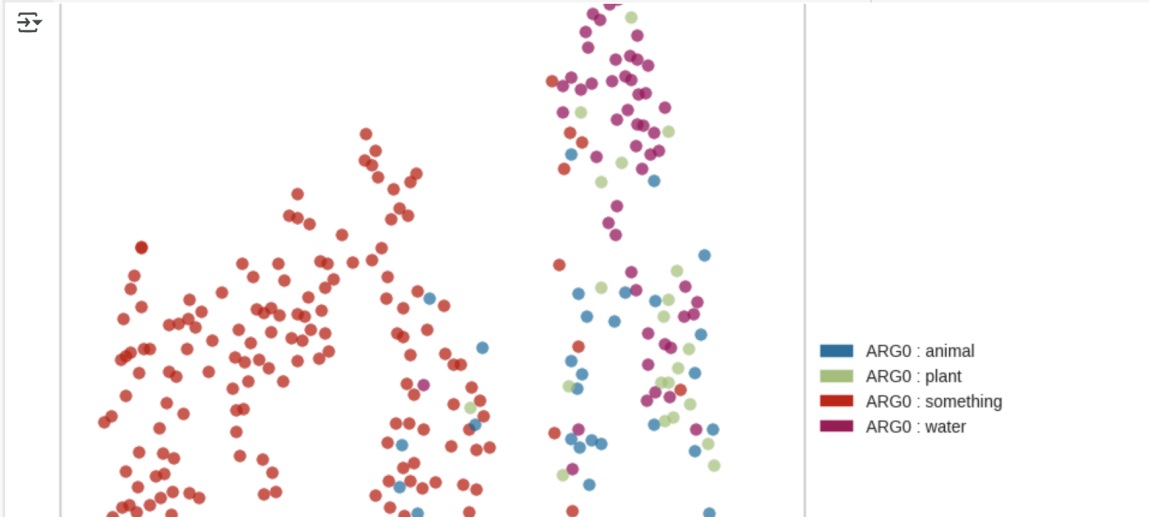


Figure 6: Example of the clustering probe use in the notebook.

```python
from langspace.probe import ArithmeticProbe
from langspace.ops.arithmetic import ArithmeticOps
from saf.importers import ListImporter

sentences = [
    ("animals require food for survival", "B-ARG0 B-V B-ARG1 B-ARGM-PRP I-ARGM-PRP"),
    ("animals require warmth for survival", "B-ARG0 B-V B-ARG1 B-ARGM-PRP I-ARGM-PRP"),
    ("water vapor is invisible", "B-ARG0 I-ARG0 B-V B-ARG1"),
    ("the water is warm", "B-ARG0 I-ARG0 B-V B-ARG1")
]
sentences_ds = ListImporter(annotations=["srl_f"])([[(tok, lbl) for tok, lbl in zip(sent[0].split(), sent[1].split())] for sent in sentences]).sentences

op_dataset = [(sentences_ds[0], sentences_ds[1]), (sentences_ds[2], sentences_ds[3])]

arith_report = ArithmeticProbe(model, op_dataset, ops=list(ArithmeticOps), annotations=annotations).report()
```

Show hidden output

```python
arith_report
```

	source	target	op	generate
0	animals require food for survival	animals require warmth for survival	sum	animals require food for survival
1	water vapor is invisible	the water is warm	sum	water water is
2	animals require food for survival	animals require warmth for survival	sub	to a resource a a effect of the the resource ...
3	water vapor is invisible	the water is warm	sub	a wireless is made of air
4	animals require food for survival	animals require warmth for survival	avg	animals require food for survival
5	water vapor is invisible	the water is warm	avg	water is is invisible

Figure 7: Example of the arithmetic probe use in the notebook.

PresentAgent: Multimodal Agent for Presentation Video Generation

Jingwei Shi[1*] **Zeyu Zhang**[1*†] **Biao Wu**[2*] **Yanjie Liang**[1*]
Meng Fang[3] **Ling Chen**[2] **Yang Zhao**[4‡]
[1]AI Geeks, Australia
[2]Australian Artificial Intelligence Institute, Australia
[3]University of Liverpool, United Kingdom
[4]La Trobe University, Australia
[*]Equal contribution. [†]Project lead. [‡]Corresponding author: y.zhao2@latrobe.edu.au.

Abstract

We present PresentAgent, a multimodal agent that transforms long-form documents into narrated presentation videos. While existing approaches are limited to generating static slides or text summaries, our method advances beyond these limitations by producing fully synchronized visual and spoken content that closely mimics human-style presentations. To achieve this integration, PresentAgent employs a modular pipeline that systematically segments the input document, plans and renders slide-style visual frames, generates contextual spoken narration with large language models and Text-to-Speech models, and seamlessly composes the final video with precise audio-visual alignment. Given the complexity of evaluating such multimodal outputs, we introduce PresentEval, a unified assessment framework powered by Vision-Language Models that comprehensively scores videos across three critical dimensions: content fidelity, visual clarity, and audience comprehension through prompt-based evaluation. Our experimental validation on a curated dataset of 30 document–presentation pairs demonstrates that PresentAgent approaches human-level quality across all evaluation metrics. These results highlight the significant potential of controllable multimodal agents in transforming static textual materials into dynamic, effective, and accessible presentation formats. Code will be available at https://github.com/AIGeeksGroup/PresentAgent.

1 Introduction

Presentations are a widely used and effective medium for conveying complex ideas. By combining visual elements, structured narration, and spoken explanations, they enable information to unfold progressively and be more easily understood by diverse audiences (Fu et al., 2022). Despite their proven effectiveness, creating high-quality presentation videos from long-form documents—such as business reports, technical manuals, policy briefs, or academic papers—typically requires considerable manual effort (Li et al., 2023). This process involves identifying key content, designing slide layouts, writing scripts, recording narration, and aligning all elements into a coherent multimodal output.

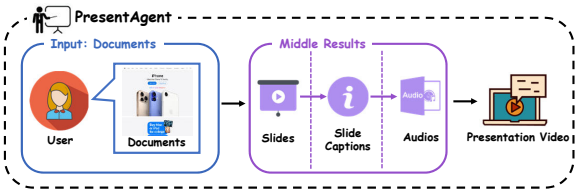


Figure 1: **Overview of PresentAgent.** It takes documents (e.g., web pages) as input and follows a generation pipeline: (1) document processing, (2) structured slide generation, (3) synchronized caption creation, and (4) audio synthesis. The final output is a presentation video combining visual slides with aligned narration. The purple-highlighted middle results emphasize the system's key transitional outputs during generation.

Although recent advancements in AI have enabled progress in related areas such as document-to-slide generation (Fu et al., 2022; Zheng et al., 2025a; Pang et al., 2025; Zhang et al., 2024) and text-to-video synthesis (Yang et al., 2024c; Li et al., 2023; Xue et al., 2025; Khachatryan et al., 2023; He et al., 2023; Solanki and Khublani, 2024), a critical gap remains: these methods either produce static visual summaries or generic video clips without structured narration, limiting their effectiveness for structured communication tasks like presentations.

To bridge this gap, we introduce the task of Document-to-Presentation Video Generation, which aims to automatically convert a structured or unstructured document into a narrated video presentation composed of synchronized slides and speech. This task presents unique challenges as it goes beyond traditional summarization (Lewis et al., 2019; Beltagy et al., 2020; Chen and Yang, 2021; Wang

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 760–773
November 4-9, 2025

et al., 2024a) or text-to-speech (Tachibana et al., 2018; Ren et al., 2019; Popov et al., 2021; Ni et al., 2022) pipelines by requiring selective content abstraction, layout-aware planning (Wang et al., 2025), and precise multimodal alignment (Li et al., 2024) between visuals and narration. In contrast to prior work that focuses on either static slide and image generation (Zheng et al., 2025a; Deng et al., 2025; Xie et al., 2024) or audio summarization in isolation, our objective is to produce a fully integrated, viewer-ready video experience that closely mimics how human presenters deliver information in real-world scenarios.

To tackle these challenges, we propose a modular generation framework named PresentAgent, as shown in Figure 1. Given an input document, the system first segments it into semantic blocks through outline planning, then generates layout-guided slide visuals for each block and rewrites the key message into oral-style narration. Subsequently, these are then synthesized into audio and combined with the slide visuals to produce a time-aligned presentation video. Importantly, our pipeline is designed to be domain-adaptable and controllable, enabling broad applicability across document types and presentation styles.

Recognizing the need for rigorous evaluation of such complex multimodal outputs, we curate a test set of 30 human-authored document-video pairs spanning diverse domains, including education, finance, policy, and scientific communication. To comprehensively assess system performance, we further introduce a two-path evaluation strategy that combines fact-based comprehension assessment (via fixed multiple-choice quizzes) and preference-based scoring using vision-language models. This dual-pronged approach captures both objective correctness and subjective quality in video delivery.

Experiment results demonstrate that our method produces fluent, well-structured, and informative presentation videos, approaching human-level performance in both content delivery and viewer comprehension. These findings highlight the potential of combining language models, layout generation, and multimodal synthesis for creating explainable and scalable presentation systems from raw documents.

In general, our contributions are summarized as follows:

- We formulate and address the novel task of document-to-presentation video generation, which aims to produce narrated, slide-structured videos from long-form documents across diverse domains.
- We propose PresentAgent, a modular generation framework that integrates document parsing, layout-aware slide composition, narration planning, and audio-visual synchronization, enabling controllable and interpretable generation.
- We introduce PresentEval, a multi-dimensional evaluation framework powered by Vision-Language Models (VLMs), which scores videos along content, visual, and comprehension dimensions via prompt-based judging.
- We create a test set of 30 real-world document–presentation pairs and demonstrate through experiments and ablations that PresentAgent approaches human-level performance and significantly outperforms competitive variants.

2 Presentation Benchmark

The benchmark supports evaluation not only of fluency and fidelity, but also of downstream comprehension. Following the methodology introduced in Paper2Poster (Pang et al., 2025), we construct a quiz-style evaluation protocol (§5), where vision-language models are asked to answer factual content questions using only the generated video (slides + narration), simulating an audience's understanding. Human-authored videos are used as reference standards for both score calibration and upper-bound comparison. As shown in Figure 5, our benchmark encompasses four representative document types (academic papers, web pages, technical blogs, and slides) paired with human-authored videos, covering diverse real-world domains like education, research, and business reports.

We adopt a unified, model-based evaluation framework to assess the generated presentation videos. All evaluations are conducted using a vision-language model, guided by dimension-specific prompts tailored to different assessment objectives. The framework consists of two complementary components: (1) objective quiz evaluation, which measures factual accuracy through multiple-choice question answering; and (2) subjective scoring, which rates Content Quality, Visual or Audio Quality, and Comprehension Clarity on a 1–5 scale. Together, these metrics provide a comprehensive assessment of both the quality and informativeness

Figure 2: **Overview of our framework.** Our approach addresses the full pipeline of document-to-presentation video generation and evaluation. Left: Given diverse input documents—including papers, websites, blogs, slides, and PDFs—PresentAgent generates narrated presentation videos by producing synchronized slide decks with audio. Right: To evaluate these videos, we introduce PresentEval, a two-part evaluation framework: (1) Objective Quiz Evaluation (top), which measures factual comprehension using Qwen-VL; and (2) Subjective Scoring (bottom), which uses vision-language models to rate content quality, visual design, and audio comprehension across predefined dimensions.

of the generated videos.

2.1 Doc2Present Dataset

To support the evaluation of document to presentation video generation, we curate the Doc2Present Benchmark, a diverse dataset of document–presentation video pairs spanning multiple domains. Unlike prior benchmarks focused on research abstracts or slide generation, our dataset includes documents such as business reports, product manuals, policy briefs, and instructional texts, each paired with a human-crafted presentation video.We collect 30 high-quality video samples from public platforms, educational repositories, and professional presentation archives, further details regarding the data sources and statistical information of the dataset can be found in the appendix F.

2.2 PresentEval

To assess the quality of generated presentation videos, we adopt two complementary evaluation strategies: Objective Quiz Evaluation and Subjective Scoring, as shown in Figure 2. For each video, we provide the vision-language model with the complete set of slide images and the full narration transcript as a unified input—simulating how a real viewer would experience the presentation. In Objective Quiz Evaluation, the model answers a fixed set of factual questions to determine whether the video accurately conveys the key information from the source content. In Subjective Scoring, the model evaluates the video along three dimensions: the coherence of the narration, the clarity and design of the visuals, and the overall ease of understanding. All evaluations are conducted without ground-truth references and rely entirely on the model's interpretation of the presented content.

Objective Quiz Evaluation To evaluate whether a generated presentation video effectively conveys the core content of its source document, we use a fixed-question comprehension evaluation protocol. Specifically, we manually design five multiple-choice questions for each document, tailored to its content. These questions focus on key aspects such as topic recognition, structural understanding, and main argument extraction. As shown in Table 2, during evaluation, a vision-language model is given the video, including both visual frames and audio transcript, and asked to answer the five questions. Each question has four options, with one correct answer, annotated based on a human-created reference video. The final comprehension score (ranging from 0 to 5) reflects how many questions the model answered correctly, serving as a direct measure of how well the video communicates the original document.

Subjective Scoring To evaluate the quality of generated presentation videos, we adopt a prompt-based assessment using vision-language models. Instead of relying on human references or fixed metrics, we ask the model to evaluate each video from a viewer's perspective, using its own reasoning and preferences. The evaluation focuses on three aspects: coherence of narration, clarity and aesthetics of visuals, and overall ease of understanding. The model is shown the video and audio, and gives a score (1–5) with a brief explanation

for each aspect. This enables scalable, consistent, and human-aligned evaluation without manual references. As shown in Table 3, we design different prompts for different modalities and tasks to ensure targeted and effective assessment.

3 PresentAgent

To convert a long-form document into a narrated presentation video, we design a multi-stage generation framework that mirrors how human presenters prepare slides and talk tracks, as shown in Figure 3. Our method proceeds in four steps: segmenting the document into semantic units, composing slides with layout-aware structures, generating oral-style narration for each slide and assembling the visual and audio components into a synchronized video. This modular design supports controllability, interpretability, and multimodal alignment, enabling both high-quality generation and fine-grained evaluation. The following sections describe each component in detail.

3.1 Problem Formulation

Our method is designed to transform a long-form document into a structured presentation video through a multi-stage generation pipeline. We provide a formal description to highlight the key difference between our approach and conventional slide-based methods.

Conventional approaches often focus on generating slide elements S directly from a document chunk C, as in Equation 1, where each element includes text or image content, layout attributes, and visual style:

$$S = \{e_1, e_2, ..., e_n\} = f(C) \tag{1}$$

In contrast, we treat the entire document D as a globally structured input and generate a presentation in three steps: (1) a sequence of semantic segments $\{C_1, ..., C_K\}$ via outline planning, (2) a set of slides $\{S_1, ..., S_K\}$, each paired with a narrated audio track T_k generated by first producing a slide-specific script and then converting it to speech, and (3) a video V composed of visual and audio content aligned over time. This is defined as:

$$V = \texttt{Compose}(\{(S_1, T_1), ..., (S_K, T_K)\}) = g(D) \tag{2}$$

Rather than editing predefined templates or layouts, our system first identifies high-level structure in the document and then generates slide visuals and narration from scratch. This pipeline supports controllability, modular evaluation, and multimodal alignment for downstream comprehension and quality assessment.

3.2 Slide Planning and Composition

Our slide generation module is inspired by the editing-based paradigm proposed in PPTAgent (Zheng et al., 2025b), which formulates presentation construction as a structured editing process over HTML-like layouts. While PPTAgent focuses on producing editable `.pptx` slides, our goal is to generate visually coherent, narration-ready slide frames for downstream video synthesis. We re-implement the core idea in a self-contained pipeline tailored to multimodal synchronization.

We begin by segmenting the input document into coherent content blocks using a lightweight LLM-based parser. Each block is assigned a corresponding slide type such as bullet slide, figure-description, or title-intro, and matched with a predefined layout schema encoded in HTML. Unlike retrieval-based template matching, our system uses semantic and structural cues to map content to layout patterns in a rule-guided manner.

To populate the slide, we define a set of editable operations such as `replace_text`, `insert_image`, and `add_list`, which are applied to the layout structure. These instructions are generated by prompting a language model with the content block and layout constraints. Slides are then rendered into static visual frames using `python-pptx` or HTML-based renderers.

3.3 Narration and Audio Synthesis

To transform the static slides into an engaging presentation, we generate a spoken narration for each slide and synthesize it into audio. The process involves two components: narration script generation and text-to-speech synthesis.

For each content block corresponding to a slide, we prompt a language model to generate a concise, oral-style narration. The model is instructed to rewrite the key message of the slide into natural spoken language, avoiding dense text or technical jargon. We apply length control to ensure each narration falls within a target duration, typically between 30 and 150 seconds. Once the narration script is obtained, we synthesize the corresponding audio using a text-to-speech system. Each narration audio is paired with its slide and timestamped,

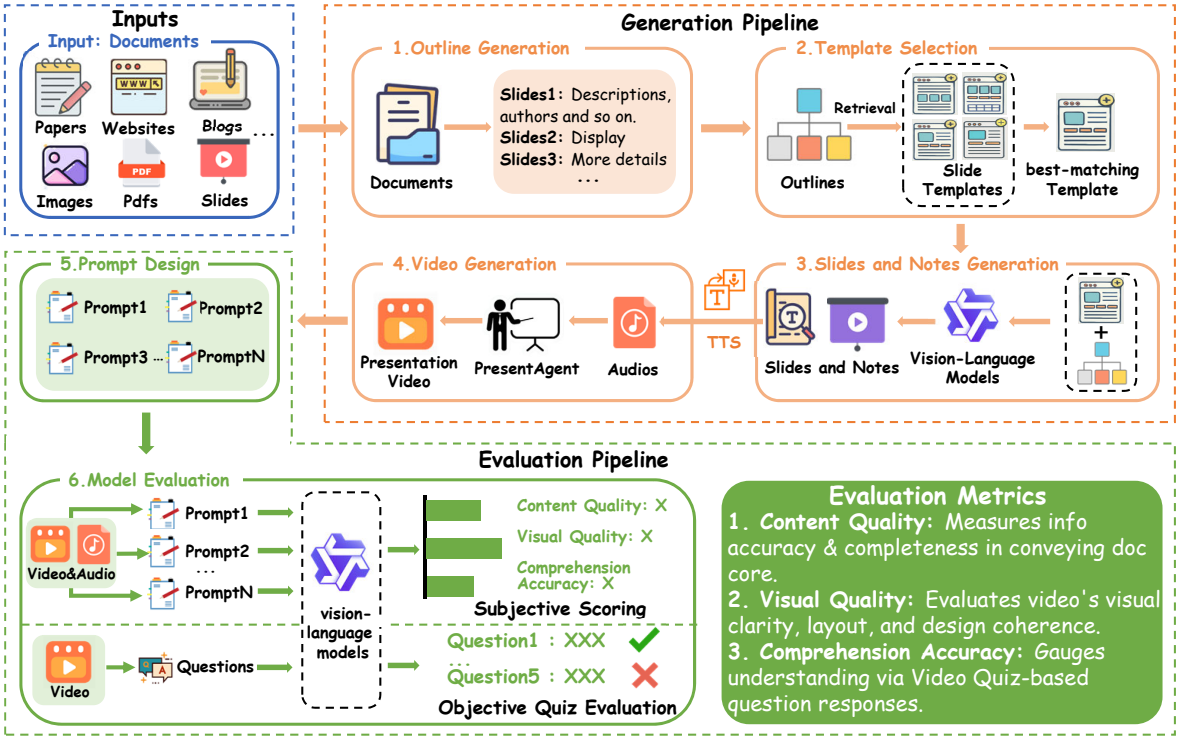


Figure 3: **Overview of the PresentAgent framework.** Our system takes diverse documents (e.g., papers, websites, PDFs) as input and follows a modular generation pipeline. It first performs outline generation (Step 1) and retrieves the most suitable template (Step 2), then generates slides and narration notes via a vision-language model (Step 3). The notes are converted into audio via TTS and composed into a presentation video (Step 4). To evaluate video quality, we design multiple prompts (Step 5) and feed them into a VLM-based scoring pipeline (Step 6) that outputs dimension-specific metrics.

forming the basis for synchronized video rendering in the next stage.

3.4 Video Assembly

In the final stage, we assemble the slide images and narration audio into a coherent, time-aligned presentation video. Each slide frame is displayed for the duration of its corresponding audio segment, with optional transitions between segments. We use video processing libraries such as `ffmpeg` to compose the visual and audio tracks. Each slide is rendered as a static frame, and the narration is added as synchronized voiceover audio. The output is a fully rendered video file in standard formats such as `.mp4`, suitable for presentation, sharing, or further editing. This stage completes the transformation from a raw document into a narrated, structured presentation video.

4 Experiments

We conduct experiments to evaluate the effectiveness of our proposed system in generating high-quality, narrated presentation videos. Given the novelty of the task, our focus is not on competing with existing baselines, but rather on assessing the performance of our full system relative to human-created presentations. Comprehension accuracy is determined based on performance in the PresentEval task. Evaluation setup can be found in appendix H.

4.1 Main Results

Table 1 presents evaluation results, covering both factual comprehension (Quiz Accuracy) and preference-based quality scores for video and audio outputs. In terms of quiz accuracy, most PresentAgent variants perform comparably to or better than the human reference (0.56), with Claude-3.7-sonnet (Anthropic, 2024) achieving the highest accuracy at 0.64, suggesting strong alignment between the generated content and the source document. Other models such as Qwen-VL-Max (Bai et al., 2025) and Gemini-2.5-flash (DeepMind, 2024) scored slightly lower (0.52), indicating room for improvement in factual grounding.

In terms of subjective quality, human-created

Method	Model	Quiz Accuracy	Video Score				Audio Score			
			Content	Visual	Comp.	Mean	Content	Audio	Comp.	Mean
Human	Human	0.56	4.0	4.6	4.8	4.47	4.8	4.6	5.0	4.80
PresentAgent	Claude-3.7-sonnet	0.64	4.0	4.0	4.0	4.00	4.2	4.6	4.8	4.53
PresentAgent	Qwen-VL-Max	0.52	4.2	4.8	4.4	4.47	4.6	4.2	5.0	4.60
PresentAgent	Gemini-2.5-pro	0.52	4.2	4.4	4.4	4.33	4.2	4.0	4.8	4.33
PresentAgent	Gemini-2.5-flash	0.52	4.2	5.0	3.8	4.33	4.2	4.2	4.8	4.40
PresentAgent	GPT-4o-Mini	0.64	4.8	4.6	4.6	4.67	4.0	4.4	4.8	4.40
PresentAgent	GPT-4o	0.56	4.0	4.2	3.6	3.93	4.2	4.4	4.8	4.47

Table 1: Detailed evaluation results on the 5-document test set. Fact-based evaluation includes accuracy on five fixed quiz questions (Q1–Q5). Preference-based evaluation includes 1–5 scale scores for content fidelity, visual design, and overall clarity. Each Quality Score group has a calculated mean column.

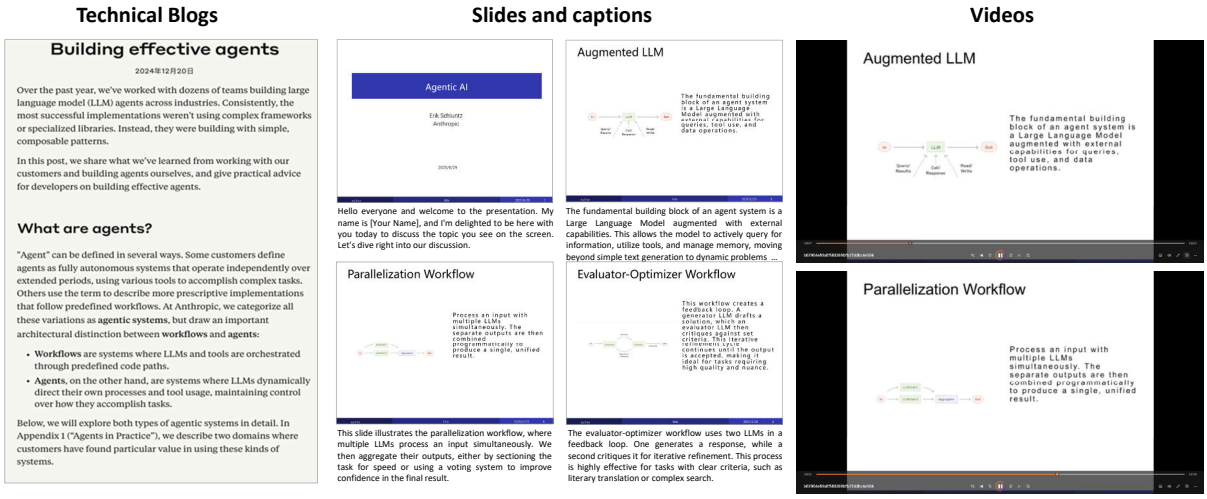


Figure 4: **PresentAgent Demo.** Automatically generates academic-style slides and narrated videos from research papers, streamlining the transformation from written content to engaging visual presentations.

presentations still lead with the highest video and audio scores overall. However, several PresentAgent variants show competitive performance. For example, GPT-4o-Mini (Achiam et al., 2023) achieves top scores in video content and visual appeal (both at or near 4.8), while Claude-3.7-sonnet (Anthropic, 2024) delivers the most balanced audio quality (mean 4.53). Interestingly, Gemini-2.5-flash (DeepMind, 2024) scores highest in visual quality (5.0) but lower in comprehension, reflecting a trade-off between aesthetics and clarity. These results highlight the effectiveness of our modular pipeline and the usefulness of our unified PresentEval framework in capturing diverse aspects of presentation quality.

4.2 Analysis

Figure 4 Presents a full example of a PresentAgent-auto-generated presentation video, showing a technical blog turned into a narrated presentation. The system identifies structural segments (e.g., introduction, technical explanations) and generates slides with oral-style captions and synchronized speech, covering topics like "parallelization workflow" and "agent system architecture" to demonstrate its ability to keep technical accuracy while delivering content clearly and conversationally.

5 Conclusion

In conclusion, we presented PresentAgent, a modular system for transforming long-form documents into narrated presentation videos. By addressing the challenges of slide planning, narration synthesis, and synchronized rendering, PresentAgent enables structured, controllable, and reusable multimodal outputs. To evaluate this novel task, we introduced a diverse benchmark and proposed complementary factual and preference-based metrics. Experimental results show that PresentAgent generates coherent, engaging, and informative presentations, approaching human quality. This work lays the groundwork for automated, explainable content

generation and opens new directions for research in multimodal communication across education, business, and accessibility.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Anthropic. 2024. Claude 3 technical overview. https://www.anthropic.com/news/claude-3. Accessed: 2025-06-30.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Jiaao Chen and Diyi Yang. 2021. Structure-aware abstractive conversation summarization via discourse and action graphs. *arXiv preprint arXiv:2104.08400*.

Google DeepMind. 2024. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. https://deepmind.google/technologies/gemini/. Accessed: 2025-06-30.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, and 1 others. 2025. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*.

Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. 2022. Doc2ppt: Automatic presentation slides generation from scientific documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 634–642.

Jiaxin Ge, Zora Zhiruo Wang, Xuhui Zhou, Yi-Hao Peng, Sanjay Subramanian, Qinyue Tan, Maarten Sap, Alane Suhr, Daniel Fried, Graham Neubig, and Trevor Darrell. 2025. Autopresent: Designing structured visuals from scratch. *arXiv preprint arXiv:2501.00912*.

Yingqing He, Menghan Xia, Haoxin Chen, Xiaodong Cun, Yuan Gong, Jinbo Xing, Yong Zhang, Xintao Wang, Chao Weng, Ying Shan, and 1 others. 2023. Animate-a-story: Storytelling with retrieval-augmented video generation. *arXiv preprint arXiv:2307.06940*.

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. 2023. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15954–15964.

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*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and 1 others. 2024. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Xin Li, Wenqing Chu, Ye Wu, Weihang Yuan, Fanglong Liu, Qi Zhang, Fu Li, Haocheng Feng, Errui Ding, and Jingdong Wang. 2023. Videogen: A reference-guided latent diffusion approach for high definition text-to-video generation. *arXiv preprint arXiv:2309.00398*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Qinchen Wu, Mingyi Yan, Zhengyuan Yang, Lijuan Wang, and Mike Zheng Shou. 2024a. Videogui: A benchmark for gui automation from instructional videos. *arXiv preprint arXiv:2406.10227*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024b. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. 2025. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*.

Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. 2025. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. *arXiv preprint arXiv:2503.15661*.

Junrui Ni, Liming Wang, Heting Gao, Kaizhi Qian, Yang Zhang, Shiyu Chang, and Mark Hasegawa-Johnson. 2022. Unsupervised text-to-speech synthesis by unsupervised automatic speech recognition. *arXiv preprint arXiv:2203.15796*.

Wei Pang, Kevin Qinghong Lin, Xiangru Jian, Xi He, and Philip Torr. 2025. Paper2poster: Towards multimodal poster automation from scientific papers. *arXiv preprint arXiv:2505.21497*.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. 2021. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International conference on machine learning*, pages 8599–8608. PMLR.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, and et al. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Shivam R Solanki and Drupad K Khublani. 2024. From script to screen: Unveiling text-to-video generation. In *Generative Artificial Intelligence: Exploring the Power and Potential of Generative AI*, pages 81–112. Springer.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, and 1 others. 2024. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. *arXiv preprint arXiv:2412.19723*.

Hideyuki Tachibana, Katsuya Uenoyama, and Shunsuke Aihara. 2018. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4784–4788. IEEE.

Baode Wang, Biao Wu, Weizhen Li, Meng Fang, Yanjie Liang, Zuming Huang, Haozhe Wang, Jun Huang, Ling Chen, Wei Chu, and 1 others. 2025. Infinity parser: Layout aware reinforcement learning for scanned document parsing. *arXiv preprint arXiv:2506.03197*.

Guanghua Wang, Priyanshi Garg, and Weili Wu. 2024a. Segmented summarization and refinement: A pipeline for long-document analysis on social media. *Journal of Social Computing*, 5(2):132–144.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, and 1 others. 2024b. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, and 1 others. 2024c. Opendevin: An open platform for ai software developers as generalist agents. *arXiv preprint arXiv:2407.16741*.

Yuan Wang, Di Huang, Yaqi Zhang, Wanli Ouyang, Jile Jiao, Xuetao Feng, Yan Zhou, Pengfei Wan, Shixiang Tang, and Dan Xu. 2024d. Motiongpt-2: A general-purpose motion-language model for motion generation and understanding. *arXiv preprint arXiv:2410.21747*.

Biao Wu, Yanda Li, Meng Fang, Zirui Song, Zhiwei Zhang, Yunchao Wei, and Ling Chen. 2024. Foundations and recent trends in multimodal mobile agents: A survey. *arXiv preprint arXiv:2411.02006*.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. 2024. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, and 1 others. 2025. Qwen2. 5-omni technical report. *arXiv preprint arXiv:2503.20215*.

Qiyao Xue, Xiangyu Yin, Boyuan Yang, and Wei Gao. 2025. Phyt2v: Llm-guided iterative self-refinement for physics-grounded text-to-video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18826–18836.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024a. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, and 1 others. 2024b. If llm is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *arXiv preprint arXiv:2401.00812*.

Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023a. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36:71995–72007.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. 2023b. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, and 1 others. 2024c. Cogvideox: Text-to-video diffusion

models with an expert transformer. *arXiv preprint arXiv:2408.06072*.

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*.

Murong Yue, Wenlin Yao, Haitao Mi, Dian Yu, Ziyu Yao, and Dong Yu. 2024. Dots: Learning to reason dynamically in llms via optimal reasoning trajectories search. *arXiv preprint arXiv:2410.03864*.

Zeyu Zhang, Yiran Wang, Biao Wu, Shuo Chen, Zhiyuan Zhang, Shiya Huang, Wenbo Zhang, Meng Fang, Ling Chen, and Yang Zhao. 2024. Motion avatar: Generate human and animal avatars with arbitrary motion. *arXiv preprint arXiv:2405.11286*.

Hao Zheng, Xinyan Guan, Hao Kong, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2025a. Pptagent: Generating and evaluating presentations beyond text-to-slides. *arXiv preprint arXiv:2501.03936*.

Hao Zheng, Xinyan Guan, Hao Kong, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2025b. Pptagent: Generating and evaluating presentations beyond text-to-slides. *arXiv preprint arXiv:2501.03936*.

Zixiang Zhou, Yu Wan, and Baoyuan Wang. 2024. Avatargpt: All-in-one framework for motion understanding planning generation and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1357–1366.

A Related Work

A.1 Document-to-Multimodal Generation

Recent advances in large language models (LLMs) and multimodal generation have sparked growing interest in converting documents into diverse output formats, such as slides, posters, or audio summaries (Xu et al., 2025; Wang et al., 2025; Pang et al., 2025; Sun et al., 2024). Systems like PPTAgent (Zheng et al., 2025b) and Doc2PPT (Fu et al., 2022) treat document-to-slide generation as a structured summarization problem, focusing on layout-aware slide construction. Other works, such as Paper2Poster (Pang et al., 2025) extend this idea by producing single-page visual summaries using layout planning and visual feedback. However, these systems typically generate static outputs and do not model time-dependent delivery such as narration or slide progression. Our work builds upon these foundations, but further introduces temporal planning and audio-visual synchronization, enabling the generation of fully narrated presentation videos.

A.2 Vision-Language Agents

Recent advances have highlighted the expanding capabilities of vision language models (VLMs) beyond traditional language understanding. Techniques such as ReAct (Yao et al., 2023; Yang et al., 2023b; Yue et al., 2024) have shown that LLMs can operate as autonomous agents, capable of step-by-step reasoning and dynamic interaction through code execution (Wang et al., 2024c; Yang et al., 2024a,b), API function calls (Schick et al., 2023; Lu et al., 2025; Yang et al., 2023a), user interface manipulation (Lin et al., 2024b; Qin et al., 2025; Nayak et al., 2025; Wu et al., 2024), and motion generation (Zhang et al., 2024; Zhou et al., 2024; Wang et al., 2024d). Despite these developments, general-purpose agents still struggle with professional tasks that demand accuracy, domain-specific knowledge, and reliable interaction (Lin et al., 2024a). A closely related area is slide automation (Ge et al., 2025; Zheng et al., 2025a), which agents translate short text prompts into executable Python code to render presentation slides. In contrast, our proposed presentation video generation task is significantly more challenging: instead of taking a short prompt as input, the system processes an entire long-form document—such as a research paper, product manual, or technical report—and produces a well-structured presentation video with oral-style narration. This task imposes higher demands on content understanding, multimodal alignment, speech generation, and video synthesis. To address these challenges, we design a generation pipeline along with an automatic evaluation framework to systematically assess the generated videos in terms of information delivery, visual quality, and overall comprehensibility.

B Implementation Details

PresentAgent adopts a highly modular multimodal-generation architecture. At the language-understanding and generation layer, we run six primary LLM back ends in parallel—GPT-4o, GPT-4o-mini, Qwen-VL-Max, Gemini-2.5-Flash, Gemini-2.5-Pro, and Claude-3.7-Sonnet—and select or ensemble them on-the-fly with a dynamic routing policy that weighs input length, conversational complexity, and latency budget. For visual-language evaluation, we introduce the lightweight VLM Qwen-VL-2.5-3B-Instruct to score slide layout, chart readability, and cross-modal consistency, feeding its self-critique back into generation. Speech synthesis is unified on MegaTTS3, which outputs 24 kHz, 16-bit high-fidelity narration and supports prosody-tag controls for fine-grained rate, pitch, and emotion adjustment.

The experimental pipeline converts any input document—PDF, Markdown, DOCX, or web snapshot through three automated stages:

1. Structured parsing & re-ordering that maps content to a hierarchical topic–subtopic tree.

2. Per-slide generation with the chosen LLM, producing a PowerPoint deck containing titles, bullet points, graphic placeholders, and Alt-Text, while retrieving and inserting relevant images for key nouns.

3. Synchronized narration generation with MegaTTS3 in Chinese or English, followed by an FFmpeg script that assembles a 1080 p video with fade-in/out transitions and optional captions.

C Discussion

In this work, we synthesized presentation-style videos that integrate visual slides, textual narration, and spoken audio, simulating realistic multimodal communication scenarios. While our current evaluation focuses on the individual quality of each modality—such as visual clarity, textual relevance, and audio intelligibility—these dimensions are treated independently. However, in real-world

applications, the effectiveness of communication often hinges on the semantic and temporal coherence across modalities.

Future research should thus move beyond isolated assessments and aim toward fusion-aware understanding and evaluation. This entails not only modeling the interactions and alignment among image, audio, and text modalities, but also enabling the system to reason over their combined meaning. Existing models like ImageBind offer a unified embedding space for multiple modalities, but lack the capacity for high-level inference and semantic comprehension.

A promising direction lies in bridging representation alignment with multimodal reasoning, by integrating aligned modality encoders with powerful language models. This would allow the system to jointly perceive, interpret, and respond to complex multimodal inputs—such as explaining a visual concept based on both audio narration and visual cues, or identifying inconsistencies across modalities. Developing such reasoning-capable, fusion-aware models will be critical for advancing robust, coherent multimodal understanding in real-world applications.

D Limitations

Our work faces two key constraints: (1) Due to the high computational costs of commercial LLM/VLM APIs (e.g., GPT-4o and Gemini-2.5-Pro), evaluation was limited to five academic papers, potentially underrepresenting the document diversity shown in our benchmark (Figure 5); (2) PresentAgent currently generates static slides without dynamic animations/effects due to architectural constraints in video synthesis and trade-offs between generation speed and visual quality, as noted in ChronoMagic-Bench's temporal coherence studies. Future improvements could involve lightweight distillation models and physics-aware rendering engines.

E Evaluation Benchmark

As Shown in Figure 5, we showcase four of the representative document types in our benchmark: academic papers, web pages, technical blogs, and presentation slides. These documents cover a broad spectrum of real-world content domains, such as educational tutorials, research briefs, product manuals, scientific articles, news commentary, and business reports. Each document is paired with a manually authored presentation video, providing a diverse and realistic testbed for evaluating document-to-video generation systems in terms of multimodal coherence, content preservation, and presentation quality.

F Doc2Present Dataset Details

Data Source. We collect 30 high-quality video samples from public platforms, educational repositories, and professional presentation archives. Each video follows a structured narration format, combining slide-based visuals with synchronized voiceover. We manually align each video with its source document and ensure the following conditions are met: (1) the content structure of the video follows that of the document; (2) the visuals convey document information in a compact, structured form; and (3) the narration and slides are well-aligned temporally.

Data Statistics. The average document length is 3,000–8,000 words, while the corresponding videos range from 1 to 2 minutes and contain 5-10 slides. This setting highlights the core challenge of the task: transforming dense, domain-specific documents into effective and digestible multimodal presentations.

G PresentEval

G.1 Prompts of Objective Quiz Evaluation

Table 2 presents the prompting content for the evaluation method utilizing objective quiz-based assessment. Each set of questions included in this evaluation is crafted manually, with its creation firmly rooted in the actual content of the relevant documents. The formulation of these questions places a distinct emphasis on three key aspects: topic recognition, which involves the ability to accurately identify and grasp the central themes of the source material; structural understanding, referring to the comprehension of the organizational framework and logical arrangement of the document; and key argument identification, focusing on the capacity to pinpoint the core viewpoints and supporting arguments within the content. These carefully designed questions serve as a means to evaluate the extent to which the generated video successfully conveys the essential information, core ideas, and structural logic of the original source material, thereby assessing the effectiveness of the video in communicating the source content.

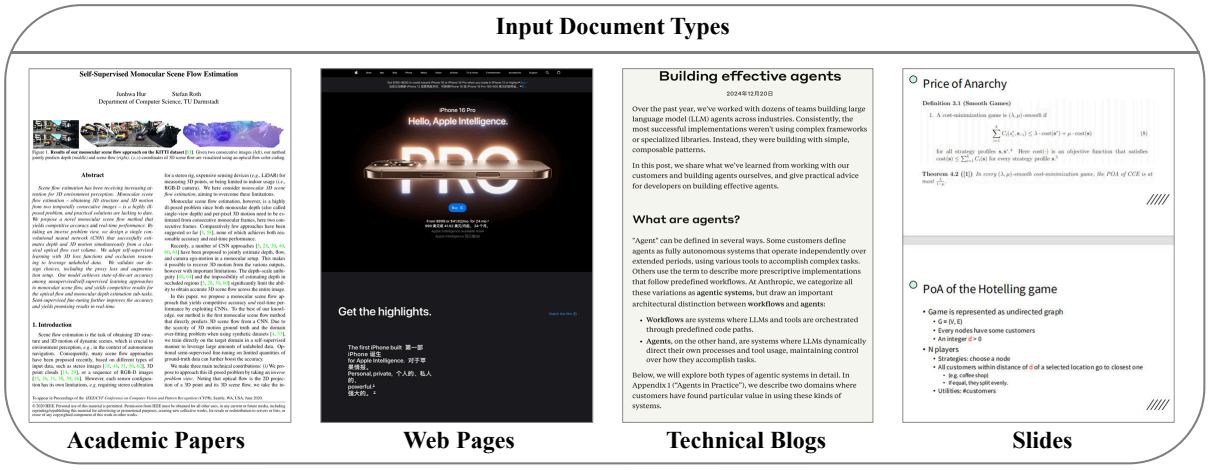


Figure 5: Document Diversity in Our Evaluation Benchmark.

Prensentation of Web Pages	What is the main feature highlighted in the iPhone's promotional webpage?
A.	A more powerful chip for faster performance
B.	A brighter and more vibrant display
C.	An upgraded camera system with better lenses
D.	A longer-lasting and more efficient battery
Prensentation of Academic Paper	What primary research gap did the authors aim to address by introducing the FineGym dataset?
A.	Lack of low-resolution sports footage for compression studies
B.	Need for fine-grained action understanding that goes beyond coarse categories
C.	Absence of synthetic data to replace human annotations
D.	Shortage of benchmarks for background context recognition

Table 2: Prompt of evaluation via Objective Quiz Evaluation. Each question set is manually created based on the actual document content, with a focus on topic recognition, structural understanding, and key argument identification. These questions evaluate how well the generated video communicates the source material.

G.2 Prompts of Subjective Scoring

Prompt of evaluation via subjective scoring is shown in table 3. This table showcases the prompting content employed in the subjective scoring-based evaluation approach. Each individual prompt within this set is precisely targeted at a specific evaluative dimension. These dimensions encompass narrative coherence, which pertains to the logical flow and consistency of the storytelling; visual appeal and audio appeal, focusing on the attractiveness and engaging nature of the visual elements and audio components respectively; and comprehension difficulty, referring to the level of ease or challenge in understanding the presented content. These prompts are meticulously designed to serve as a guiding framework for vision-language models, enabling them to assess presentations from a human-centric perspective. This means that the evaluation aligns with human perceptions, preferences, and ways of understanding, ensuring that the assessment results are more in line with how humans would judge the quality of the presentations.

H Evaluation Setup

We construct a test set consisting of 30 long-form documents, each paired with a manually created presentation video that serves as a human-level reference. These documents span a diverse range of topics, including education, product explanation, research overviews, and policy briefings. For each document, we generate a corresponding presentation video using our full generation pipeline.

All videos, both human-created and machine-generated, are evaluated using our unified evaluation framework, PresentEval. Each synthesized video is approximately two minutes in length. However, due to the current lack of a single multimodal model capable of jointly assessing visual and audio quality for videos longer than two minutes, we adopt a split evaluation strategy.

In the Objective Quiz stage, we use Qwen-VL-2.5-3B (Wang et al., 2024b) to evaluate the accuracy of the entire video using a fixed set of multiple-choice comprehension questions. In the Subjective Scoring stage, we extract short video/audio

Video	**Scoring Prompt**
Narr. Coh.	*"How coherent is the narration across the video? Are the ideas logically connected and easy to follow?"*
Visual Appeal	*"How would you rate the visual design of the slides in terms of layout, aesthetics, and overall quality?"*
Comp. Diff.	*"How easy is it to understand the presentation as a viewer? Were there any confusing or contradictory parts?"*
Audio	**Scoring Prompt**
Narr. Coh.	*"How coherent is the narration throughout the audio? Are the ideas logically structured and easy to follow?"*
Audio Appeal	*"How pleasant and engaging is the narrator's voice in terms of tone, pacing, and delivery?"*
Comp. Diff.	*"How easy is it to understand the spoken content? Were there any unclear or confusing parts in the audio?"*

Table 3: Prompt of evaluation via Subjective Scoring. Each prompt targets a specific dimension—narrative coherence, visual/audio appeal, or comprehension difficulty—and is designed to guide vision-language models in assessing presentations from a human-centric perspective. Abbreviations: Narr. Coh. = Narrative Coherence; Comp. Diff. = Comprehension Difficulty.

segments and evaluate them individually to assess quality in a more focused and scalable manner, using Qwen-Omni-7B (Xu et al., 2025).

Both models are guided by dimension-specific prompts and score each video or audio sample along three axes: Content Quality, Visual Quality, and Comprehension Accuracy.

PromptSculptor: Multi-Agent Based Text-to-Image Prompt Optimization

Dawei Xiang[1], Wenyan Xu [2], Kexin Chu [1], Tianqi Ding [3], Zixu Shen [1], Yiming Zeng [1], Jianchang Su [1], Wei Zhang[1]*,

[1]University of Connecticut, [2] Central University of Finance and Economics, [3] Baylor University

[1] {ieb24002, kexin.chu, qzt24001, yiming.zeng, jianchang.su, wei.13.zhang} @uconn.edu,

[2] 2022211032@email.cufe.edu.cn, [3] kirk_ding1@baylor.edu

Abstract

The rapid advancement of generative AI has democratized access to powerful tools such as Text-to-Image (T2I) models. However, to generate high-quality images, users must still craft detailed prompts specifying scene, style, and context—often through multiple rounds of refinement. We propose PromptSculptor, a novel multi-agent framework that automates this iterative prompt optimization process. Our system decomposes the task into four specialized agents that work collaboratively to transform a short, vague user prompt into a comprehensive, refined prompt. By leveraging Chain-of-Thought (CoT) reasoning, our framework effectively infers hidden context and enriches scene and background details. To iteratively refine the prompt, a self-evaluation agent aligns the modified prompt with the original input, while a feedback-tuning agent incorporates user feedback for further refinement. Experimental results demonstrate that PromptSculptor significantly enhances output quality and reduces the number of iterations needed for user satisfaction. Moreover, its model-agnostic design allows seamless integration with various T2I models, paving the way for industrial applications.

1 Introduction

The rapid development of large-scale models since 2022 has introduced Generative AI tools to a wide audience. Text-to-Image (T2I) models (Zhang et al., 2023) like Midjourney and Large Language Models (LLMs) such as ChatGPT now produce astonishing, often human-level creative outputs. However, effectively harnessing these technologies typically requires subtle and specific prompting. For T2I models, for instance, prompts must include precise details on scene composition, style parameters, and even technical jargon, creating a high entry barrier for new AI users (Mahdavi Goloujeh et al., 2024). This challenge has led to the emergence of specialized professionals—"Midjourney Artists"—who need to optimize customers' brief inputs into detailed, high-quality prompts through iterative refinement based on generated outputs and user feedback.

Previous works (Cao et al., 2023; Feng et al., 2023; Mo et al., 2024) have tried to build a system to automate this prompt engineering workflow. But there are significant challenges for the system. A primary challenge is inferring user intent: initial prompts are often short and vague, peppered with abstract metaphors. For example, "draw a painting as a birthday blessing for my friend, he is like a lion" might lead a T2I model to interpret "lion" literally as a real fierce lion rather than as a symbol of confidence and courage. The second challenge is enriching these sparse inputs with detailed scene and background descriptions (Mahdavi Goloujeh et al., 2024). This requires the system to associate abstract concepts with concrete visual elements. Besides T2I models usually cannot fully satisfy user requirements in a single generation attempt. Therefore, prompts must be adjusted iteratively based on user feedback after observing initial results.

However previous works have primarily focused on parameter editing while neglecting to enrich prompts with detailed background and scene descriptions. As a result, the generated outputs often remain too similar to the original, lacking the necessary contextual depth. Moreover, these approaches typically do not address the challenge of interpreting abstract concepts—if a prompt contains hard-to-visualize terms, the generated image may fail to capture the user's true intent. In addition, most methods lack an iterative mechanism for updating the prompt based on feedback after the generation process.

To address these challenges, we propose a multi-agent system (MAS) that decomposes the task

*Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 774–786
November 4-9, 2025

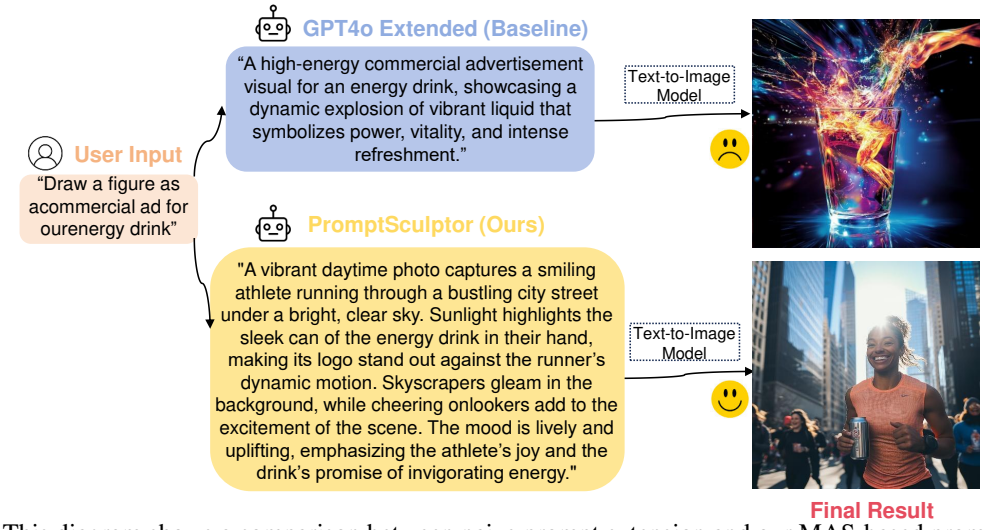


Figure 1: This diagram shows a comparison between naive prompt extension and our MAS-based prompt optimization. Our model successfully depicts a commercial ad scene to propaganda the drink which better aligns with user's intent.

of prompt optimization into specialized functions. Our system includes four primary agents: 1) Intent Inference Agent; 2) Scene and Style Agent; 3) Feedback and Tuning Agent ; 4) Self-Evaluation Agent.

The core advantage of our method lies in its multi-agent architecture, which significantly enhances language understanding over traditional single-agent systems. First, our Intent Inference Agent decomposes brief and ambiguous inputs by breaking down abstract terms into detailed concepts and specific objects. This refined information is then passed to the Scene and Style Agent, which enriches the prompt with detailed scene, background, and style descriptions by linking these concepts to concrete visual elements. The Self-Evaluation Agent and the Feedback and Tuning Agent further refine the prompt until the final output meets expectations. Since our system is built on LLM agents and leverages user feedback, it can seamlessly transfer across different T2I models without requiring additional fine-tuning.

Our system also integrate Chain-of-Thought (CoT) reasoning (Wei et al., 2022) into user intent understanding and scene enrichment process. which requires agent to provides transparent, step-by-step rationales for each enrichment stage. This decomposition process increase the agent's ability to understand complex terms and requirements.

Our innovation and contributions are as follows:

- We propose a novel multi-agent framework that decomposes the complex T2I prompt optimization task into specialized agents, each handling a distinct subtask. This simplifies the process and significantly improves generation quality and flexibility compared to prior single-agent systems (Cao et al., 2023; Mo et al., 2024). To the best of our knowledge, this is the first use of a MAS for T2I prompt optimization, achieving state-of-the-art performance as shown in Table 1.
- Our system features a self-evaluation agent based on a Vision-Language Model (VLM) for semantic alignment, paired with a feedback-tuning agent. This feedback loop iteratively refines prompts until the generated image matches the user's intent, reducing the number of required runs, as demonstrated in Table 2. This is the first introduction of such a feedback mechanism in T2I prompt optimization.
- Our framework is model-agnostic and works across various T2I models (e.g., Midjourney, DALL·E 3, Stable Diffusion) without model-specific tuning, ensuring strong scalability and broad applicability.

2 Related Work

2.1 Prompt optimization

Prompt optimization has gained attention for enhancing LLMs without parameter fine-tuning. Early research focused on white-box models like AutoPrompt(Shin et al., 2020) and Fluent-Prompt(Shi et al.), which optimize by accessing embeddings or logits. With the rise of closed-

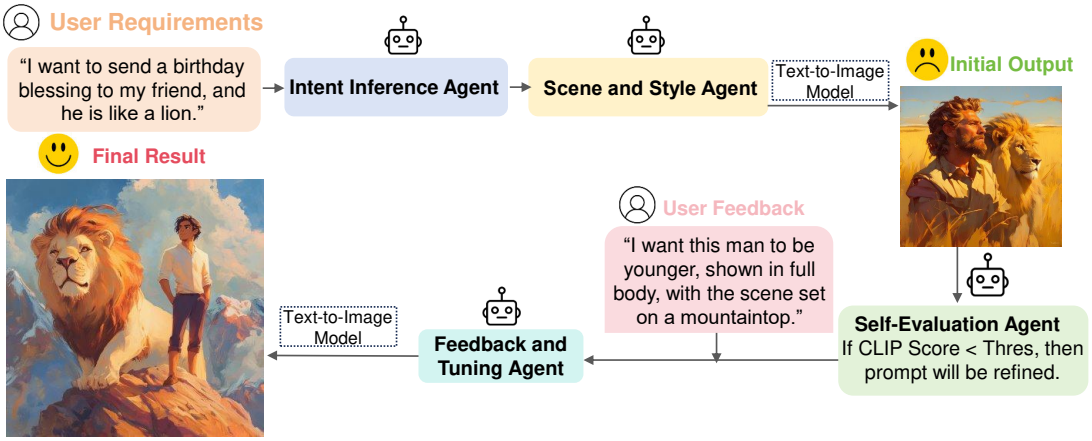


Figure 2: This diagram shows the PromptSculptor model's process, where the user's prompt is analyzed, enriched, and refined by different agents. The system optimizes the prompt and generates an image that aligns with the user's intent, adjusting based on feedback and evaluation scores.

source models like GPT(Brown et al., 2020), research shifted toward black-box optimization. While methods like BBTv2(Sun et al., 2022), and Clip-Tuning still rely on embeddings, recent approaches like GRIPS(Prasad et al., 2023) and APO(Pryzant et al., 2023) optimize by editing and selecting candidate prompts. Other methods use evolutionary algorithms, reinforcement learning, or planning strategies, such as EvoPrompt(Guo et al., 2025), Promptbreeder(Fernando et al., 2024), BDPL(Diao et al.), and PromptAgent(Wang et al.). APE(Zhou et al.) generates candidate prompts with LLMs, and OPRO(Zhang et al., 2024) frames optimization as a black-box problem solved by LLMs. OPT2I(Mañas et al.) refines prompts to improve consistency. Some studies, like InstructZero(Chen et al., 2024b) and INSTINCT(Lin et al.), treat prompt optimization as a continuous problem, while ZOPO(Hu et al., 2024) enhances performance through zero-order optimization. Recently, DPO(Rafailov et al., 2023) and SLiC(Khorasgani et al., 2022) bypass reinforcement learning using preference datasets. Unlike these methods, this paper proposes a plug-and-play multi-agent system for prompt optimization, without fine-tuning.

2.2 Multi-agent system

Multi-agent system works by coordinating multiple large language models to finish a complex task. Recent advances in large language models have catalyzed multi-agent paradigms into dynamic coordination (Chen et al., 2024a; Guo et al., 2024; Leong and Wu, 2024), which has being explored in multiple areas including text and code generation (Zhang et al., 2025b; Zeng et al., 2025b), data labeling (Lu et al., 2025), bioinformatics analysis (Wu et al., 2024), financial forecasting(Xu et al., 2025b,c,a; Zhang et al., 2025a), autonomous driving (Zeng et al., 2025a) and privacy protection (Chu et al., 2025b,a). Building on these trends, we position our approach as a plug-and-play multi-agent prompt optimization framework that operates at the prompt level without fine-tuning, and is compatible with closed-source or API-only models.

2.3 Improve the Consistency of T2I Model

In recent years, some studies have focused on using cross-attention to guide LLMs in generating prompts that better align with user intent(Feng et al.; Epstein et al., 2023; Wu et al., 2023a; Chefer et al., 2023). Other research transforms text prompts into layouts, which are then fed into layout-to-image generation models(Cho et al., 2023; Lian et al.). Some approaches fine-tune T2I models using human feedback(Lee et al., 2023; Wu et al., 2023b; Wallace et al., 2024), AI model feedback(Sun et al.), or image selection(Karthik et al., 2023). Unlike these methods, our approach doesn't require model fine-tuning but uses a correction agent to check if the input prompt aligns with the natural language query, operating only at the prompt level. This allows our method to work with more T2I models, including those only accessible via API as Midjourney (Midjourney, 2024).

3 Methodology

In this section, we introduce the design of our model, PromptSculptor, which comprises three main stages: (1) a MAS for initial prompt optimization, (2) VLM-based prompt alignment and (3) a feedback-based tuning agent. Our system is intended for users with limited experience in prompt engineering, where the initial T2I prompt may be vague or too simplistic. To address this, our system first analyzes the user's intent to automatically enrich and detail the prompt. It then evaluates whether the generated image aligns with the user's intent and, if necessary, further refines the prompt using a pre-trained VLM.

3.1 Multi-Agent System Overview

As shown in Figure 2, our system consists of four specialized agents:

1. **Intent Inference Agent**: Extracts the user's core idea and missing details.
2. **Scene and Style Agent**: Builds on that intent to craft a vivid, detailed scene.
3. **Self-Evaluation Agent**: Generates an image, checks its CLIP score against the original intent, and fine-tunes the prompt if the score is too low.
4. **Feedback and Tuning Agent**: Incorporates user feedback to further refine the prompt and close any remaining gaps.

3.2 Intent Inference Agent

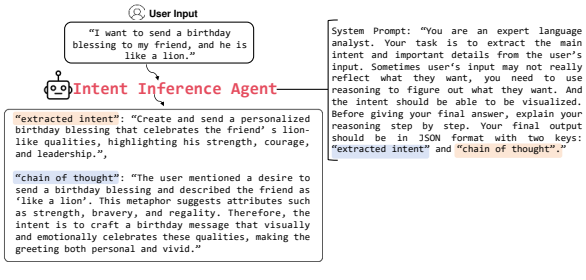


Figure 3: Prompt display of the Intent Inference Agent.

The primary function of the Intent Inference Agent is to perform a deep analysis of the user's input, capturing not only the explicit request but also the latent, nuanced intent embedded within brief or ambiguous prompts. As shown in Figure 3, our system prompt is meticulously designed to guide the model to probe beyond surface-level text—extracting implicit cues, contextual hints, and even emotional undertones.

To ensure transparency and enhance interpretability, the agent is equipped with a CoT mechanism. This mechanism prompts the agent to document its reasoning process step by step. For example, when processing a prompt like "A birthday painting for a friend who is like a lion," the agent's CoT may detail steps such as identifying the explicit elements ("birthday painting" and "friend"), interpreting "lion" as a metaphor for qualities like strength, majesty, or courage, and synthesizing these insights to reveal the underlying intent. This comprehensive extraction of both overt and hidden intentions, supported by its explicit CoT, forms the foundation for further prompt enrichment in our multi-agent pipeline.

3.3 Scene and Style Agent

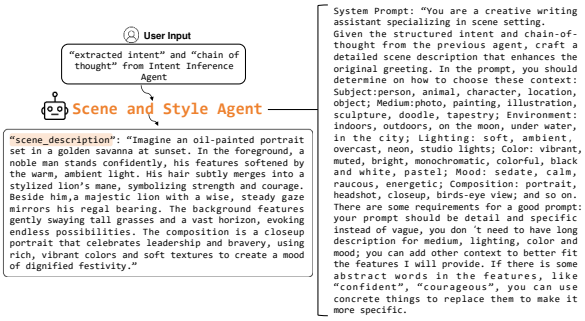


Figure 4: Prompt display of the Scene and Style Agent.

After Intent Inference Agent has detailed the user's intent and context, the second step is for Scene and Style Agent to set up the scene and enrich the detail in the figure. As shown in Figure 4 this agent will try to visualize the abstract concept in the previous prompt and use detail object to visualize them to make the figure more vivid. This is exactly what human artists are doing: they use some detailed object to represent abstract things to make them more impressive and easy to understand.

When artist creates a painting, there are several factors they need to consider including the subjects, medium, environment, lighting, color, mood composition and so on. We all instruct this agent to consider all these factors based on the previous inference. If without the inference and context explanation from the previous agent, it may be more difficult for the Scene and Style Agent to generate an accurate settings. This shows the advantage of our MAS.

3.4 Self-Evaluation agent

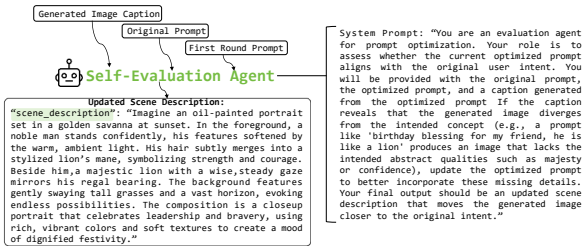


Figure 5: Prompt display of the Self-Evaluation Agent.

Although the Scene and Style Agent enriches the scene description, the inherent vagueness of the original input may still result in generated images that do not fully align with the user's intent. To address this gap, we designed a self-evaluation agent to further refine the prompt if needed.

Our self-evaluation agent serves as a quality assurance module within our multi-agent prompt optimization pipeline. Its primary role is to verify that the generated image accurately reflects the user's original intent. To achieve this, the agent first computes a CLIP similarity score (Patashnik et al., 2021) between the generated image and the original prompt. As shown in Algorithm 1 if the similarity score falls below a predetermined threshold, the agent leverages the BLIP-2 model (Li et al., 2023)—a state-of-the-art VLM—to generate a detailed caption for the image. This caption provides an independent description of the image's visual content.

By comparing the BLIP-2-generated caption with both the original prompt and the current optimized prompt, the agent identifies discrepancies or missing elements. For example, if the user's input emphasizes abstract qualities like "lion-like majesty" but the caption describes only a generic birthday scene, the agent infers that critical visual attributes (e.g., strength, regality) may be underrepresented. Based on this analysis, the system automatically refines the optimized prompt, adding or modifying details to better capture the intended semantic content.

This iterative feedback loop—driven by the combination of CLIP-based evaluation and BLIP-2 captioning—ensures that the enriched prompt aligns closely with the user's intent. It provides a robust, quantitative measure to guide the continuous improvement of prompt quality, ultimately enabling consistent and high-quality image generation across different T2I models.

Algorithm 1 Self-Evaluation Agent (SEA) for Prompt Improvement

Require: Generated Image I, Original Prompt P_o, Optimized Prompt P_{opt}, Threshold τ

$s \leftarrow \text{CLIP}(I, P_o)$ ▷ Compute similarity score
if $s \geq \tau$ **then**
  return P_{opt} ▷ No further refinement needed
else
  $C \leftarrow \text{BLIP2}(I)$ ▷ Generate a caption for the image
  $P_{improved} \leftarrow \text{SEA}(P_o, P_{opt}, C)$ ▷ Refine prompt
  return $P_{improved}$
end if

3.5 Feedback and Tuning Agent

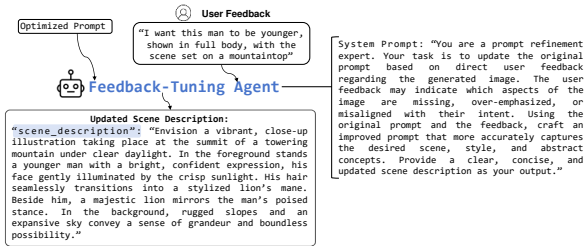


Figure 6: Prompt display of the Feedback-Tuning Agent.

While the Self-Evaluation Agent ensures that the generated image scores highly against the original prompt, it may still miss the user's true intent. To bridge this gap, we introduce a Feedback and Tuning Agent that takes direct user feedback and uses it to iteratively refine the prompt. Because the initial prompt often contains only limited information, this feedback loop is essential for aligning the final image with the user's vision. As shown in Figure 6, the agent applies user comments to continuously improve the optimized prompt until the generated result fully meets the user's requirements.

4 Experiments

4.1 Experiment Settings

Implementation detail We used the API from OpenAI to build our multi-agent system. We use GPT-4o as the basemodel. Each input prompt will first pass the Intent Inference Agent and Scene and Style Agent sequentially. Then an image will generate from the extended prompt. Later the Self-Evaluation agent will keep improve the prompt until its generated image is aligned with the original prompt. If there are feedbacks from user, the Feedback and Tuning Agent will further improve the prompt. We utilize the Stable-Diffusion-

Method	CLIP Score(↑)	PickScore(↑)	Aes Score(↑)
Original	**0.289**	19.43	5.87
Extended	0.232	20.28	6.21
MagicPrompt	0.246	18.69	6.11
Ours w/o SEA	0.257	20.26	6.68
Ours	0.263	**21.31**	**6.96**

Table 1: Performance comparison for different methods. "SEA" stands for Self-Evaluation Agent.

XL(SDXL) (Podell et al., 2023) and Midjourney (Midjourney, 2024) as the T2I model. For each test, all methods to compare will choose the same T2I model.

Methods to compare We have selected a few famous models as our baselines and counterparts:

- Original prompt.
- Simple extend prompt: we ask GPT 4 to add more detail to the original prompt.
- MagicPrompt (Gustavosta): A prompt enrichment framework trained with 80000 images from stable diffusion. They use simple-enriched prompt pair to fine-tune a GPT-2.
- PromptAgent(Wang et al.): State-of-the-Art in prompt optimization using multi-agent system.

Metrics It is not straightforward to compare the quality of different optimized prompts. We use the generated image from T2I model as a metrics for the quality of a prompt. Then we calculate the PickScore (Kirstain et al., 2023), CLIP score (Radford et al., 2021), Aesthetic score (Schuhmann et al., 2022) between original prompts and the generated images. CLIP score evaluate the similarity between the prompts and the generated image. PickScore evaluate the alignment between the prompts and the generated image and also the overall quality. Aesthetic score evaluate the human preference on the aesthetic of the generated images.

4.2 Overall Results

Table 1 presents a comprehensive performance comparison. Here, “Original” denotes the initial short prompt, “Extended” refers to the GPT-4 expanded prompt, and “MagicPrompt” or "PromptAgent" represents the prompt extended using MagicPrompt or PromptAgent from the original simple prompt. Our method achieves the highest PickScore, Aesthetic Score and human expert preference score , demonstrating that it effectively aligns the optimized prompt with the generated image and produces higher-quality images with enhanced aesthetic appeal and user preference. Additionally, our method attains the second highest CLIP score, indicating strong alignment between the generated image and the original user request. Note that the CLIP score is calculated using the original prompt and the generated image, which is why the image generated from the original prompt shows the highest similarity to that prompt.

Method	Preference Score(↑)	Number of Runs(↓)
Original	69.85 %	6.08
Extended	75.32 %	4.22
MagicPrompt	67.28 %	5.33
Ours	**80.12 %**	**2.35**

Table 2: Human Evaluation Comparison.

Ablation Study Table 1 shows that our Self-Evaluation Agent (SEA) consistently improve the evaluation performance compared to without this agent. Thanks to the automatically alignment evaluation, our method can achieve higher output quality without additional user’s feedback.

4.3 Human Evaluation

In addition to our model-based evaluation, we conducted a human evaluation. Twenty volunteers generated 60 real-world prompts that were intentionally vague, using abstract words and metaphors. They then rated the first-round generated images on a scale from 0 to 100 and recorded the number of prompt modifications needed to reach satisfaction. As shown in Table 2, our method consistently achieved the highest preference scores. This indicates that our approach meets real-world prompt auto-completion requirements. Moreover, thanks to the Self-Evaluation Agent’s ability to identify misalignments and the Feedback-Tuning Agent’s efficient prompt optimization, our method required the fewest modifications to achieve satisfaction.

4.4 Visualization Results

We have included visualization results in the Appendix A. Our method consistently shows better output quality and close alignment to users’ intent compared to other methods.

We also include the prompt comparison in the appendix.

5 Industry Impact

We are collaborating with a startup to build a platform that integrates T2I model prompt autocompletion and optimization. As we have mentioned, there are many people who lack Prompt Engineering Experience but are willing to try Text-to-Image models. The goal of this platform is to empower these users in generating impressive figures from just a simple idea. The platform in under development and is supposed to be public very soon.

6 Conclusion

In this paper, we introduce PromptSculptor, a novel multi-agent framework that automates the iterative refinement of T2I prompts. By decomposing the complex prompt refinement process into specialized agents, our approach produces detailed, context-aware prompts that better align with user expectations from simple and vague input. Our experiments demonstrate that PromptSculptor significantly enhances both the quality and aesthetic appeal of generated images from simple inputs.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Tingfeng Cao, Chengyu Wang, Bingyan Liu, Ziheng Wu, Jinhui Zhu, and Jun Huang. 2023. Beautiful-Prompt: Towards automatic prompt engineering for text-to-image synthesis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1–11, Singapore. Association for Computational Linguistics.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. 2023. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM transactions on Graphics (TOG)*, 42(4):1–10.

Junzhe Chen, Xuming Hu, Shuodi Liu, Shiyu Huang, Wei-Wei Tu, Zhaofeng He, and Lijie Wen. 2024a. Llmarena: Assessing capabilities of large language models in dynamic multi-agent environments. *arXiv preprint arXiv:2402.16499*.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2024b. Instructzero: Efficient instruction optimization for black-box large language models. In *International Conference on Machine Learning*, pages 6503–6518. PMLR.

Jaemin Cho, Abhay Zala, and Mohit Bansal. 2023. Visual programming for step-by-step text-to-image generation and evaluation. *Advances in Neural Information Processing Systems*, 36:6048–6069.

Kexin Chu, Zecheng Lin, Dawei Xiang, Zixu Shen, Jianchang Su, Cheng Chu, Yiwei Yang, Wenhui Zhang, Wenfei Wu, and Wei Zhang. 2025a. Selective kv-cache sharing to mitigate timing side-channels in llm inference. *arXiv preprint arXiv:2508.08438*.

Kexin Chu, Zixu Shen, Dawei Xiang, and Wei Zhang. 2025b. Safekv: Safe kv-cache sharing in llm serving. In *Machine Learning for Computer Architecture and Systems*.

Shizhe Diao, Zhichao Huang, Ruijia Xu, Xuechun Li, LIN Yong, Xiao Zhou, and Tong Zhang. Black-box prompt learning for pre-trained language models. *Transactions on Machine Learning Research*.

Dave Epstein, Allan Jabri, Ben Poole, Alexei Efros, and Aleksander Holynski. 2023. Diffusion self-guidance for controllable image generation. *Advances in Neural Information Processing Systems*, 36:16222–16239.

Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Reddy Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis. In *The Eleventh International Conference on Learning Representations*.

Yingchaojie Feng, Xingbo Wang, Kam Kwai Wong, Sijia Wang, Yuhong Lu, Minfeng Zhu, Baicheng Wang, and Wei Chen. 2023. Promptmagician: Interactive prompt engineering for text-to-image creation. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):295–305.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *International Conference on Machine Learning*, pages 13481–13544. PMLR.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Ziyu Guo, Renrui Zhang, Chengzhuo Tong, Zhizheng Zhao, Peng Gao, Hongsheng Li, and Pheng-Ann Heng. 2025. Can we generate images with cot? let's verify and reinforce image generation step by step. *arXiv preprint arXiv:2501.13926*.

Gustavosta. Gustavosta/magicprompt-stable-diffusion.

Wenyang Hu, Yao Shu, Zongmin Yu, Zhaoxuan Wu, Xiaoqiang Lin, Zhongxiang Dai, See-Kiong Ng, and Bryan Kian Hsiang Low. 2024. Localized zeroth-order prompt optimization. *Advances in Neural Information Processing Systems*, 37:86309–86345.

Shyamgopal Karthik, Karsten Roth, Massimiliano Mancini, and Zeynep Akata. 2023. If at first you don't succeed, try, try again: Faithful diffusion-based text-to-image generation by selection. *arXiv preprint arXiv:2305.13308*.

Salar Hosseini Khorasgani, Yuxuan Chen, and Florian Shkurti. 2022. Slic: Self-supervised learning with iterative clustering for human action videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. 2023. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:36652–36663.

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. 2023. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*.

H.Y. Leong and Y. Wu. 2024. Why should next-gen llm multi-agent systems move beyond fixed architectures to dynamic, input-driven graphs? *SSRN Electronic Journal*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *Transactions on Machine Learning Research*.

Xiaoqiang Lin, Zhaoxuan Wu, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, and Bryan Kian Hsiang Low. Use your instinct: Instruction optimization for llms using neural bandits coupled with transformers. In *Forty-first International Conference on Machine Learning*.

Yao Lu, Zhaiyuan Ji, Jiawei Du, Yu Shanqing, Qi Xuan, and Tianyi Zhou. 2025. From llm-anation to llm-orchestrator: Coordinating small models for data labeling. *arXiv preprint arXiv:2506.16393*.

Atefeh Mahdavi Goloujeh, Anne Sullivan, and Brian Magerko. 2024. Is it ai or is it me? understanding users' prompt journey with text-to-image generative ai tools. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Oscar Mañas, Pietro Astolfi, Melissa Hall, Candace Ross, Jack Urbanek, Adina Williams, Aishwarya Agrawal, Adriana Romero-Soriano, and Michal Drozdzal. Improving text-to-image consistency via automatic prompt optimization. *Transactions on Machine Learning Research*.

Midjourney. 2024. Midjourney. https://www.midjourney.com/. Accessed: 2024-07-05.

Wenyi Mo, Tianyu Zhang, Yalong Bai, Bing Su, Ji-Rong Wen, and Qing Yang. 2024. Dynamic prompt optimizing for text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26627–26636.

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. 2021. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2085–2094.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. Grips: Gradient-free, edit-based instruction search for prompting large language models. In *EACL*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294.

Weijia Shi, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. Toward human readable prompt tuning: Kubrick's the shining is a good movie, and a good prompt too? In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Jiao Sun, Deqing Fu, Yushi Hu, Su Wang, Royi Rassin, Da-Cheng Juan, Dana Alon, Charles Herrmann, Sjoerd van Steenkiste, Ranjay Krishna, et al. Dreamsync: Aligning text-to-image generation with image understanding feedback. In *Synthetic Data for Computer Vision Workshop@ CVPR 2024*.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuan-Jing Huang, and Xipeng Qiu. 2022. Bbtv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. 2024. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*.

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.

Qiucheng Wu, Yujian Liu, Handong Zhao, Trung Bui, Zhe Lin, Yang Zhang, and Shiyu Chang. 2023a. Harnessing the spatial-temporal attention of diffusion models for high-fidelity text-to-image synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7766–7776.

Xiaoshi Wu, Keqiang Sun, Feng Zhu, Rui Zhao, and Hongsheng Li. 2023b. Human preference score: Better aligning text-to-image models with human preference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2096–2105.

Xidong Wu, Sumin Jo, Yiming Zeng, Arun Das, Ting-He Zhang, Parth Patel, Yuanjing Wei, Lei Li, Shou-Jiang Gao, Jianqiu Zhang, et al. 2024. regulogpt: Harnessing gpt for end-to-end knowledge graph construction of molecular regulatory pathways. In *2024 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, pages 1–8. IEEE.

Wenyan Xu, Rundong Wang, Chen Li, Yonghong Hu, and Zhonghua Lu. 2025a. Hrft: Mining high-frequency risk factor collections end-to-end via transformer. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 538–547.

Wenyan Xu, Dawei Xiang, Yue Liu, Xiyu Wang, Yanxiang Ma, Liang Zhang, Chang Xu, and Jiaheng Zhang. 2025b. Finmultitime: A four-modal bilingual dataset for financial time-series analysis. *arXiv preprint arXiv:2506.05019*.

Wenyan Xu, Dawei Xiang, Rundong Wang, Yonghong Hu, Liang Zhang, Jiayu Chen, and Zhonghua Lu. 2025c. Learning explainable stock predictions with tweets using mixture of experts. *arXiv preprint arXiv:2507.20535*.

Shuang Zeng, Xinyuan Chang, Mengwei Xie, Xinran Liu, Yifan Bai, Zheng Pan, Mu Xu, and Xing Wei. 2025a. Futuresightdrive: Thinking visually with spatio-temporal cot for autonomous driving. *arXiv preprint arXiv:2505.17685*.

Yiming Zeng, Wanhao Yu, Zexin Li, Tao Ren, Yu Ma, Jinghan Cao, Xiyan Chen, and Tingting Yu. 2025b. Bridging the editing gap in llms: Fineedit for precise and targeted text modifications. *Preprint*, arXiv:2502.13358.

Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. 2023. Text-to-image diffusion models in generative ai: A survey. *arXiv preprint arXiv:2303.07909*.

J. Zhang, J. Gao, W. Ouyang, W. Zhu, and H.Y. Leong. 2025a. Time-llama: Adapting large language models for time series modeling via dynamic low-rank adaptation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*. Association for Computational Linguistics (ACL 2025). Poster.

Tuo Zhang, Jinyue Yuan, and Salman Avestimehr. 2024. Revisiting opro: The limitations of small-scale llms as optimizers. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1727–1735.

Zhenhua Zhang, Jianfeng Wang, Zhengyang Li, Yunpeng Wang, and Jiayun Zheng. 2025b. Anncoder: A mti-agent-based code generation and optimization model. *Symmetry*, 17(7).

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

A Appendix

Figure 7 presents four prompt types used to generate images for six abstract themes. The results for each theme are analyzed as follows:

In Figure 8, our prompt shows a schoolboy imagining himself as a future astronaut exploring space,

Figure 7: Four Prompt Types for Constructing Images Based on Six Abstract Themes

creating a vivid visual metaphor that highlights the theme "Dreams Fuel Growth." In contrast, other prompts are more abstract or simplistic, lacking the kind of concrete and engaging imagery that sparks imagination. Moving to Figure 9, our prompt depicts sunlight breaking through dark clouds, with doves symbolizing hope, freedom, and resilience. This scene conveys a deeper emotional meaning and a sense of uplift, whereas the other prompts fall short of capturing such layered symbolism. In Figure 10, our prompt illustrates children in a rural classroom studying diligently, expressing their desire to change their fate through knowledge. This powerful imagery embodies hope and perseverance in the face of hardship. By comparison, the other prompts rely on more literal depictions of reading and fail to capture the abstract essence behind the pursuit of education. However, in Figure 11, none of the prompts—including ours—clearly emphasize the crucial sense of distance needed to effectively express the idea that "True Love Transcends Distance." This remains an area for improvement. In Figure 12, unlike other prompts that directly show injured people and clocks, our prompt uses the imagery of rebuilding after a fire to symbolically convey the idea that "Time Heals All Wounds." This metaphorical approach offers a more thoughtful and layered interpretation. Finally, in Figure 13, our prompt integrates more human and natural elements than the others, with greater attention to emotional depth and visual detail. As a result, it presents a more genuine and optimistic atmosphere of peace, effectively capturing the intended theme.

In summary, our prompts offer a more effective interpretation and representation of abstract concepts.

Theme / By method	Dreams Fuel Growth
Original Prompts	I want to draw a figure showing “Dreams Fuel Growth”.
Extended Prompts	I want to draw a figure showing “Dreams Fuel Growth” – a child gazing up at a glowing tree sprouting from an open book, its branches filled with stars and symbols of imagination, reaching toward the sky."
Magic Prompts	I want to draw a figure titled “Dreams Fuel Growth”, in the style of +250k concept art and character illustrations by Greg Rutkowski, Craig Mullins, and Daniel Dociu – trending on ArtStation. The scene features a cloudy background and ultra-realistic digital art.
Our Prompts	I want to draw a figure showing “Dreams Fuel Growth”. A young boy with a backpack stands on a quiet rooftop at dusk, gazing up at the sky. On the right side of the scene, his imagined future self floats in space as an astronaut, surrounded by stars and planets. A glowing trail of stardust connects the boy to his dream, symbolizing growth and aspiration. Soft ambient lighting, vibrant colors against a deep twilight sky, calm and inspiring mood, cinematic composition.".

Figure 8: Four Prompt Types for Constructing Images Based on the Theme ’Dreams Fuel Growth’

Theme / By method	Hope in the Darkness
Original Prompts	I want to draw a figure showing “Hope in the Darkness".
Extended Prompts	I want to draw a figure showing "Hope in the Darkness” – a single light source breaking through shadows, highlighting a small symbol of resilience like a blooming flower or an outstretched hand.
Magic Prompts	“I want to draw a color figure showing "Hope in the Darkness" (W1 9 8 2), trending on artstation, award winning painting, cgi, art by greg rutkowski
Our Prompts	I want to draw a figure showing “Hope in the Darkness". A breathtaking scene above dense, stormy black clouds – a radiant golden sun breaks through, casting light that the darkness cannot reach. High above, a few birds glide effortlessly in the sunlight, symbolizing freedom and resilience. The contrast between the shadowy clouds below and the glowing sky above creates a powerful image of hope rising beyond despair.

Figure 9: Four Prompt Types for Constructing Images Based on the Theme ’Hope in the Darkness’

Theme / By method	**Knowledge is Power**
Original Prompts	I want to draw a figure showing “Knowledge is Power”.
Extended Prompts	I want to draw a figure showing “Knowledge is Power” – a stylized silhouette absorbing light from an open book, radiating energy or transformation, in a clean, symbolic style.
Magic Prompts	I want to draw a figure showing “Knowledge is Power”, drawing of a concept art by Mark Brooks and Brad Kunkle and Craig Mullins, trending on artstation,4k
Our Prompts	I want to draw a figure showing “Knowledge is Power”. A group of ragged children huddled in a dilapidated classroom, studying intently by the faint light streaming through cracked windows, icy wind curling around them. Despite the cold and decay, a soft, ambient glow surrounds their books, casting warm light onto their faces – as if knowledge itself is illuminating them. The room is bleak and gray, but the children’s focused expressions and the golden light symbolize hope and strength. Vibrant contrast between the dull surroundings and the inner light. Cinematic composition, slightly low angle, moody and atmospheric.

Figure 10: Four Prompt Types for Constructing Images Based on the Theme ’Knowledge is Power’

Theme / By method	**Love Transcends Distance**
Original Prompts	I want to draw a figure showing "Love Transcends Distance".
Extended Prompts	I want to draw a figure showing "Love Transcends Distance". Zhinu and Niulang gaze at each other from afar, separated by a shimmering Milky Way. Starlight and drifting cosmic dust surround them in silence.
Magic Prompts	I want to draw a figure showing "Love Transcends Distance". Love Transcends Distance", trending on artstation, concept art.
Our Prompts	I want to draw a figure showing 'Love Transcends Distance' – two people reaching out to each other from opposite sides of a vast landscape or starry sky.

Figure 11: Four Prompt Types for Constructing Images Based on the Theme ’Love Transcends Distance’

Theme / By method	Time Heals All Wounds
Original Prompts	I want to draw a figure showing “Time Heals All Wounds”.
Extended Prompts	I want to draw a figure showing “Time Heals All Wounds” – a person sitting quietly as seasons change around them, scars on his body gradually fading into blooming flowers, symbolizing healing and renewal."
Magic Prompts	I want to draw a color figure “Time Heals All Wounds” by Marc Simonetti, trending on artstation, Midjourney.
Our Prompts	I want to draw a figure showing “Time Heals All Wounds”. A split-scene landscape showing 'Time Heals All Wounds' – on the left, charred ruins and ash-covered ground from a fire, barren trees under a gray, overcast sky; on the right, the same place fully rebuilt with tidy villas, blooming gardens, and children playing under warm sunlight. A gentle transition of light and color connects the two halves, symbolizing recovery and hope. Soft ambient lighting, vibrant greenery, emotional yet calm atmosphere, cinematic wide-angle view.

Figure 12: Four Prompt Types for Constructing Images Based on the Theme ’Time Heals All Wounds’

Theme / By method	World Peace
Original Prompts	I want to draw a figure showing the world will have peace in the end.
Extended Prompts	I want to create an image that speaks of hope and unity—a central, radiant figure standing at the heart of a softly glowing globe. This figure, draped in flowing robes of gentle blues and whites, reaches out with open arms toward a horizon where the first light of dawn touches a landscape of lush meadows, calm seas, and ancient trees. Around the figure, symbols of peace—fluttering doves, delicate olive branches, and intertwined hands—float in a serene dance. The overall atmosphere is one of quiet strength and optimism, capturing the timeless promise that no matter the challenges, the world will ultimately embrace peace.
Magic Prompts	I want to draw a figure showing the world will have peace in the concept art, low angle, high detail, warm lighting, volumetric, godrays, vivid, beautiful, trending on artstation, by Jordan grimmer, huge scene, grass, art greg Rutkowski.
Our Prompts	I want to draw a figure showing the world will have peace in the end. A breathtaking scene of global peace: a sprawling meadow ablaze with vibrant flowers, edged by a gently winding river. Diverse people—men, women, and children—play and laugh under warm, golden sunlight, while graceful birds soar in a clear blue sky, symbolizing hope and freedom.

Figure 13: Four Prompt Types for Constructing Images Based on the Theme ’World Peace’

EasyDistill: A Comprehensive Toolkit for Effective Knowledge Distillation of Large Language Models

Chengyu Wang[1]*, **Junbing Yan**[1], **Wenrui Cai**[1,2], **Yuanhao Yue**[1], **Jun Huang**[1]
[1] Alibaba Cloud Computing [2] Shanghai Jiao Tong University
chengyu.wcy@alibaba-inc.com

Abstract

In this paper, we present *EasyDistill*, a comprehensive toolkit designed for effective black-box and white-box knowledge distillation (KD) of large language models (LLMs). Our framework offers versatile functionalities, including data synthesis, supervised fine-tuning, ranking optimization, and reinforcement learning techniques specifically tailored for KD scenarios. The toolkit accommodates KD functionalities for both System 1 (fast, intuitive) and System 2 (slow, analytical) models. With its modular design and user-friendly interface, *EasyDistill* empowers researchers and industry practitioners to seamlessly experiment with and implement state-of-the-art KD strategies for LLMs. In addition, *EasyDistill* provides a series of robust distilled models and KD-based industrial solutions developed by us, along with the corresponding open-sourced datasets, catering to a variety of use cases. Furthermore, we describe the seamless integration of *EasyDistill* into Alibaba Cloud's Platform for AI (PAI). Overall, the *EasyDistill* toolkit makes advanced KD techniques for LLMs more accessible and impactful within the NLP community. The toolkit, together with source codes, all model checkpoints and datasets, is released at: https://github.com/modelscope/easydistill.

1 Introduction

The proliferation of large language models (LLMs) has been transformative for NLP (Zhao et al., 2023; Yadagiri and Pakray, 2025), pushing the boundaries of what machines can understand and generate in human language. However, the extensive size and complexity of these models present significant challenges, including high computational costs and substantial energy consumption. Knowledge distillation (KD) offers a viable solution to this dilemma, where smaller models are trained to replicate the performance of their larger counterparts, enabling efficient use of resources without sacrificing much accuracy (Xu et al., 2024; Yang et al., 2024c). Despite its potential, effective KD of LLMs is not straightforward, often requiring advanced algorithms and domain expertise. In addition, the lack of tools for LLM-based KD can exacerbate these challenges, limiting exploration and adaptation in industrial settings.[1]

In this paper, we introduce *EasyDistill*, a comprehensive toolkit designed to simplify the KD process for LLMs under both black-box and white-box settings, utilizing proprietary and open-source LLMs as teacher models. *EasyDistill* offers a wide array of functionalities, including data synthesis and augmentation, supervised fine-tuning (SFT), ranking optimization, and reinforcement learning (RL), all tailored for KD scenarios. By supporting both System 1 (fast, intuitive) and System 2 (slow, analytical) models (Li et al., 2025), *EasyDistill* facilitates the KD process across various types of LLMs. *EasyDistill* is easy to use and extend; it provides a modular design with a simple command-line interface for invoking these algorithms.

In addition, *EasyDistill* is more than just an open-source toolkit; it integrates several techniques to support KD for industrial practitioners. The contributions are threefold: i) It includes a series of robust distilled models (e.g., *DistilQwen*), along with open-source datasets, to demonstrate the effectiveness of KD. ii) It features several KD-based industrial solutions (i.e., *EasyDistill-Recipes*) that serve as practical guides for diverse application needs. iii) *EasyDistill* is integrated into Alibaba Cloud's Platform for AI (PAI), showcasing its adaptability and potential for large-scale deployment. By bridg-

* Corresponding author.

[1] Note that there are a few open-source toolkits that support KD for LLMs, such as DistillKit (https://github.com/arcee-ai/DistillKit). To the best of our knowledge, there is a lack of support for various types of KD algorithms and practical solutions (as described below) within the open-source community.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 787–795

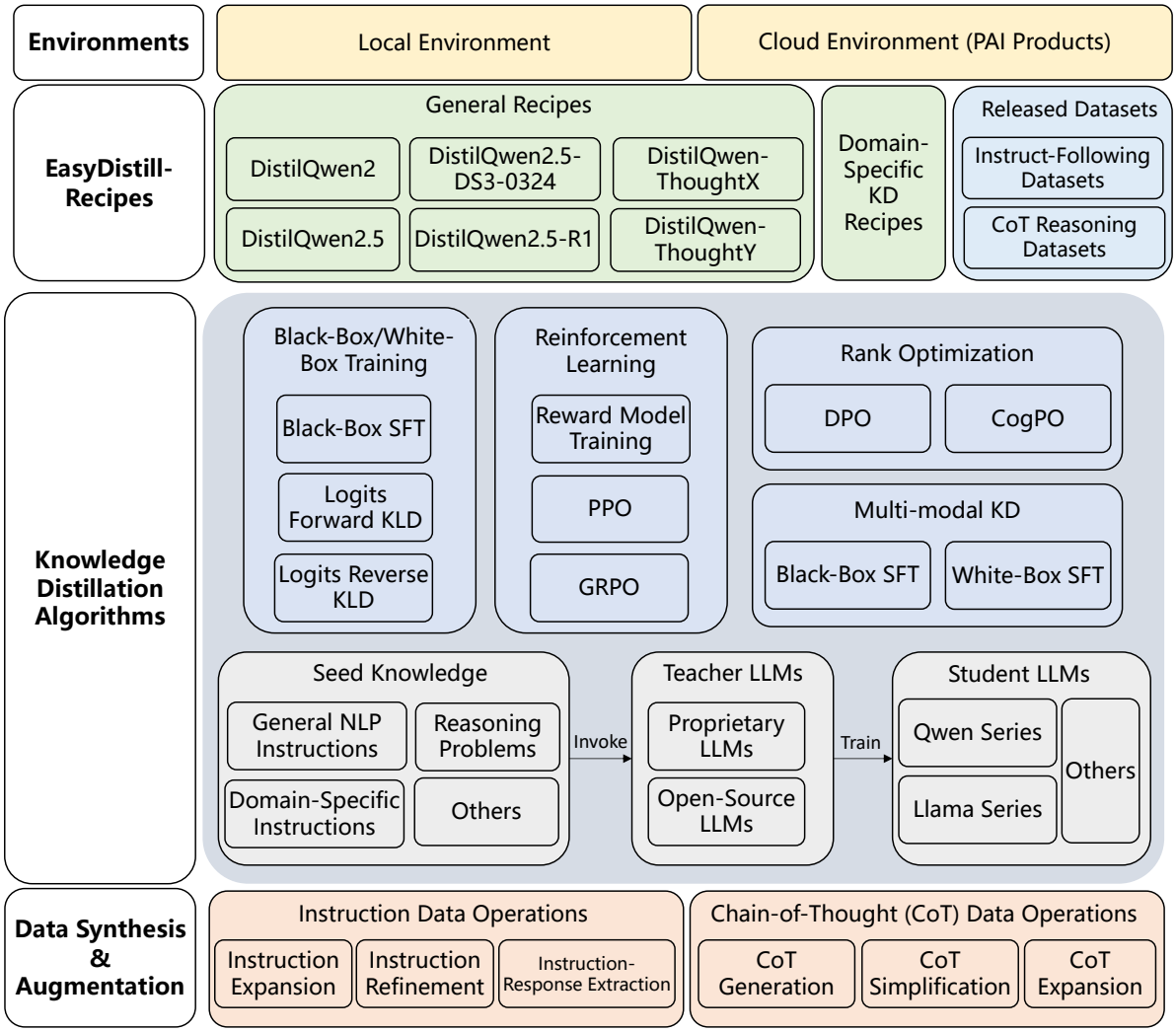


Figure 1: The overall architecture of the *EasyDistill* toolkit.

ing the gap between cutting-edge KD techniques and practical applicability, *EasyDistill* enhances the accessibility within the NLP community.

The remainder of this paper is organized as follows: Section 2 details the architecture and functionalities of *EasyDistill*. Section 3 showcases several practical solutions (i.e., *EasyDistill-Recipes*). Finally, Section 4 concludes the paper and discuss the future work.

2 Architecture and Functionalities

In this section, we formally introduce the *EasyDistill* toolkit. The overall architecture is shown in Figure 1. We begin by presenting the basic KD functionalities, alongside the command-line tool for invoking these functionalities. Following that, we describe the collection of practical solutions (i.e., *EasyDistill-Recipes*). Finally, we briefly describe the integration of the toolkit into PAI products for users to perform KD on the cloud.

2.1 Basic KD Functionalities

2.1.1 Data Synthesis & Augmentation

Synthetic data plays a pivotal role in developing robust LLMs (Liu et al., 2024), especially given the typically limited size of seed datasets for KD. This enhances the KD process, ensuring that the distilled student models not only replicate the output behavior of teacher LLMs but also extend their generalization abilities to previously unseen tasks. In *EasyDistill*, we offer various data synthesis and augmentation operators, utilizing both proprietary and open-source teacher LLMs. These operators are designed to create high-quality seed datasets for KD, enriching them not only in volume but also in diversity across tasks, topics or domains.

The first group of operators supported by *EasyDistill* focuses on the enhancement of instructional data corresponding to a variety of NLP tasks, which form the core inputs (i.e., seed knowledge) to every KD algorithm for LLMs. In this context, we extend our previous work (Yue et al., 2024a) to engineer several functionalities, including instruction expansion, instruction refinement, and the automatic generation of instruction-response pairs from the knowledge expressed in raw texts.

The second group of our operators in *EasyDistill* concentrates on Chain-of-Thoughts (CoTs) (Wei et al., 2022), which are particularly important for distilling the problem-solving capacities of large reasoning models (LRMs) (Li et al., 2025), also known as System 2 models. In addition to the basic operator for producing CoTs grounded in instructions or instruction-response pairs, we further integrate operators for simplifying and extending CoTs to effectively address reasoning problems, as overly long or short CoTs may not be suitable for developing strong LRMs (Yang et al., 2025) We further suggest that combining these two types of operators enhances the KD process for LRMs, as they enable the creation of enriched training sets accompanied by high-quality CoTs.

2.1.2 Training Algorithms for KD Scenarios

The core KD pipeline for LLMs is straightforward. The input is seed knowledge, consisting of instructions for any target tasks, which is leveraged to prompt the selected teacher LLM to generate detailed outputs. In our framework, we support both proprietary and open-source LLMs as teacher models, and open-source LLMs as student models. In the following, we elaborate different types of algorithms tailored to the KD scenarios.

Black-Box/White-Box Training. Since we can only obtain output tokens from proprietary LLMs, the direct KD approach involves supervised fine-tuning (SFT), treating these output tokens as the ground truth for student LLMs.

For open-source teacher LLMs, in addition to SFT, leveraging the models' hidden knowledge as guidance often leads to improved KD performance. In this approach, we obtain the token-level logits from the teacher model and minimize the divergence between the logits distributions of the teacher and student models. The loss functions employed in *EasyDistill* include Kullback–Leibler divergence (KLD) (Gu et al., 2024), reverse KLD (Wu et al., 2025), among others. In the implementation, the forward pass of the teacher model is performed prior to the training of the student model to optimize GPU memory consumption. Furthermore, based on our previous findings (Wang et al., 2025), the sum of the probabilities of the top-10 tokens is almost equal to 1. Thus, *EasyDistill* offers users options to leverage only the top-k token logits from the teacher model and match the corresponding logits from the student model. Subsequently, the computation of loss functions is approximated by considering only k selected logits. This approach not only reduces computation time but also enhances the speed of storing and reading the logits. We do not recommend minimizing the gap between hidden representations, such as attention matrices,

of teacher and student LLMs due to the excessive computational requirements.

Reinforcement Learning (RL). A basic principle of KD is to make the student model mimic the behavior of the teacher models. However, this approach may cause the student model to "overfit" the teacher outputs, rather than exploring more possibilities to enhance its generalization abilities. RL-based approaches, on the other hand, leverage feedback from the teacher to train student models.

The first type of RL-based KD functionalities in *EasyDistill* involves training reward models using feedback from teacher models, similar to the Reinforcement Learning from AI Feedback (RLAIF) framework (Lee et al., 2024). Specifically, we employ teacher models to generate synthetic "chosen" and "rejected" responses as preference data, which are then used to train the reward model based on any targeted LLM backbone with scalar outputs of the predicted reward values.

The second type involves supporting RL optimization to obtain the policy model, i.e., the RL-optimized student model. *EasyDistill* integrates popular RL algorithms for training LLMs, particularly Proximal Policy Optimization (PPO) (Schulman et al., 2017) for System 1 models, and Group Relative Policy Optimization (GRPO) (Shao et al., 2024) for System 2 models. Unlike general RL toolkits, *EasyDistill* emphasizes the entire pipeline of distilling knowledge from teacher models to develop more robust student models, as demonstrated in previous works (Bai et al., 2022; Trung et al., 2024; Yang et al., 2024d).

Preference Rank Optimization. A potential drawback of RL-based algorithms is the instability in training. Preference rank optimization-based approaches directly incorporate preferences into LLMs, making the training process more stable. In *EasyDistill*, we integrate the direct preference optimization (DPO) method (Rafailov et al., 2023) based on the training pipeline from Tunstall et al. (2023) for KD. For System 2 models, which possess strong reasoning capabilities, it is important that distilled smaller models have different capacities and cognitive trajectories than their larger counterparts. To address this, *EasyDistill* integrates our cognitive preference optimization (CogPO) algorithm (Cai et al., 2025b) to enhance the reasoning abilities of smaller models by aligning their cognitive processes with their inherent capacities.

Multi-modal KD. In addition, *EasyDistill* enables the distillation of knowledge not only from text-based sources but also incorporating visual and other data modalities, using multi-modal language models as teacher and student models. This enhances the toolkit's versatility and effectiveness in various application scenarios, allowing users to exploit cross-modal relationships to refine model understanding and predictions.

2.2 Command-Line Interface

To facilitate the KD process using our framework, we provide a user-friendly command-line tool that supports running KD jobs with a simple `JSON` configuration file, specifying the input, output, and all necessary arguments. For example, typical SFT training jobs for black-box KD can be configured as shown in Code 1 and Code 2, utilizing different sources of teacher models. For white-box KD, users can provide additional hyper-parameters and specify the path to store the teacher logits (as shown in Code 3). Once the `JSON` configuration is set, the KD process can be invoked simply by one line of command, shown as follows:

```
easydistill -config=kd.json
```

with the entire pipeline running automatically.

In the provided sample codes, the `inference` section contains essential information, particularly the URL of the model service and its API key, for making inferences with the teacher model. In the online mode, any APIs compatible with the OpenAI API format can be utilized. Therefore, *EasyDistill* is compatible with any teacher models in this case. For offline batch inference, we support vLLM for accelerated model inference (Kwon et al., 2023) when the model can be downloaded to local storage. The `training` configuration includes critical hyper-parameters for the training phase. *EasyDistill* supports all DeepSpeed acceleration techniques by default (Rasley et al., 2020), such as ZeRO and CPU offloading, which can be customized for advanced uses. In the future, other distributed learning frameworks will be supported in *EasyDistill* as well.

```
{
  "job_type": "black_box_kd_api",
  "dataset": {
    "instruction_path": "train.json",
    "labeled_path": "train_labeled.json",
    "template" : "chat_template.jinja",
    "seed": 42
  },
  "inference":{
    "base_url": "ENDPOINT",
    "api_key": "TOKEN",
    "stream": "true",
    "system_prompt" : "You are a helpful
     assistant.",
    "max_new_tokens": 512
```

```
  },
  "models": {
    "student": "student/Qwen/Qwen2.5-0.5B-Instruct/"
  },
  "training": {
    "output_dir": "result/",
    "num_train_epochs": 3,
    "per_device_train_batch_size": 1,
    "gradient_accumulation_steps": 8,
    "save_steps": 1000,
    "logging_steps": 1,
    "learning_rate": 2e-5,
    "weight_decay": 0.05,
    "warmup_ratio": 0.1,
    "lr_scheduler_type": "cosine"
  }
}
```

Code 1: Sample JSON configuration for black-box KD (online inference with a proprietary or open-source teacher model where the teacher model can be accessed by any inference API in the OpenAI format and does not need to be specified in the configuration).

```
{
  "job_type": "black_box_kd_local",
  "dataset": {
    ...
  }
  "inference":{
    "enable_chunked_prefill": true,
    "seed": 777,
    "gpu_memory_utilization": 0.9,
    "temperature": 0.8,
    "trust_remote_code": true,
    "enforce_eager": false,
    "max_model_len": 4096,
    "max_new_tokens": 512
  },
  "models": {
    "teacher": "teacher/Qwen/Qwen2.5-32B-Instruct/",
    "student": "student/Qwen/Qwen2.5-0.5B-Instruct/"
  },
  "training": {
    ...
  }
}
```

Code 2: Sample JSON configuration for black-box KD (offline inference with an open-source teacher model).

```
{
  "job_type": "white_box_kd_local",
  "dataset": {
    "logits_path": "logits.json",
    ...
  }
  "inference":{
    "enable_chunked_prefill": true,
    "seed": 777,
    "gpu_memory_utilization": 0.9,
    "temperature": 0.8,
    "trust_remote_code": true,
    "enforce_eager": false,
    "max_model_len": 4096,
    "max_new_tokens": 512
  },
  "distillation": {
    "kd_ratio": 0.5,
    "max_seq_length": 512,
    "distillation_type": "forward_kld"
  },
  "models": {
    "teacher": "teacher/Qwen/Qwen2.5-7B-Instruct/",
    "student": "student/Qwen/Qwen2.5-0.5B-Instruct/"
  },
  "training": {
    ...
  }
}
```

Code 3: Sample JSON configuration for white-box KD.

2.3 *EasyDistill-Recipes*: Practical Solutions

In this section, we further introduce *EasyDistill-Recipes*, a collection of KD-based solutions that produce lightweight LLMs built on *EasyDistill*. Specifically, all the produced models (i.e., the *DistilQwen* series) are also released to public.

2.3.1 General KD Recipes: *DistilQwen*

In general KD recipes, we offer detailed solutions for producing the *DistilQwen* series using *EasyDistill*. This series includes both System 1 and System 2 models, which are lightweight LLMs built upon the Qwen series. We make these solutions available to enable users to create their own models utilizing the KD techniques in our framework. A brief summary of these models are shown in Table 1. Detailed descriptions and the performance of these models can be found in their respective Hugging Face model cards.

The first collection is *DistilQwen2*, an enhanced version of the Qwen2 models (Yang et al., 2024a), equipped with improved instruction-following capabilities. During the distillation training of *DistilQwen2*, we employ GPT-4 and Qwen-max as teacher models to generate high-quality responses. Specifically, before conducting black-box SFT training, we utilize the method described in (Yue et al., 2024b) to balance the task distributions of input instructions. Following SFT, a rank optimization process is performed using the DPO algorithm (Rafailov et al., 2023) to enhance alignment between the student models and the teacher models. In response to the release of the Qwen2.5 model series (Yang et al., 2024b), *DistilQwen2.5* models are trained using a combination of black-box and white-box KD algorithms. For further details, readers may refer to the report (Wang et al., 2025).

With the release of large System 2 models such as DeepSeek-R1 (DeepSeek-AI, 2025), the concept of "LLM with slow thinking" has become a standard strategy to extend the intelligent boundaries of LLMs. We introduce the *DistilQwen2.5-R1* model series, which utilizes DeepSeek-R1 as the teacher model, based on fine-tuning over a collection of DeepSeek-R1's CoT distillation data. To align the reasoning abilities of smaller distilled models with their intrinsic cognitive capacities, the models are further refined using our CogPO algorithm (Cai et al., 2025b). Additionally, we transfer the fast-thinking, non-reasoning capabilities from

Model Series	Model Type	Parameter Sizes	Teacher LLMs	Student LLMs
DistilQwen2	System 1	1.5B, 7B	GPT-4, Qwen-max	Qwen2
DistilQwen2.5	System 1	0.5B, 1.5B, 3B, 7B	GPT-4, Qwen-max, Qwen2.5-72B-Instruct	Qwen2.5
DistilQwen2.5-DS3-0324	System 1	7B, 14B, 32B	DeepSeek-R1, DeepSeek-V3-0234	Qwen2.5
DistilQwen2.5-R1	System 2	7B, 14B, 32B	DeepSeek-R1	Qwen2.5
DistilQwen-ThoughtX	System 2	7B, 32B	DeepSeek-R1, QwQ-32B	Qwen2.5
DistilQwen-ThoughtY	System 2	4B, 8B, 32B	DeepSeek-R1, DeepSeek-R1-0528, QwQ-32B	Qwen3

Table 1: A summary of the *DistilQwen* model series.

DeepSeek-V3-0324[2] to the *DistilQwen2.5-DS3-0324* models. Here, we first reduce the number of tokens in the training data for *DistilQwen2.5-R1*. Combined with DeepSeek-V3-0324's CoT distillation data, we develop the *DistilQwen2.5-DS3-0324* model series.

The most recent *DistilQwen* series includes *DistilQwen-ThoughtX* and *DistilQwen-ThoughtY*, which exhibit improved reasoning abilities and generate CoTs with more optimal lengths compared to their predecessors. The *DistilQwen-ThoughtX* model series is developed from the innovative OmniThought dataset by utilizing the novel Reasoning Verbosity (RV) and Cognitive Difficulty (CD) scores introduced in OmniThought (Cai et al., 2025a). These scores ensure that models receive rich, high-quality training data reflecting optimal CoT output length and difficulty. *DistilQwen-ThoughtY* is an improved version of *DistilQwen-ThoughtX*, leveraging high-quality CoT distillation data from DeepSeek-R1-0528[3]. Overall, *DistilQwen-ThoughtX* and *DistilQwen-ThoughtY* represent new distilled reasoning models with "adaptive thinking" paradigms, which adaptively solve complicated reasoning problems based on their own knowledge.

2.3.2 Domain-Specific KD Recipes

Within the *EasyDistill-Recipes* module, we further integrate domain-specific recipes for real-world applications. Taking code generation as an example, it generates executable code snippets, assisting developers in writing functional blocks, refining logic, and adapting boilerplate code based on prompts. This capability significantly accelerates software development. In the context of code generation tasks, the primary evaluation metric is *Pass@1*, which measures the model's ability to produce correct, runnable code in a single attempt. A key challenge lies in balancing model capability and inference efficiency: while larger models may achieve higher *Pass@1* scores, they often incur higher computational costs, impacting deployment scalability. Thus, the core optimization goal is to maximize generation accuracy while maintaining a lightweight architecture. To verify the efficacy of *EasyDistill*, we distill two models using prompts and outputs distilled from DeepSeek-R1 based on the OpenCodeReasoning dataset[4]. The detailed performance of the models is presented in Table 2, demonstrating their effectiveness in improving performance in specific tasks.

Model	LiveCodeBench V2	Speedup
Qwen2.5-3B-Instruct	11.35	2.3x
Qwen2.5-3B-Code	**16.62**	2.3x
Qwen2.5-7B-Instruct	30.72	-
Qwen2.5-7B-Code	**35.32**	-

Table 2: Performance comparison of code generation models on LiveCodeBench V2 and inference speedup.

2.3.3 Released Datasets

To assist the community developers in improving instruction-following and CoT reasoning capabilities of LLMs, we have open-sourced two datasets: *DistilQwen_100K* and *DistilQwen_1M*, which are part of the distilled training sets of the *DistilQwen* model series. These datasets cover a range of contents, including mathematics, code, knowledge-based QA, instruction following, and creative generation, with a total dataset size of 100K and 1M entries. For CoT reasoning, we have released *OmniThought*, which is a large-scale dataset featuring 2M CoT processes generated and validated by DeepSeek-R1 and QwQ-32B. Each CoT process is annotated with novel Reasoning Verbosity (RV) and Cognitive Difficulty (CD) scores, which de-

[2] https://huggingface.co/deepseek-ai/DeepSeek-V3-0324

[3] https://huggingface.co/deepseek-ai/DeepSeek-R1-0528

[4] https://huggingface.co/datasets/nvidia/OpenCodeReasoning

Dataset	Size	Task Type	URL
DistilQwen_100K	100K	IF	[URL]
DistilQwen_1M	1M	IF	[URL]
OmniThought	2M	CoT reasoning	[URL]
OmniThought-0528	365K	CoT reasoning	[URL]

Table 3: The summarization of our released datasets. IF refers to "instruction following".

scribe the appropriateness of CoT verbosity and cognitive difficulty level for models to comprehend these reasoning processes. For details, please refer to Cai et al. (2025a). In addition, *OmniThought-0528* is a supplement of *OmniThought* that specifically fauces on the distillation of DeepSeek-R1-0528, which also have rich annotation data regarding the characteristics of CoTs. The information of our released datasets is shown in Table 3.

2.4 Integration to PAI Products

Apart from releasing our toolkit to the open-source community for users to run all kinds of KD algorithms in local environments, we have integrated its key functionalities into Alibaba Cloud's Platform for AI (PAI)[5], a cloud-native machine learning platform. In the platform, all the distilled models produced using *EasyDistill* (e.g., the *DistilQwen* series) are available in the PAI-Model Gallery. This platform supports the entire lifecycle of the LLM usage, including training, evaluation, compression, and deployment of these models. The KD pipelines and practical solutions can be seamlessly executed on deep learning containers on PAI.

Note that although we have provided product integration for *EasyDistill*, the toolkit itself is not platform-dependent; it can be run in any environment satisfying the Python requirements, including other cloud platforms.

3 Conclusion and Future Work

In this paper, we have introduced *EasyDistill*, a comprehensive toolkit focusing on KD for LLMs. It encompasses a suite of advanced algorithms, including data synthesis, SFT, ranking optimization, and RL techniques, all specifically tailored for KD scenarios. Additionally, it includes several practical solutions and is integrated with Alibaba Cloud's Platform for AI (PAI) for large-scale deployment. In the future, we aim to extend the toolkit by supporting a wider range of advanced KD algorithms and by adding more domain-specific solutions to align it even more closely with practical needs.

[5] https://www.alibabacloud.com/en/product/machine-learning

Limitations

There are a few limitations that should be acknowledged. Firstly, the toolkit primarily focuses on established methods for KD, which may limit the exploration of non-standard KD techniques that require further manual integration into the toolkit. Secondly, although *EasyDistill* includes a variety of industrial solutions, the effectiveness can vary based on the specific domains and the quality of available datasets. Finally, while *EasyDistill* enhances accessibility within the NLP community, the toolkit assumes a certain level of technical proficiency for effective utilization. Users lacking deep familiarity with KD processes or LLMs may face a steep learning curve when attempting to leverage the advanced features provided by the toolkit.

Ethic Considerations and Broader Impact

The development of *EasyDistill* makes complex KD processes more accessible to both academic researchers and industry practitioners. It offers a means for companies and educational institutions with limited resources to implement cutting-edge AI models. *EasyDistill*'s integration into Alibaba Cloud's Platform for AI (PAI) and its practical solutions further enhance the toolkit's impact by demonstrating its viability for large-scale deployment. Moreover, the open source of *EasyDistill* encourages community involvement, which could lead to new enhancements in KD techniques.

However, the deployment of models distilled using *EasyDistill* also requires careful consideration of ethical implications, including the potential for bias inherent in LLMs. Ensuring ethical standards are upheld will be crucial to mitigating potential negative social impacts.

Acknowledgments

This work was partially supported by Alibaba Research Intern Program.

References

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. 2022. Constitutional AI: harmlessness from AI feedback. *CoRR*, abs/2212.08073.

Wenrui Cai, Chengyu Wang, Junbing Yan, Jun Huang, and Xiangzhong Fang. 2025a. Reasoning with omnithought: A large cot dataset with verbosity and cognitive difficulty annotations. *CoRR*, abs/2505.10937.

Wenrui Cai, Chengyu Wang, Junbing Yan, Jun Huang, and Xiangzhong Fang. 2025b. Training small reasoning llms with cognitive preference alignment. *CoRR*, abs/2504.09802.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*. OpenReview.net.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626. ACM.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. RLAIF vs. RLHF: scaling reinforcement learning from human feedback with AI feedback. In *Forty-first International Conference on Machine Learning*. OpenReview.net.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang Guo, Le Song, and Cheng-Lin Liu. 2025. From system 1 to system 2: A survey of reasoning large language models. *CoRR*, abs/2502.17419.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. 2024. Best practices and lessons learned on synthetic data for language models. *CoRR*, abs/2404.07503.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3505–3506. ACM.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Luong Quoc Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 7601–7614. Association for Computational Linguistics.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of LM alignment. *CoRR*, abs/2310.16944.

Chengyu Wang, Junbing Yan, Yuanhao Yue, and Jun Huang. 2025. Distilqwen2.5: Industrial practices of training distilled open lightweight language models. *CoRR*, abs/2504.15027.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022*.

Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, Zhe Zhao, and Ngai Wong. 2025. Rethinking kullback-leibler divergence in knowledge distillation for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5737–5755. Association for Computational Linguistics.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *CoRR*, abs/2402.13116.

Annepaka Yadagiri and Partha Pakray. 2025. Large language models: a survey of their development, capabilities, and applications. *Knowl. Inf. Syst.*, 67(3):2967–3022.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *CoRR*, abs/2407.10671.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024b. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Chuanpeng Yang, Wang Lu, Yao Zhu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. 2024c. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *CoRR*, abs/2407.01885.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024d. RLCD: reinforcement learning from contrastive distillation for LM alignment. In *The Twelfth International Conference on Learning Representations*. OpenReview.net.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2025. Towards thinking-optimal scaling of test-time compute for LLM reasoning. *CoRR*, abs/2502.18080.

Yuanhao Yue, Chengyu Wang, Jun Huang, and Peng Wang. 2024a. Building a family of data augmentation models for low-cost LLM fine-tuning on the cloud. *CoRR*, abs/2412.04871.

Yuanhao Yue, Chengyu Wang, Jun Huang, and Peng Wang. 2024b. Distilling instruction-following abilities of large language models with task-aware curriculum planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6030–6054. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

AM4DSP: Argumentation Mining in Structured Decentralized Discussion Platforms for Deliberative Democracy

Sofiane Elguendouze[1] **Lucas Anastasiou**[2] **Erwan Hain**[1]
Elena Cabrio[1] **Anna De Liddo**[2] **Serena Villata**[1]
[1]Université Côte d'Azur, CNRS, Inria, I3S, France
[2]The Open University, Milton Keynes, United Kingdom
{name.lastname}@univ-cotedazur.fr,
{name.lastname}@open.ac.uk

Abstract

Argument(ation) mining (AM) is the automated process of identification and extraction of argumentative structures in natural language. This field has seen rapid advancements, offering powerful tools to analyze and interpret complex and large discourse in diverse domains (political debates, medical reports, etc.). In this paper we introduce an AM-boosted version of BCause, a large-scale deliberation platform. The system enables the extraction and analysis of arguments from online discussions in the context of deliberative democracy, which aims to enhance the understanding and accessibility of structured argumentation in large-scale deliberation processes.

1 Introduction

Deliberative democracy is a form of democracy in which citizens actively participate in public deliberation (Council of Europe, 2023), considering different perspectives and engaging in thoughtful discussion before decisions are made, which provides a mean to improve policy outcomes. This form of democracy aims to enhance policy outcomes by ensuring that public reasoning is central to the decision-making process (Elstub, 2018).

In today's digital age, technologies have expanded the potential reach of public deliberation. However, effectively scaling these deliberative practices while maintaining quality discourse remains a significant challenge (Klein, 2012). Collective deliberation can become a bottleneck in decision-making due to the complexity of processing large volumes of arguments and the diversity of viewpoints. This situation calls for innovative approaches to large-scale civic engagement.

In this context of large-scale deliberations, our work represents a systematic effort to address these challenges through an integrated socio-technical approach that combines Natural Language Processing (NLP), Computational Models of Arguments and Deliberative Democratic practices. More precisely, this paper presents the AM4DSP system, that combines the analytical power of argument mining (AM) methods and leverages BCause[1] strength in organising discussions in argumentative structures.

Effective deliberation platforms, such as BCause, are designed to transcend the limitations of chronologically organized discussions typically found in traditional forums (Anastasiou, 2023). To achieve this goal, this type of platforms incorporate features that organize discussions and structure arguments which significantly improve clarity, facilitate connections between ideas, and promote more informed contributions from participants (Rinner, 2006). Additionally, implementing argument-centric structures that allow users to directly reply to specific points helps maintain focus and enhances sense-making throughout the deliberation process (Irani et al., 2024).

Large-scale deliberations, which are characterized by multiple voices and perspectives, often lead to an overwhelming volume of textual information. This makes them a natural domain for the application of AM, which can automatically extract, analyze, and evaluate arguments from large civic discussions (Lawrence and Reed, 2019; Stede and Schneider, 2019). By leveraging NLP and deep learning methods, AM performs an automated detection of the arguments expressed in participants' statements, their structure and the interactions between them. This can improve decision and policy making by providing policy-makers with a clear view of the underlying positions and justifications presented by participants, feedback on the dynamics of the discussion, key areas of conflict etc. Moreover, AM helps contextualizing arguments by linking them to broader discussions, making it easier to track the evolution of ideas and identify trends in participants' preferences.

[1]https://bcause.app/

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 796–805

2 Related Work

2.1 Deliberation Platforms

Large-scale online deliberation platforms have evolved to address complex societal issues where traditional tools like email, forums and wikis fall short (Klein, 2015). Early systems such as MIT's Deliberatorium (Klein, 2011) and Consider.it (Kriplean et al., 2012) introduced approaches to support large-scale argumentation by reducing redundancy and encouraging clarity in exploring complex problems.

More recent platforms like DebateVis (South et al., 2020) and Kialo (Mei et al., 2024) incorporate visualization techniques to present argumentative structures intuitively, enabling participants to comprehend key discussion points and positioning along opinion spectrums (Shum et al., 2000).

However, these platforms face scaling challenges with large crowds. A balance must be maintained between structured argumentation and natural conversation flow to prevent platforms from becoming overly formal while still organizing argumentative structures effectively (Iandoli et al., 2009).

2.2 Argument(ation) Mining

Argument(ation) Mining (AM) (Cabrio and Villata, 2018; Vecchi et al., 2021) deals with the automated analysis of argumentation structures in written and oral texts (Lawrence and Reed, 2020) from various domains, such as legal cases, persuasive essays, scientific articles, user-generated content, and political debates. The ability of identifying argumentative components (e.g., Premises, Claims) and predicting their relations (e.g., Attack, Support) in these texts opens the door to cutting-edge tasks like fact-checking, counter-argumentation generation and argument quality assessment.

2.2.1 Argumentation Mining Models

Numerous approaches have been proposed for AM, typically relying on hand-crafted syntactic and lexical features (Stab and Gurevych, 2014b), pre-trained language models (Agarwal et al., 2022) or both (Cocarascu et al., 2020). Most recent works leverage supervised approaches with autoencoding transformer architectures like BERT (Devlin et al., 2019), due to their capabilities in understanding the surrounding context and long-term dependencies of arguments. (Habernal et al., 2024) applied argument mining to decisions from the European Court of Human Rights (ECHR). They built models based on pre-trained BERT and RoBERTa (Liu et al., 2019), and achieved comparable results.

2.2.2 Argumentation Mining Data

Since the beginning of the AM field, political debates and user generated content on debate platforms and social media have been considered as promising scenarios to automatically extract and analyse arguments. For the first, we can mention the dataset of political debates from the US presidential elections called USElecDeb60To16 (Haddadan et al., 2019) spanning several decades, where numerous candidates were engaged in multiple discussions. For the user generated content, online debate platforms such as Kialo[2], idebate.org[3] and the subreddit ChangeMyView[4], provide access to publicly-available participatory conversations annotated with argumentative relations (Agarwal et al., 2022; Mezza et al., 2024).

3 BCause Deliberation Platform

BCause is a structured discussion platform designed for large-scale online deliberations that addresses limitations in traditional social media platforms for supporting quality discourse. Unlike conventional social media that often leads to polarization and divisive interactions, BCause incorporates specialized design elements to promote cohesive discussion and reduce group bias (Anastasiou and De Liddo, 2023).

BCause structures deliberation through a layered argumentative approach following the IBIS (Issue-Based Information System) paradigm (Kunz and Rittel, 1970). At its core, the platform organizes discussions around clearly defined debate topics (the issue), positions (opinions or possible solutions to the topic), and arguments (statements that either support "pro" or oppose "con" the parent position). This structure aims to improve the signal-to-noise ratio and provides logical organization to discussions, see Figure 1.

The platform visualizes argumentative structures through two primary methods: (1) A time-ordered timeline providing chronological context. (2) An argument tree displaying the logical relationship between positions and supporting/opposing arguments (evidences). The IBIS model implemented in BCause organizes content hierarchically around three primary elements: Debate topics (issues to

[2]https://www.kialo.com/
[3]https://idebate.net/
[4]https://www.reddit.com/r/changemyview/

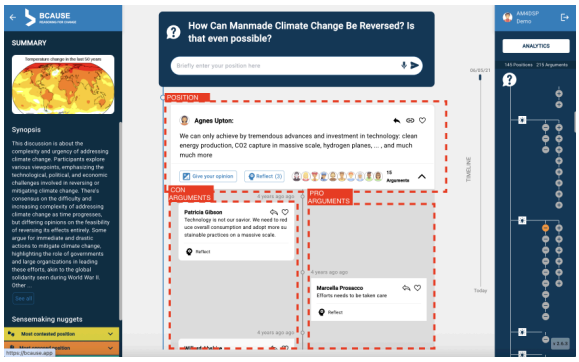


Figure 1: BCause discussion interface with IBIS components annotated

be discussed), Positions (opinions or possible solutions to the topic) and Arguments/Evidences (statements that either support "pro" or oppose "con" a position). This structure intentionally simplifies the argumentative process to make it accessible to non-expert users while still capturing the essential elements of reasoned deliberation. It does however introduce some interesting interplay with classic argumentation models that we explore in the rest of this paper. When the deliberation process is hybrid (involving both face-to-face and online discussions) BCause can provide the online space for continuing a live event, by offering the opportunity for discussion moderators and admins to upload transcript (natural dialogic text among the participants) that occurred during the live event.

4 AM4DSP System Overview

AM4DSP is a boosted version of the BCause deliberation platform, designed to automatically process argumentation structures and relations. These additional functionalities enable the system to perform a deeper analysis of deliberative discussions, extending beyond the standard structure of BCause by integrating AM models at different stages. Figure 2 illustrates the complete architecture of our system.

4.1 System's Main Facilities

The argumentation models integrated into BCause open to a novel range of functionalities, going beyond the traditional setup of discussions on BCause. The main key features are:

- **Discussion-level (statement-level) argumentation,** to automatically analyze user contributions and classify each statement as a position, a supporting or attacking evidence, aligning with BCause's native structure. This would allow to automatically classify and integrate in BCause the arguments put forward by the participants without relying on manual labels.
- **Component-level argumentation,** to break down each statement into its argumentative components (claims and premises) and link them both within and across statements. It offers deeper structural insights and finer-grained analysis for interpreting complex statements where users often blend their main point with supporting evidence.
- **Cross-statement analysis,** to examine the connections between arguments from different participants. It allows identifying broader argumentative structures that span across the full discussion, offering a more comprehensive view of the dialogue and its underlying arguments, that cannot be currently captured by BCause.
- **Transcript analysis,** to analyze longer and more natural dialogues from live events, such as those generated from audio recordings of deliberative events.

While prior user evaluations have been conducted on the core BCause platform, assessing its discussion structure, aesthetics, impact on sensemaking and quality of discussion, these tests did not involve the AM functionalities showcased here. The complete AM4DSP system, including its analytical interface, is scheduled for user testing in real-world use cases as part of the upcoming ORBIS project[5] pilots, which will provide crucial feedback on its usability and deliberation effectiveness.

5 Experimental Setting

In this Section, we present the annotated datasets collected to train the AM models, the models used in the experiments as well as the obtained results.

5.1 Datasets

As discussed earlier in Section 3, the BCause platform structures discourse at the statement level, where each statement (i.e., an argumentative sentence articulated by a participant) is either a position (i.e., a stance taken on a particular argumentative topic) or an evidence that either supports or attacks a given position. We use a sample of structured data from BCause to train the AM models

[5] https://orbis-project.eu/

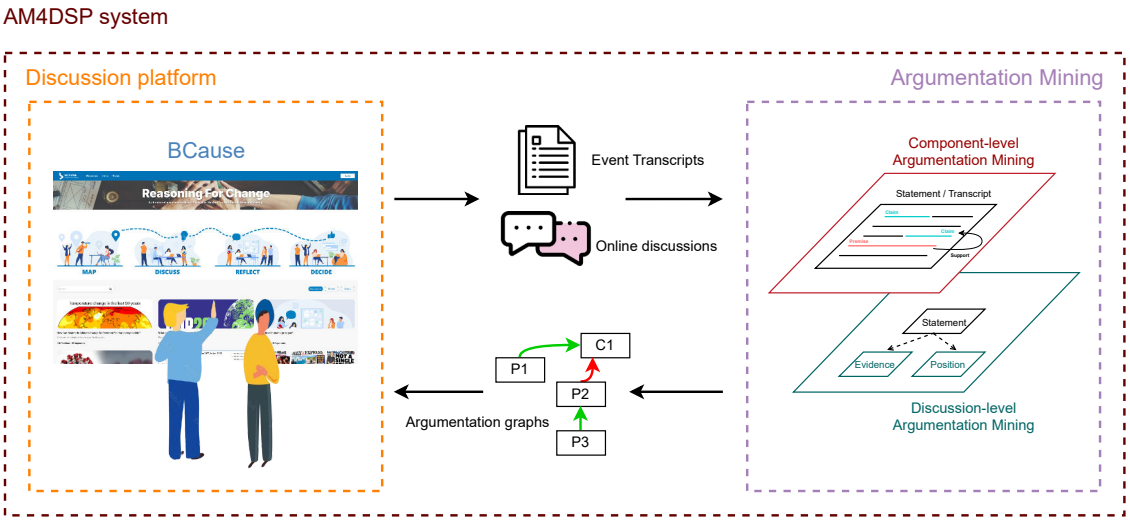


Figure 2: AM4DSP performs argumentation analysis at both the discussion-level and component-level for structured discussions, and provides component-level analysis for transcripts.

targeting what we name *discussion-level argumentation* (Sec. 5.2). In addition to the data from BCause, we have also used Touche23-ValueEval (Mirzakhmedova et al., 2023), a dataset derived from discussions publicly available on internet, mapping the relations *against* and *in support of* with *con* and *pro*, respectively.

Given that we do not have annotated data from BCause to test the *component level* task, to build our classifiers for argumentative components detection and relations detection we used standard datasets as the annotated transcripts of the televised political debates in the US presidential campaigns (1960-2016) (Haddadan et al., 2019), Persuasive Essays (Stab and Gurevych, 2014a) and the dataset presented in (Habernal and Gurevych, 2017).

Tables 1, 2 and 3 provide statistics on the datasets used in our experimental settings (before the train/dev/test splitting).

Dataset	Statement classification		Relation classification	
	Position	Evidence	Attack	Support
BCause	3.9k	5k	2.9	2.2k
Touche23-ValueEval	0.5k	8.7k	3.9k	4.7k

Table 1: Data for *discussion-level argumentation*

Dataset	Argument Components		Argument Relations		
	Claim	Premise	Attack	Support	NoRel
USElecDeb60To16	29k	26k	2.8k	19.8k	23k
Persuasive Essays	2257	3832			
Habernal Gurevych 2017	195	538	-	-	-

Table 2: Data for the *component-level argumentation*

Dataset	O	B-Premise	I-Premise	B-Claim	I-Claim
USElecDeb60To16	566492	26055	350079	29624	338941
Persuasive Essays	35946	2257	29828	3832	59652
Habernal Gurevych 2017	61414	195	3491	538	20566

Table 3: Detailed statistics on datasets used for building *component-level argumentation* models following the BIO-tagging scheme

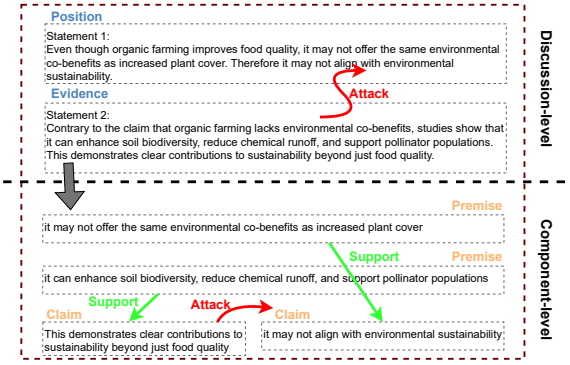


Figure 3: Example showcasing both *discussion-level* and *component-level* argumentation tasks.

5.2 AM Models

Since discussions in BCause adhere to a statement-centric structure, this required an adaptation of the standard AM pipeline to align with the structural requirements of the platform. Two AM steps, namely *discussion-level* and *component-level*, are therefore performed, as exemplified in Figure 3. The first step is performed at the statement level, and its goal is to first classify statements into positions and evidences, and then to identify the relations between

them (support or attack). Importantly, the goal is not to link the evidences to their corresponding parent positions as this is already encoded within BCause's structured data, but rather to uncover additional relationships that can emerge between statements on a broader, discussion-wide scale.

On top of this, the second step concerns a granular analysis performed at the component level following the standard AM pipeline, where argumentative components (premises and claims) within each statement are identified and their relationships within and across statements are determined. As for event transcripts, which are typically longer and more unstructured than the discussions on BCause, only the component argumentation task is applied.

We trained a set of AM models[6] on the datasets described in Section 5.1. We tested BERT-based architectures (RoBERTa and DeBERTa (He et al., 2021)), as well as autoregressive large language models, in particular GPT-2 XL version (1.5B) (Radford et al., 2019) and OPT-1.3B (Zhang et al., 2022). The rationale behind selecting these models despite the availability of more recent and powerful paid alternatives (e.g. GPT-4o) is grounded in the constraints of the ORBIS project. We aim to rely exclusively on open-source and freely available models to ensure long-term sustainability and avoid ongoing operational costs after the project's conclusion.

5.3 Experimental Settings and Results

5.3.1 Discussion-level Argumentation

Statement Classification. We trained our models on a combination of BCause and Touche datasets. The target classes are: Position and Evidence. The training was conducted using a learning rate of 2e-5, a batch size of 32, and a maximum sequence length of 128 tokens, across 5 epochs.

Statement Relation Classification. The target classes are Support and Attack. We use the same dataset and configuration described above, and increase the maximum sequence length to 256.

Table 4 shows the results (f1-macro scores) of our models finetuned on different datasets and evaluated on a test partition composed of 500 new statements obtained from the latest data crawls from BCause. Deberta-v3 model trained on merged data from BCause and Touche achieves the best score for both statement and relation classification.

[6] https://github.com/orbis-marianne/orbis-am-models

Train dataset	Statement classification	Relation classification
Bcause+Touche (deberta-v3)	**0.89**	**0.90**
Bcause+Touche (roberta)	0.83	0.34
Bcause (deberta-v3)	0.71	0.55
Touche (deberta-v3)	0.53	0.65

Table 4: Results for *discussion-level* argumentation

5.3.2 Component-level Argumentation

Component Detection. For BERT-based models, the task is framed as a sequence tagging problem using the standard BIO-tagging scheme. We fine-tune the DeBERTa-v3 model for token classification over 10 epochs with a learning rate of 1e-4 and a maximum sequence length of 64 tokens. For OPT and GPT models, we redesign the task as text generation problem to align with their decoder-only architecture. We convert BIO labels into tagged sequences (e.g., <premise>...</premise>, <claim>...</claim>) and prepare prompts consisting of an instruction describing the task followed by the input sequence in plain text. Figure 4 shows an example of the prompt-response pair used to fine-tune the LLM models. The expected output is a replication of the input plain text with appropriate argument tags inserted.

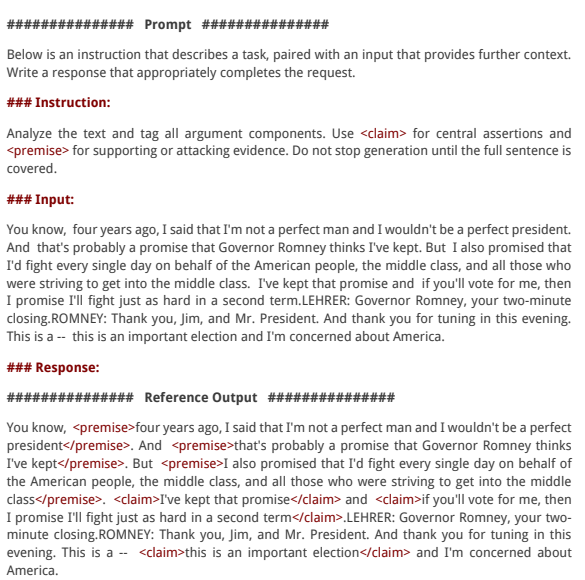


Figure 4: Full prompt and generation example with OPT

To reduce creativity during generation, we apply a low temperature (0.01) and a narrow nucleus sampling threshold (top-p = 0.1), constraining the model to deterministic behavior. Input sequences are chunked into sub-sequences of up to 1024 tokens to fit the models' context limits. Both models are fine-tuned over 10 epochs, with the best checkpoint (based on macro F1 on the validation set)

used for testing. The models are designed to identify three target classes: Premises, Claims, and Non-argumentative tokens. All datasets follow an 80/10/10 split. Table 5 shows the F1 macro results for argument component detection with various models trained on various datasets and evaluated with multiple test configurations (cross testing).

Model	ED	PE	HG	Merged
USElecDeb60To16 (deberta-v3)	0.47	0.47	0.33	0.47
USElecDeb60To16 (roberta)	0.47	0.45	0.30	0.47
Persuasive Essays (deberta-v3)	0.27	0.71	0.24	0.32
Persuasive Essays (roberta)	0.28	0.69	0.41	0.33
Habernal Gurevych 2017 (deberta-v3)	0.18	0.15	0.32	0.18
Habernal Gurevych 2017 (roberta)	0.21	0.22	0.36	0.22
Merged (deberta-v3)	**0.47**	**0.90**	**0.71**	**0.49**
Merged (roberta)	0.46	0.83	0.58	0.48
Merged (fine-tuned OPT-1.3B)	-	-	-	**0.77**
Merged (fine-tuned GPT-2-1.5B)	-	-	-	**0.77**

Table 5: Cross evaluation scores for argument component detection (ED=USElecDeb60To16; PE=Persuasive Essays; HG=Habernal Gurevych 2017)

Table 6 provides a detailed view on the component detection results following various configurations (using different datasets and models). The results are provided in terms of the F1 macro scores obtained on the test partitions of the datasets used for model training.

Component Relation Classification. We fine-tuned the DeBERTa-v3 base model as a sequence classifier, to classify relations into three distinct classes: Support, Attack, and NoRelation. We used a learning rate of 1e-4, with input sequences capped at 64 tokens. The training process spans 10 epochs. Results on USElecDeb60To16 dataset (the only one among the ones we selected, that is annotated with relations) are shown in Table 7.

As expected, algorithms perform best when trained and tested on the same data, and generalize poorly across datasets due to differences in writing style, topic, and argumentation structure. BERT-based models trained on the merged dataset outperform all single-dataset models across all test sets with substantial improvements, especially on PE and HG. This suggests that combining datasets introduces greater variety and richer argumentation patterns, which helps the model generalize better across domains. DeBERTa-v3 consistently outperforms RoBERTa across nearly all configurations, reflecting its stronger representation capacity, especially for this token classification task. The fine-tuned OPT and GPT models achieve the highest F1 score on the merged test set (0.77), showing that generative LLMs can be effectively repurposed for token classification as they may better capture long-range dependencies and context when properly fine-tuned with task-specific formatting. All these experiments allowed us to select the best models to be included in AM4DSP.

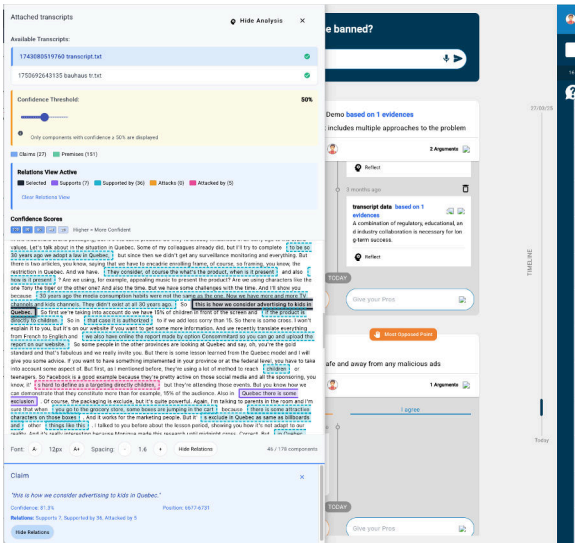


Figure 5: Argument mining applied on attached transcript as rendered in BCause. The selected argumentative component is highlighted in black and the related components in distinct colours: purple for supporting, cyan for supported by, red for attacked by components.

6 AM4DSP Integration in BCause

The AM system is served through a REST API[7] facilitating integration with online deliberation platforms, namely BCause. The code is a Django application, using Django-Rest-Framework to build the REST API. The deployment of the API is heavily dependent on docker and docker-compose. The Django application runs with a Gunicorn server, behind a Nginx SSL proxy that is configured within the docker-compose file.

The system follows a multi-step workflow: (1) Pre-processing, where raw data (statements and transcripts) is cleaned and formatted for analysis, and (2) Argument Mining, where pre-processed data is passed to the AM service for a multi-stage analysis (discussion-level, then component-level). Once identified, argument components and their relationships are returned to BCause in the form of argumentation graphs allowing for interactive visualisation (Figures 5, 6).

[7]API: https://orbis.i3s.unice.fr/api/docs/, source code: https://github.com/orbis-marianne/orbis-argument-mining-tool

Model	B-C	I-C	B-P	I-P	O	Acc	F1-Macro	F1-Avg	Support
USElecDeb60To16 (deverta-v3)	0.39	0.38	0.47	0.44	0.66	0.57	0.47	0.6	194452
USElecDeb60To16 (roberta)	0.39	0.39	0.46	0.46	0.65	0.57	0.47	0.59	194941
Persuasive Essay (deberta-v3)	0.6	0.74	0.58	0.79	0.82	0.75	0.71	0.75	15072
Persuasive Essay (roberta)	0.59	0.68	0.59	0.78	0.19	0.74	0.69	0.74	15241
Habernal Gurevych 2017 (deberta-v3)	0	0	0.4	0.34	0.86	0.76	0.32	0.73	1945
Habernal Gurevych 2017 (roberta)	0	0	0.56	0.41	0.82	0.71	0.36	0.72	1950
Merged (deberta-v3)	0.41	0.43	0.46	0.5	0.63	0.56	0.49	0.58	209865
Merged (roberta)	0.4	0.4	0.46	0.49	0.66	0.57	0.48	0.60	210518
Merged (OPT-1.3B)	0.75	0.73	0.71	0.73	0.92	0.80	0.77	0.80	85288
Merged (GPT-2)	0.76	0.73	0.72	0.72	0.92	0.80	0.77	0.79	89455

Table 6: Detailed f1-scores for argument component detection with various models

Dataset	Relation classification
USElecDeb60To16 (deberta-v3)	**0.69**
USElecDeb60To16 (roberta)	0.61

Table 7: Results obtained for relation classification on the *component-level* argumentation.

Discussion-level argumentation in turn generates an aggregated key argument graph representing the discussion's logical structure. This graph is accessible through BCause's analytics view and unveils hidden argumentative relations, reduces misinterpretations of argument polarity (supporting vs. opposing), and identifies duplicated arguments. Users can explore a synthesised representation of complex issues based on the main arguments while maintaining the ability to seamlessly navigate back to the source data for detailed context.

7 AM workflow in AM4DSP

To demonstrate how the system[8] supports argument mining, we designed a typical workflow that begins with creating a discussion space. In order to populate the discussion with some initial positions and pro/con arguments, users can upload a transcript (e.g., from a public debate or video) through the Import Transcript tool, then automatically extract argument components and visualize them in a preliminary argument tree. At this stage, further user contributions can be added, enabling a mix of automatically extracted and user-supplied arguments.

Once the data are available, the platform provides interactive tools for finer-grained analysis. The transcript viewer highlights argumentative components (premises and claims) directly in the text, with a confidence slider to control the granularity of detection. Selecting a highlighted claim reveals its related premises (support/attack). For a broader overview, the platform generates an Argument Network Analysis, where claims and premises appear as nodes in a force-directed graph, with edges indicating support or attack relations. Users can toggle views to inspect arguments originating from posts, transcripts, or both in combination.

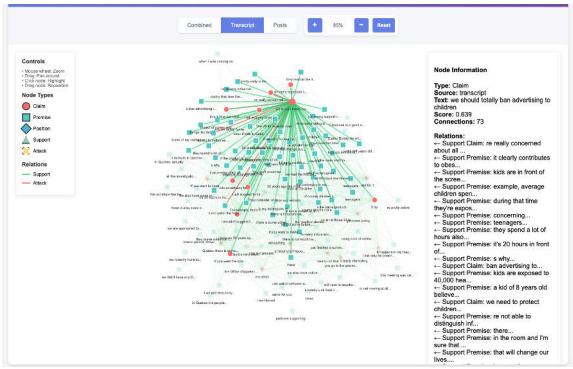


Figure 6: The network visualization of argumentative discourse in BCause analytics, showing interconnected nodes representing claims (red circles), premises (blue squares), and their relationships via support (green lines) and attack (red lines).

8 Conclusion

In this paper, we introduced AM4DSP, an enhanced version of the BCause deliberation platform, augmented with AM capabilities. This publicly available open-access system extends the basic discussion schema by incorporating a dual-level argumentative analysis: a higher-level discussion (or statement) analysis and a fine-grained component-level argumentation. AM4DSP also provides broader insights by identifying cross-statement relations, enabling the mapping of argumentative connections across multiple statements and contributions from different participants. The argumentation mining framework can be integrated into other platforms beyond BCause via a REST API.

[8] A screencast video demonstrating the system can be found at the following link https://youtu.be/c4uMSdTnm5k; the live demo can be accessed at `https://bcause.app/discussions/-OMMdBg09OoAPTFhifkR`

Limitations

This work demonstrates the potential of providing an advanced argument mining service directly integrated into the UX of existing deliberation systems. However, the value and effectiveness of the argument extraction from the perspective of human evaluation and the impact this might have on the quality of human deliberation have yet to be evaluated. This also depends on factors, such as the UX of the chosen deliberation system and, as we have pointed out, the data structure used. Future research should be aimed at conducting user studies in which these aspects can be considered more thoroughly. A key limitation of our current system is the reliance on large-scale, high-quality annotated datasets for building accurate Argument Mining models, particularly at the component level, where data scarcity is a significant challenge. Unlike other NLP tasks, Argument Mining requires domain-specific expertise for annotation, making the process costly and labor-intensive. Recent advancements in large language models (LLMs) offer potential solutions through techniques like in-context learning and few-shot learning, where models such as Mistral and Llama can generalize tasks with minimal data. However, despite their potential, these models are often impractical for everywhere deployment due to their size, computational demands, and concerns over data privacy (when relying on external LLM APIs). Addressing these challenges will require continued research into lightweight, domain-adapted LLMs that balance performance, efficiency, and usability in practical AM scenarios.

Ethical considerations

The integration of Argument Mining (AM) into BCause opens valuable opportunities for structured analysis of deliberative discussions, but it also presents some ethical risks. Automated analysis could be misused to manipulate or bias discussions, for example by selectively highlighting arguments to favor certain positions, misclassifying user contributions to downplay dissent, or profiling participants based on their expressed opinions. These risks are particularly concerning in politically sensitive or high-stakes decision-making contexts, where the neutrality and transparency of the system are crucial.

To mitigate such risks, several safeguards are/should be implemented. First, the models and datasets used in AM are openly documented to ensure transparency about their training, limitations, and potential biases. Second, user privacy is preserved by anonymizing data, securely storing it, and never transfer it outside trusted environments. The final system should remain a support tool rather than a decision-maker: human oversight is essential to validate outputs and contextualize results. Finally, mechanisms for accountability should be put in place, such as audit trails and explainable AI components, to allow stakeholders to understand and contest the system's reasoning when necessary.

Acknowledgments

This work has been supported by the European Project ORBIS "Augmenting participation, co-creation, trust and trasparency in Deliberative Democracy at all scales" under the Horizon Europe Programme (Grant Agreement No. 101094765). This work has also been partially supported by the French government, through the 3IA Cote d'Azur investments in the project managed by the National Research Agency (ANR) with the reference number ANR-23-IACL-0001, and by UKRI under the UK Government's Horizon Europe Guarantee scheme (Reference Number: 10048874).

References

Vibhor Agarwal, Sagar Joglekar, Anthony P. Young, and Nishanth Sastry. 2022. Graphnli: A graph-based natural language inference model for polarity prediction in online debates. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2729–2737, New York, NY, USA. Association for Computing Machinery.

Lucas Anastasiou. 2023. Computational argumentation approaches to improve sensemaking and evidence-based reasoning in online deliberation systems. Unpublished.

Lucas Anastasiou and Anna De Liddo. 2023. BCause: Reducing group bias and promoting cohesive discussion in online deliberation processes through a simple and engaging online deliberation tool. In *Proceedings of the First Workshop on Social Influence in Conversations (SICon 2023)*, pages 39–49, Toronto, Canada. Association for Computational Linguistics.

Elena Cabrio and Serena Villata. 2018. Five years of argument mining: A data-driven analysis. In *IJCAI*, volume 18, pages 5427–5433.

Oana Cocarascu, Elena Cabrio, Serena Villata, and Francesca Toni. 2020. Dataset independent baselines for relation prediction in argument mining. In *Computational Models of Argument*, pages 45–52. IOS Press.

Council of Europe. 2023. Report on deliberative democracy. Accessed: 2023-02-25.

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.

Stephen Elstub. 2018. Deliberative and participatory democracy. *The Oxford handbook of deliberative democracy*, pages 187–202.

Ivan Habernal, Daniel Faber, Nicola Recchia, Sebastian Bretthauer, Iryna Gurevych, Indra Spiecker genannt Döhmann, and Christoph Burchard. 2024. Mining legal arguments in court decisions. *Artificial Intelligence and Law*, 32(3):1–38.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Shohreh Haddadan, Elena Cabrio, and Serena Villata. 2019. Disputool – a tool for the argumentative analysis of political debates. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6524–6526. International Joint Conferences on Artificial Intelligence Organization.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *Preprint*, arXiv:2006.03654.

Luca Iandoli, Mark Klein, and Giuseppe Zollo. 2009. Enabling on-line deliberation and collective decision-making through large-scale argumentation: A new approach to the design of an internet-based mass collaboration platform. *Int. J. Decis. Support Syst. Technol.*, 1:69–92.

Arman Irani, Michalis Faloutsos, and Kevin Esterling. 2024. Argusense: Argument-centric analysis of online discourse. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 18, pages 663–675.

Mark Klein. 2011. The mit deliberatorium: Enabling large-scale deliberation about complex systemic problems. *2011 International Conference on Collaboration Technologies and Systems (CTS)*, pages 161–161.

Mark Klein. 2012. Enabling large-scale deliberation using attention-mediation metrics. *Computer Supported Cooperative Work (CSCW)*, 21:449–473.

Mark Klein. 2015. A critical review of crowd-scale online deliberation technologies.

Travis Kriplean, Jonathan Morgan, Deen Freelon, Alan Borning, and Lance Bennett. 2012. Supporting reflective public thought with considerit. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, page 265–274, New York, NY, USA. Association for Computing Machinery.

W. Kunz and H. W. Rittel. 1970. *Issues as elements of information systems (Vol. 131, p. 14)*. Institute of Urban and Regional Development, University of California, Berkeley, CA.

John Lawrence and Chris Reed. 2019. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

John Lawrence and Chris Reed. 2020. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

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*.

Bing Mei, Peipei Xiong, and Hongyu Xu. 2024. Kialo edu. *RELC Journal*, page 00336882231226156.

Stefano Mezza, Wayne Wobcke, and Alan Blair. 2024. Exploiting dialogue acts and context to identify argumentative relations in online debates. In *Proceedings of the 11th Workshop on Argument Mining (ArgMining 2024)*, pages 36–45, Bangkok, Thailand. Association for Computational Linguistics.

Nailia Mirzakhmedova, Johannes Kiesel, Milad Alshomary, Maximilian Heinrich, Nicolas Handke, Xiaoni Cai, Barriere Valentin, Doratossadat Dastgheib, Omid Ghahroodi, Mohammad Ali Sadraei, Ehsaneddin Asgari, Lea Kawaletz, Henning Wachsmuth, and Benno Stein. 2023. The touché23-valueeval dataset for identifying human values behind arguments. *Preprint*, arXiv:2301.13771.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*, 1(8).

Claus Rinner. 2006. Argumentation mapping in collaborative spatial decision making. In *Collaborative geographic information systems*, pages 85–102. IGI Global.

Simon Buckingham Shum, Albert M. Selvin, and White Plains. 2000. Structuring discourse for collective interpretation.

Laura South, Michail Schwab, Nick Beauchamp, Lu Wang, John P. Wihbey, and Michelle A. Borkin. 2020. Debatevis: Visualizing political debates for non-expert users. *2020 IEEE Visualization Conference (VIS)*, pages 241–245.

Christian Stab and Iryna Gurevych. 2014a. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014b. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 46–56.

Manfred Stede and Jodi Schneider. 2019. *Argumentation Mining*. Springer International Publishing.

Eva Maria Vecchi, Neele Falk, Iman Jundi, and Gabriella Lapesa. 2021. Towards argument mining for social good: A survey. 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 1338–1352, Online. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

TRACE: Training and Inference-Time Interpretability Analysis for Language Models

Nura Aljaafari[1†], **Danilo S. Carvalho**[3], **André Freitas**[1,2,3]
[1] Department of Computer Science, University of Manchester, United Kingdom
[2] Idiap Research Institute, Switzerland
[3] National Biomarker Centre, CRUK-MI, Univ. of Manchester, United Kingdom
{firstname.lastname}@[postgrad.][†]manchester.ac.uk

github.com/neuro-symbolic-ai/trace_package Short Video

Abstract

Understanding when and how linguistic knowledge emerges during language model training remains a central challenge for interpretability. Most existing tools are post hoc, rely on scalar metrics, or require nontrivial integration effort, making comprehensive interpretability analysis difficult to deploy and maintain. We introduce **TRACE**, a modular toolkit for training and inference-time interpretability analysis of transformer models. It enables lightweight, in-training analysis of linguistic and representational signals, including features probing, intrinsic dimensionality, Hessian curvature, and output diagnostics. It integrates with **ABSynth**, a controllable synthetic corpus generator that provides structured annotations for precise evaluation of linguistic feature acquisition. Experiments with autoregressive transformers demonstrate that TRACE reveals developmental phenomena such as early syntactic emergence, delayed semantic acquisition, and representational compression, signals overlooked by traditional scalar metrics such as loss or accuracy. With minimal integration effort, the tool enables layer-wise diagnostics, convergence-based early stopping, and detection of structural errors, making transformer analysis interpretable, actionable, and reproducible.

1 Introduction

Interpreting the behaviour of transformer-based language models (LMs) has been gaining interest and importance (Nostalgebraist, 2020; Wang et al., 2023; Hanna et al., 2023; Belrose et al., 2023; Meng et al., 2022), especially with the widespread use of them across various domains. This interpretation is fundamental for verifying their correctness, analysing reasoning processes, and improving robustness (Bereska and Gavves, 2024).

Although various approaches to interpretability have been proposed, most are post hoc: they examine fully trained models using input-output correlations (Kokhlikyan et al., 2020; Lundberg and Lee, 2017; Zhao and Shan, 2024), structured tasks such as mathematics or algorithmic reasoning (Hanna et al., 2023; Nanda et al., 2023), or mechanistic analysis (Belrose et al., 2023; Meng et al., 2022). Tools that operate in training time, such as TensorBoard or Weights & Biases, focus narrowly on scalar metrics like loss or gradient norms. They reveal *whether* a model is learning, but not *what* is being learned or *when* key capabilities emerge. Lightweight, modular tools for tracking representational development remain scarce.

This creates a critical gap: researchers lack accessible tools for comprehensive analysis of *when* and *how* semantic structure emerges, whether during training or in deployed models. As a result, the relationship between optimisation objectives and representational development remains poorly understood, despite its importance for training decisions, generalisation, and debugging failures. This blind spot hinders both scientific understanding of language acquisition in LMs and the practical optimisation of model training workflows.

We introduce **TRACE** (Tracking Representation Abstraction and Compositional Emergence), a modular toolkit for training and inference-time interpretability analysis of transformer models.[1] TRACE exposes internal learning dynamics through lightweight instrumentation requiring minimal code changes, enabling structured tracking of semantic emergence, representational compression, and loss landscape evolution—signals that are invisible to conventional training logs.

To support systematic experimentation, TRACE pairs with **ABSynth**, a controllable synthetic corpus generator that provides structured linguistic data with aligned annotations[1]. While TRACE can analyse any transformer training process, ABSynth enables precise, repeatable experiments grounded in known linguistic structures.

[1]TRACE & ABSynth are released under GPLv3 License.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 806–820
November 4-9, 2025

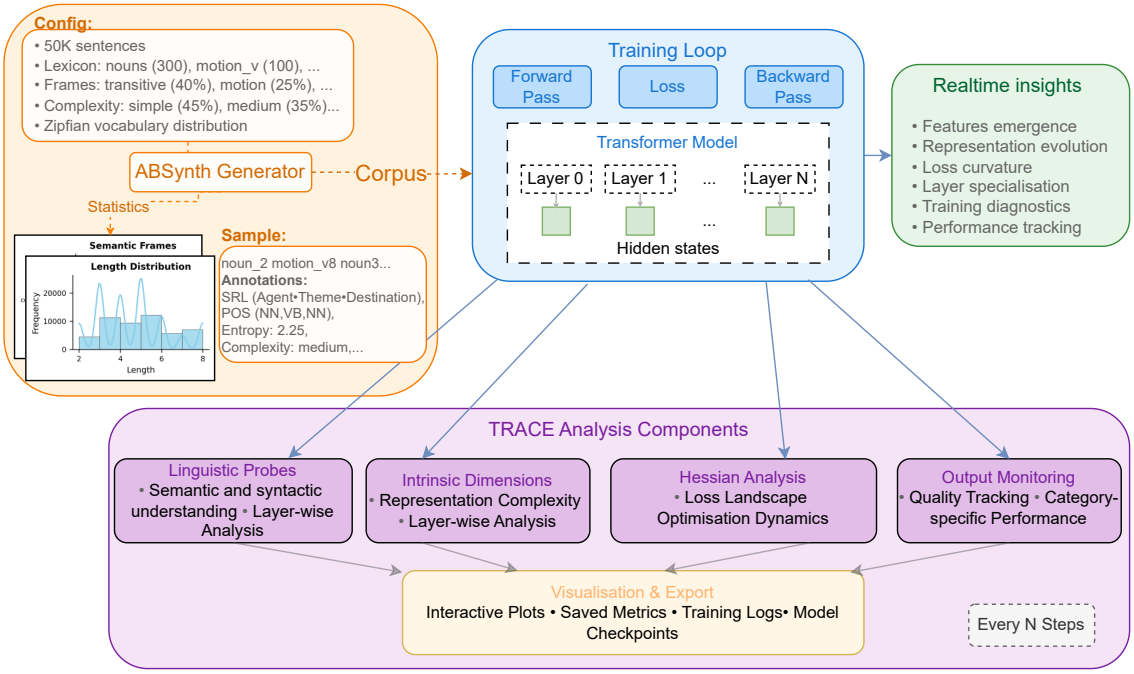


Figure 1: TRACE system overview. ABSynth generates controlled linguistic datasets with explicit annotations, which feed into transformer training loops instrumented with lightweight analysis hooks. Four modular components track linguistic emergence: semantic/syntactic probes, intrinsic dimensionality analysis, Hessian landscape exploration, and output monitoring. All modules generate automated visualisations and can operate independently during training or post-hoc analysis.

TRACE is designed for researchers and developers seeking a modular, easy-to-integrate interpretability tool to better understand, debug, or intervene on transformer models. Demonstrations on decoder-only models trained with ABSynth reveal distinct developmental phases in syntax and semantics, compression of internal representations, and curvature shifts in optimisation dynamics, insights not available through traditional monitoring alone. Thus, it transforms training from an opaque optimisation process into an interpretable, research-ready workflow for studying representational learning and guiding practical model development, with additional support for inference-time analysis.

2 Overview

TRACE (Figure 1) is a modular toolkit for interpretability analysis of transformer training. It supports dynamic monitoring of linguistic, geometric, and optimisation signals, integrating seamlessly into training loops and model evaluation pipelines. It is also model-agnostic, extensible, and requires only minimal code changes. By default, it supports the construction of standard transformer architectures as described in (Vaswani et al., 2017), and adapting to custom models typically requires only minor adjustments. A companion dependency, ABSynth, supplies linguistically annotated training data with controllable complexity.

2.1 Design Overview

The toolkit consists of two core modules and one external dependency:

- **Monitoring Hooks**: Injected into the training loop or inference pipelines, these capture hidden states, gradients, and loss signals without interfering with model execution.
- **Analysis Modules**: Pluggable components for probing, intrinsic dimensionality estimation, loss analysis, and output-level diagnostics.
- **ABSynth (external dependency)**: A synthetic corpus generator that provides structured, annotated training data with controllable linguistic complexity.

2.2 Key Capabilities

TRACE combines four capabilities that are difficult to achieve simultaneously within existing interpretability frameworks:

- **Minimal Integration Effort:** Requires only a few lines of code to enable comprehensive analysis, with modular components that can be added, removed, or extended without code refactoring.
- **Flexible Analysis Scope:** Supports both live analysis during training and post-hoc inspection of trained models, continuously monitoring linguistic probes, representational geometry, and loss curvature.
- **Automated Diagnostics:** Automatically plots learning curves, phase changes, and divergence indicators, with outputs saved in structured formats (e.g. JSON, CSV) for further analysis.
- **Actionable Insights:** Reveals how design and training choices shape model dynamics. Model size influences stabilisation patterns, often correlating with intrinsic dimensionality; removing components can increase curvature volatility without preventing generalisation. Hyperparameter changes may accelerate probe gains but often lead to sharper curvature. Such contrasts provide concrete guidance for early stopping, architecture tuning, and training schedule adjustments.

3 Controlled Corpus Generation (ABSynth)

ABSynth is a synthetic corpus generator that produces linguistically annotated datasets with explicitly controlled syntactic and semantic structures. Inspired by frame semantics (Baker et al., 1998; Fillmore, 1982), ABSynth models language as a composition of semantic frames, participant roles, and lexical units. Unlike natural corpora, which entangle linguistic properties or lack annotations, ABSynth provides fine-grained control over linguistic dimensions while generating English-like data. This enables systematic experiments on representational learning, generalisation, and abstraction.

Minimal API and Basic Usage

ABSynth supports rapid corpus generation with a simple interface:

```
from absynth.corpus import
    SyntheticCorpusGenerator
generator = SyntheticCorpusGenerator()
corpus = generator(10000)
corpus.save("corpus_full.json", indent=2)
```

A typical generated sentence appears as:

```
noun139    transitive_verb8s    noun40
preposition4 location2
```

This sentence instantiates the `transitive_action` semantic frame with structured annotations automatically generated during corpus creation.

Sentence Structure and Generation

Each sentence is constructed from a semantic frame containing a syntactic template. Frames specify roles (e.g., Agent, Patient, Location), which are mapped to lexical items from Zipfian-distributed pools (Zipf, 1949; Piantadosi, 2014). Templates determine constituent order (e.g., `[arg0, verb, arg1, prep, arg2]`) and syntactic structure. ABSynth automatically generates token-level annotations, including part-of-speech tags, semantic roles with positions, and metadata as follows:

- **Semantic Roles:**
  - `noun139` — Agent (position 0)
  - `noun40` — Patient (position 2)
  - `location2` — Location (position 4)
- **POS Tags:** `[NN, VB, NN, IN, NN]`
- **Metadata:** Complexity = `medium`; Entropy = 2.25; Length = 5

Corpus properties can be customised via frame and complexity distributions:

```
corpus = generator.generate_corpus(
    num_sentences=10000,
    complexity_distribution={
        "simple": 0.55, "medium": 0.35,
        "complex": 0.1},
    semantic_frame_distribution={
        "transitive_action": 0.4,
        "motion": 0.3,
        "intransitive_action": 0.3})
```

Key controllable dimensions include: (i) **Lexical selection:** Tokens sampled from role-specific pools with Zipfian (Zipf, 1949; Piantadosi, 2014) distributions; (ii) **Frame selection and creation:** Controls argument structure and semantic relations; (iii) **Complexity distribution:** Specifies proportions of simple, medium, and complex constructions; and (iv) **Statistical properties:** Entropy profiles, collocational strengths, and predictability patterns.

4 Dynamic Linguistic Tracking (TRACE)

As a sample use case, we present results from training a 2-layer decoder-only transformer with 3 attention heads, 384-dimensional MLP, and 96-dimensional hidden states on an ABSynth-generated corpus of 50K examples (batch size 128, learning rate $1e-3$, 70K steps). All analysis modules were enabled: probing, intrinsic dimensionality, Hessian curvature, and output diagnostics. While we show a combined diagnostic plot for conciseness, each module also outputs standalone visualisations and structured logs. A complete walkthrough example of the tool usage, including detailed inputs, outputs, and interpretations, is provided in Appendix C.

API Integration

TRACE wraps a standard training loop with minimal configuration:

```
from trace.training import Trainer,
    TrainingConfig
config = TrainingConfig(
    epochs=10,
    track_interval=500,
    save_visualization=True
)
trainer = Trainer(config, tokenizer, model)
trainer.train(train_loader, val_loader)
```

All modules run independently, log structured outputs (e.g., JSON, CSV), and support training and post-hoc visualisation.

Linguistic Structure Probing. Semantic and syntactic probes can be applied to any layer, supporting both static and dynamically updated models. Probes return confidence per linguistic role, revealing where and when linguistic features emerge and consolidate.

```
config = TrainingConfig(
    track_semantic_probes=True,
    probe_load_paths={
        (0, 'decoder'):
            './probes/pos_layer0.pt',
    }
)
```

Figure 2a shows semantic probe confidence for the decoder layer 1. Core roles (e.g., AGENT, ACTION) stabilise early, while adjunct roles (e.g., LOCATION, DESTINATION) fluctuate, indicating delayed consolidation. Several dips in confidence align with changes in curvature and intrinsic dimensionality (Figure 2b), suggesting linked representational shifts. The key capabilities of this module include:

- **Multi-label probing:** Simultaneous detection of multiple linguistic features
- **Category-specific analysis:** Performance breakdown by syntactical and semantic categories
- **Emergence tracking:** Identification of critical learning phases for different linguistic structures

Intrinsic Dimensionality Analysis. Representational complexity is estimated through intrinsic dimensionality (ID) metrics, including TwoNN (Facco et al., 2017) and PCA-based estimators (Cangelosi and Goriely, 2007):

```
config = TrainingConfig(
    track_intrinsic_dimensions=True,
    id_method="TwoNN")
```

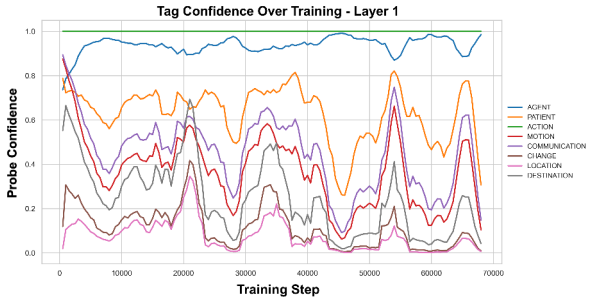
ID can be reported per layer or model average, capturing when and where representation compression or expansion occurs across model layers. Figure 2b shows an early drop in average dimensionality followed by a sharp rebound and stabilisation. These trends suggest a shift from early overcompression to stabilised abstraction. Such transitions coincide with early probe confidence dips, supporting the view that dimensionality evolution reflects changes in internal representational. These measures can be used to detect abstraction shifts, potential representational bottleneck or underutilised representational capacity.

Optimisation Landscape Analysis. Loss curvature is tracked using Lanczos-based Hessian approximations (Lanczos, 1950). The framework captures gradient–Hessian alignment, dominant eigenvalue shifts, trace, and other spectral metrics:

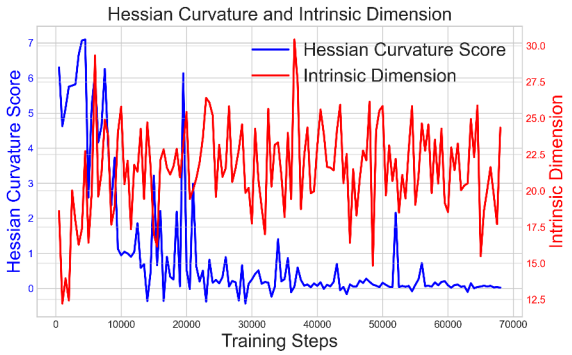
```
config = TrainingConfig(
    hessian_n_components=10,
    track_component_hessian=True,
    track_gradient_alignment=True)
```

In Figure 2b, early spikes in curvature precede shifts in ID, revealing a transition from sharp loss regions to flatter ones, memorisation and generalisation. Additional moderate spikes later in training suggest ongoing adjustments during fine-tuning phases. Beyond this visualisation, the tool simultaneously monitors other optimisation metrics, including:

- **Curvature evolution:** Changes in loss landscape sharpness and flatness, trace, and dominant eigenvalues.



(a) Probe confidence over semantic roles



(b) Intrinsic dimension and curvature evolution

Figure 2: **Sample TRACE visual diagnostics.** (a) Semantic probe confidence per role at decoder layer 1. TRACE tracks dynamic shifts, revealing phase transitions in linguistic encoding (e.g., mid-training dips and late recovery). (b) Joint evolution of Hessian curvature and average intrinsic dimensionality, exposing compression, abstraction, and optimisation phase shifts. Together, these metrics offer temporal insights missed by scalar loss alone.

- **Component analysis:** Separate tracking of components (e.g. attention, feed-forward) contributions
- **Memorisation detection:** Train/validation landscape divergence analysis
- **Gradient alignment:** Relationship between optimisation direction and principal curvature

Output-Level Diagnostics. Model predictions are analysed by token class and semantic role, enabling stratified accuracy reporting across linguistic categories. This diagnostic helps detect linguistic structural misalignment, where the model predicts the correct type (e.g., a noun or a patient role) but fails to recover the intended lexical item (e.g., `noun3` vs. `noun542`). Such errors indicate representational drift: the model learns the general grammatical form but lacks alignment to specific semantic slots or referents.

```
config = TrainingConfig(
    track_semantic_roles=True,
    semantic_roles_granularity='detailed' )
```

System Evaluation. This demonstration serves as a functional evaluation of TRACE. The example training run reveals distinct, interpretable dynamics across modules, semantic emergence in probe confidence, representational compression through intrinsic dimensionality drops, and curvature shifts in the optimisation landscape. The observed alignment between independent metrics further validates the utility of TRACE for diagnosing learning phase transitions. Full training configuration is provided in Appendix A, and further comparisons with existing interpretability tools appear in Table 1.

Research Questions and Applications. TRACE addresses fundamental questions in transformer interpretability: (i) **Emergence timing:** When and where do syntactic and semantic features first appear?; (ii) **Representational evolution:** Which layers compress or expand representational space during training?; (iii) **Generalisation patterns:** Is the model developing structural understanding or overfitting to surface patterns?; and (iv) **Training efficiency:** Can representation convergence inform early stopping decisions?

Automatic Visualisation and Reporting. All modules generate comprehensive visualisations automatically, including: (i) **Linguistic confidence evolution:** Per-category confidence scores across training steps; (ii) **Dimensionality trajectories:** Layer-wise ID evolution with compression detection; (iii) **Hessian landscape analysis:** Eigenvalue evolution and curvature dynamics; and (iv) **Performance breakdown:** Category-specific accuracy trends and structural alignment metrics. Results are also saved in standard formats (e.g., CSV, JSON) for further analysis and integration with external tools. By exposing model internals at different stages, TRACE makes transformer training interpretable, inspectable, and diagnostic by design.

5 Related Work

Post-hoc Representational Analysis. A common approach to interpreting language models involves probing classifiers or sparse autoencoders

Feature	TRACE (Ours)	Post-hoc Probing	Manual PCA / ID	TransformerLens
Timing of Analysis	During and after training	After training	After training	During and after training
Layer-wise Tracking	Yes	Yes	Yes	Yes
Temporal Resolution	High (per N steps)	Sparse (checkpoints)	Sparse	Sparse (checkpoints)
Model Support	Any custom PyTorch model	Any checkpoint	Any checkpoint	HuggingFace models only (limited)
User Effort	Low (1 config file)	High (custom scripts)	High	High (custom pipelines)
Causal Intervention Support	Partial (custom scripting required)	No	No	Partial (custom scripting required)
Emergence Detection	Yes	No	No	No
Training Support	Yes (native)	No	No	Partial (custom scripting required)
Supports Early Stopping	Yes (live signals)	No	No	No
Gradient/Loss Monitoring	Yes	Indirect	N/A	Yes

Table 1: Comparison of TRACE with post-hoc interpretability methods and TransformerLens. TRACE offers low-effort, modular interpretability with native training integration, unlike existing tools that require custom scripting and lack temporal tracking.

(SAEs) trained on frozen representations. Probes evaluate the presence of linguistic features using lightweight supervised models (Hewitt and Manning, 2019; Belinkov et al., 2018), while SAEs attempt to recover features in latent spaces in an unsupervised setting (Bricken et al., 2023; Kantamneni et al., 2025). While both techniques can be adapted for use during training, they are typically applied post hoc and require manual effort for their models construction, retraining, and integration. As standalone tools, they provide limited visibility into when features emerge or how they evolve.

Attribution and Causal Methods. Attribution tools (e.g., integrated gradients (Sundararajan et al., 2017) and attention flow analysis (Voita et al., 2019)) and causal interventions (e.g., activation patching (Meng et al., 2022) and path patching (Wang et al., 2023)) identify critical inputs or subcircuits. However, they operate on static checkpoints and provide no insight into the dynamics of representational development.

Model Inspection Frameworks. Frameworks like BertViz (Vig, 2019), InterpretDL (Li et al., 2022), and TransformerLens (Nanda and Bloom, 2022) enable weight and attention analysis. TransformerLens partially supports training-time instrumentation, but requires nontrivial modification. Platforms like TensorBoard or Weights & Biases (Biewald, 2020) track scalar metrics, offering no view into linguistic or geometric structure.

Controlled Datasets and Linguistic Benchmarks. Several datasets evaluate LMs' capabilities. Static benchmarks such as GLUE (Wang et al., 2018), BLiMP (Warstadt et al., 2020), and COGS (Kim and Linzen, 2020) evaluate linguistic competence but lack structural flexibility. CounterFact (Meng et al., 2022) and IOI (Wang et al., 2023) target specific phenomena, while mathematical datasets (Power et al., 2022; Saxton et al., 2019) offer training control but limited linguistic coverage. ABSynth provides extensible, richly annotated data aligned with TRACE's dynamic instrumentation.

Comparison with Existing Tools. Table 1 highlights how TRACE differs from typical interpretability pipelines and frameworks. It supports native training-time analysis, live monitoring, and plug-and-play extensibility with structured logging.

6 Extensibility and Integration

TRACE is implemented around a modular design where models are explicitly constructed within the framework rather than wrapped directly from external libraries. While models from Hugging Face and similar repositories expose high-level APIs, their internal states (e.g., layer outputs, normalization points, or intermediate activations) vary across implementations, making uniform integration difficult. To ensure consistent access to hidden representations and loss signals, TRACE provides a set of composable building blocks (e.g. encoder, decoder, attention, feed-forward modules) in `components.py`. New architectures can therefore be added by instantiating or extending these modules, while still benefiting from a standardised interface for monitoring and analysis. This approach requires users to

define their model structure within TRACE, but it guarantees compatibility with all analysis modules (probes, intrinsic dimensionality, Hessian curvature) without modification. In practice, this design balances flexibility for custom research architectures with stability in the interpretability pipeline. We leave direct integration with Hugging Face models as future work.

```
from trace.components import Encoder, Decoder
from trace.models import Transformer
from trace.trainer import Trainer,
    TrainingConfig

# Define custom encoder-decoder model
encoder = Encoder(num_layers=6, d_model=512,
    num_heads=8, d_ff=2048)
decoder = Decoder(num_layers=6, d_model=512,
    num_heads=8, d_ff=2048)
model = Transformer(encoder=encoder,
    decoder=decoder, vocab_size=32000)

# Configure TRACE analyses
config = TrainingConfig(
    track_semantic_probes=True,
    track_intrinsic_dimensions=True,
    track_hessian=True,
    track_interval=500,
)

trainer = Trainer(config, tokenizer, model)
trainer.train(train_loader, val_loader)
```

This example illustrates how a user can construct a model from TRACE components and automatically enable monitoring tools during training.

7 Conclusion

We presented **TRACE**, a modular toolkit for interpretability of language models, and **ABSynth**, a companion corpus generator for controlled linguistic experimentation. TRACE enables in-training analysis of linguistic, geometric, and optimisation signals, shifting interpretability from a post-hoc task to a dynamic diagnostic process. Applied to a decoder transformer, TRACE revealed phase-structured training dynamics, including semantic emergence, representational compression, and curvature shifts, not visible through traditional training metrics. These signals also enable targeted interventions such as early stopping, architecture tuning, and training schedule adjustment. The tool's modular design allows each component to operate independently or in combination, enabling a broad range of interpretability research. While current demonstrations use synthetic corpora, TRACE can generalise to natural language models. Planned extensions include Hugging Face integration and support for automatic annotation via tools such as NLTK, further broadening TRACE's scope for real-world applications. By making internal learning dynamics observable and actionable, TRACE advances both theoretical insight and practical control over LM analysis.

Limitations

While TRACE advances training-time interpretability, several limitations remain. Our demonstrations primarily rely on ABSynth, which provides controlled conditions but does not capture the full complexity of natural corpora. Although TRACE can be trained on arbitrary text data, it does not currently provide built-in annotation pipelines for linguistic probes; users must preprocess data or connect external tools to supply supervision. Hessian-based curvature estimates are efficient for medium-scale models but remain costly for very large architectures. Finally, TRACE highlights correlations between representational, linguistic, and optimisation signals, but does not establish causality. These limitations point to natural directions for future work, including broader integration with natural-language annotations, more scalable curvature analysis, and causal interventions built on TRACE's monitoring capabilities.

Acknowledgements

This work was partially funded by the SNSF project RATIONAL (200021E_229196), the CRUK National Biomarker Centre, and supported by the Manchester Experimental Cancer Medicine Centre and the NIHR Manchester Biomedical Research Centre.

References

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*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Leonard Bereska and Stratis Gavves. 2024. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*. Survey Certification, Expert Certification.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Richard Cangelosi and Alain Goriely. 2007. Component retention in principal component analysis with application to cdna microarray data. *Biology direct*, 2:1–21.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. 2017. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140.

Charles J. Fillmore. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. 2025. Are sparse autoencoders useful? a case study in sparse probing. *arXiv preprint arXiv:2502.16681*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and 1 others. 2020. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*.

Cornelius Lanczos. 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards*, 45(4):255–282.

Xuhong Li, Haoyi Xiong, Xingjian Li, Xuanyu Wu, Zeyu Chen, and Dejing Dou. 2022. Interpretdl: Explaining deep models in paddlepaddle. *Journal of Machine Learning Research*, 23(197):1–6.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36. ArXiv:2202.05262.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

Nostalgebraist. 2020. interpreting GPT: the logit lens.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21:1112–1130.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

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.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

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*.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Zhixue Zhao and Boxua Shan. 2024. "reagent: Towards a model-agnostic feature attribution method for generative language models". In *Proceedings of AAAI Workshop on Responsible Language Models*.

George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

A Experimental setup

A.1 Software Environment and Dependencies

All experiments were run on a single NVIDIA RTX A6000 GPU. The environment was built using Python 3.11.13 and configured via Conda. Core libraries include PyTorch (v2.7.1) and ABSynth (v0.1.1) for synthetic dataset generation (with the same requirements listed here). Other tools include scikit-learn (v1.7.0), SciPy (v1.15.3), NumPy, and scikit-dimension (v0.3.4). Visualisation components rely on Matplotlib (v3.10.3), Matplotlib-inline (v0.1.7), and Seaborn (v0.13.2). Additional packages include tqdm (v4.67.1). The full setup is specified in the environment.yml file included with the code repository and is fully reproducible.

A.2 Model Architecture

We use a lightweight Transformer architecture to enable rapid training and frequent checkpointing. This allows TRACE to track representational evolution with minimal overhead.

- **Architecture:** Decoder-only Transformer
- **Layers:** 2 decoder layers
- **Hidden size:** 96
- **Feed-forward dimension:** 384
- **Attention heads:** 3
- **Maximum sequence length:** 16

A.3 Training Configuration

- **Corpus:** 25,000 synthetic sentences
- **Batch size:** 128
- **Epochs:** 500
- **Learning rate:** 1×10^{-4}
- **Optimizer:** Adam
- **Tracking frequency:** Every 500 steps

B Licensing

TRACE is released under the GNU General Public License v3.0 (GPLv3). This license ensures that the software remains free and open-source. Users are free to use, modify, and distribute the code, provided that derivative works also adopt the GPLv3 license.

C Example Use Case

This appendix illustrates TRACE's functionality and interpretability workflow. It includes an end-to-end case study using a transformer model trained on an ABSynth-generated corpus. This walkthrough demonstrates how TRACE can be used to monitor semantic emergence during training and interpret the resulting behaviour via multiple analytical lenses.

C.1 ABSynth Data Generation

We provide an example of data generation, where we set all the parameters, but **emphasise** that the user has the liberty to use the default configurations by specifying only the number of examples needed, as shown in Section 3.

C.1.1 ABSynth Input Configuration

```
from absynth.lexicon import Vocabulary, LexiconGenerator

# Define vocabulary sizes
vocab = Vocabulary({
    "noun":300, "transitive_verb":40, "intransitive_verb":25,
    "communication_verb":20, "motion_verb":20, "change_verb":15, "adjective":40,
    "adverb":25, "location":150, "temporal":35, "instrument":25, "preposition":15,
    "conjunction":10, "determiner":8
})

lexicon = LexiconGenerator(
    vocab_sizes=vocab,           # Custom vocabulary sizes
    num_clusters=5,              # Number of semantic clusters to create
    zipfian_alpha=1.05,             # Alpha parameter for Zipfian distribution
    error_bias=0.00001,              # Error bias for word generation
    random_seed=42               # For reproducible generation
)

from absynth.sentence import SentenceGenerator, FrameManager
templates = FrameManager()
sentence_generator = SentenceGenerator(lexicon, templates)

from absynth.corpus import SyntheticCorpusGenerator
generator = SyntheticCorpusGenerator(lexicon=lexicon, sentence_generator=sentence_generator)
corpus = generator.generate_corpus(
    num_sentences=25000,
    complexity_distribution={"simple": 0.55, "medium": 0.35, "complex": 0.10},
    semantic_frame_distribution={
        "transitive_action": 0.1,
        "transitive_with_location": 0.15,
        "motion_with_source": 0.15,
        "temporal_action": 0.15,
        "instrumental_action": 0.15,
        "multi_action": 0.15,
        "temporal_complex": 0.15,
    }
)
from absynth.visualization import Visualizer
visualizer = Visualizer(log_dir='./plots')
visualizer.visualize(corpus)
```

This setting creates a corpus of 25K sentences, with a mix of syntactic templates and semantic frames. Sentences vary in their surface structure but preserve underlying argument structures aligned with FrameNet-like role schemas.

C.1.2 Sample Generated Data

A representative sentence from the generated corpus is:

```
Input: "noun139 transitive_verb8s noun40 preposition4 location2"
```

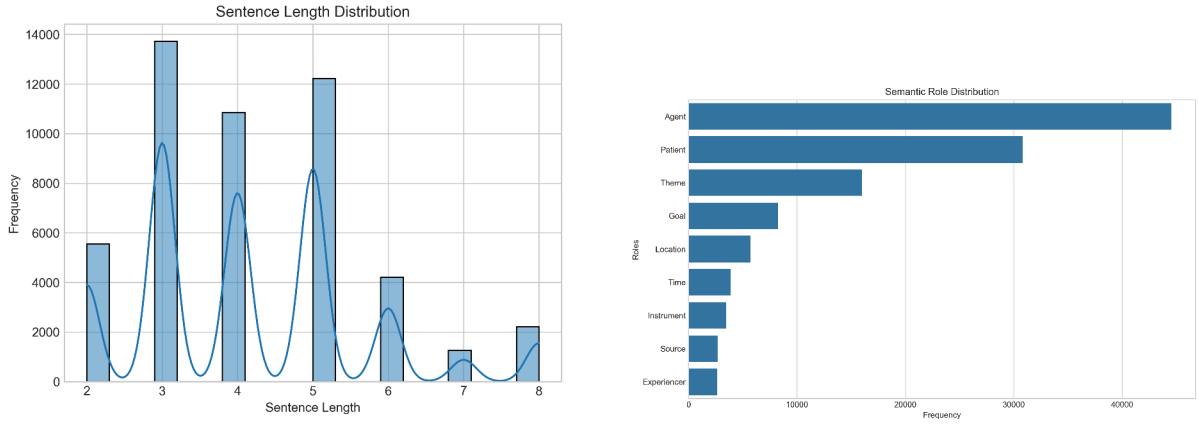
The corresponding annotations include both structural and semantic metadata:

```
{
  "sentence": "noun139 transitive_verb8s noun40 preposition4 location2",
  "semantic_roles": {
    "noun139": {"role": "Agent", "position": 0},
    "noun40": {"role": "Patient", "position": 2},
    "location2": {"role": "Location", "position": 4}
  },
  "pos_tags": ["NN", "VB", "NN", "IN", "NN"],
  "metadata": {
    "complexity": "medium",
    "frame": "transitive_action",
    "length": 5,
    "entropy": 2.25
  }
}
```


This format enables fine-grained probing of semantic representations during training and evaluation. Each token is associated with a role and position, and every sentence is traceable to its underlying generation rule.

C.1.3 Corpus Statistics and Visualisation

Figure 3, and Table 2 represents a sample of the visualisation and statistical summarisation that can be generated for each corpus.



(a) Hessian trace over training steps.

(b) Gradient and Hessian norm dynamics.

Figure 3: (Sample statistical visualisation of ABSynth: (Left) Distribution of sentence lengths in the generated ABSynth corpus. (Right) Frequency distribution of semantic roles across the dataset.

Metric	Value
Total sentences	25,000
Vocabulary size	7910 tokens
Average sentence length	4.17012 tokens
Semantic frame coverage	6 distinct frames
Zipfian compliance	$\alpha = 1.05$

Table 2: Sample ABSynth corpus summary statistics.

C.2 Model Training Setup

C.2.1 Model Configuration

To evaluate TRACE's interpretability modules in a controlled training setting, we trained a lightweight decoder-only Transformer on the ABSynth corpus introduced in Section 4. This setup enables fine-grained

temporal analysis of representational and optimisation dynamics. While this demonstration focuses on decoder-only architectures, we note that TRACE also supports encoder-only and encoder-decoder models.

```
model_config = TransformerConfig(
    model_type="decoder_only",
    vocab_size=7000,
    d_model=96,              # Hidden dimension
    num_heads=3,             # Attention heads
    num_decoder_layers=2,    # Number of layers
    d_ff=384,               # Feed-forward dimension
    max_seq_length=16,      # Maximum sequence length
    dropout=0.1
)
```

C.2.2 Training Configuration

```
training_config = TrainingConfig(
    epochs=30,
    learning_rate=1e-3,
    batch_size=128,

    # Enable all analysis modules
    track_hessian=True, # Loss landscape analysis
    track_linguistic_probes=True, # POS understanding
    track_semantic_probes=True, # Semantic role understanding
    track_intrinsic_dimensions=True, # Representation dimensionality
    track_pos_performance=True,  # Output POS accuracy
    track_semantic_roles_performance=True,  # Output semantic accuracy
    probe_load_paths=probe_paths,
    semantic_probe_load_path=semantic_probe_paths,

    # Analysis frequency
    track_interval=500,      # Every 500 steps
    save_visualization=True
)
```

C.3 TRACE Analysis Results

This section presents representative outputs from TRACE's core modules, applied to the synthetic training run described in Sections C.1–C.2. These examples demonstrate how TRACE captures distinct learning dynamics across layers, linguistic abstractions, and training stages. All inputs and outputs are automatically logged by TRACE when the corresponding tracking module is enabled by the user.

C.3.1 Linguistic Probe Evolution

We trained semantic role probes on model checkpoints throughout training to extract interpretable structure from hidden representations. The probes are trained to predict semantic roles and POS based on intermediate activations. This allows us to track when and where in training various linguistic categories emerge.

Input: Hidden states h from decoder layers 0 and 1, sampled at regular training intervals (500 in this experiment).

Output Example: Figure 4 shows probe confidence scores for Layer 1 across training steps. Confidence values indicate the strength of alignment between hidden states and semantic roles.

Interpretation:

- **Core Role Emergence:** AGENT, ACTION, and PATIENT show rapid gains in probe confidence, stabilising early in training. These roles are central to the predicate-argument structure, and their early emergence suggests the model internalises core semantics first.
- **Adjunct Role Delay:** LOCATION and other adjunct roles show lower and more delayed gains, indicating that peripheral semantic categories are acquired later and less robustly.

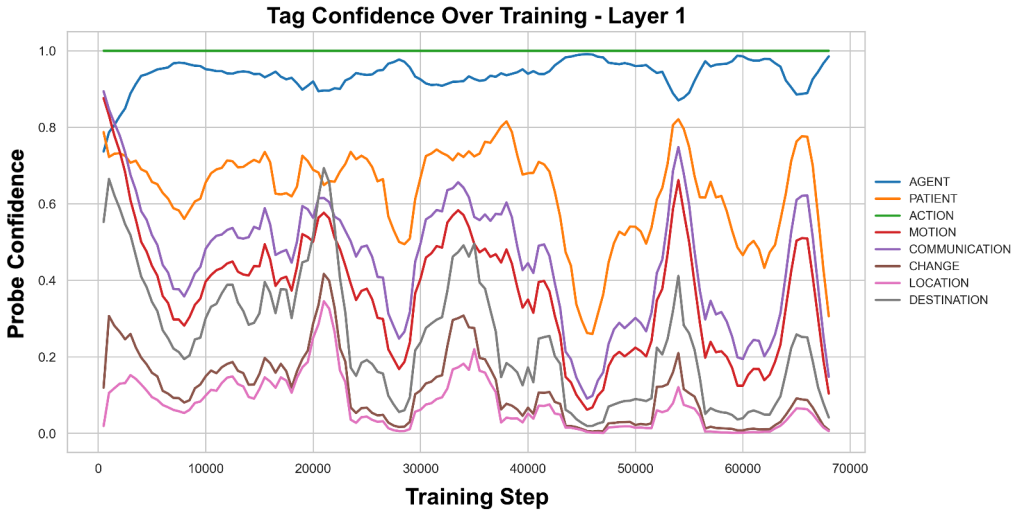


Figure 4: Semantic role probe scores across training steps. Layer 1 shows alignment with core roles over time with steps where there was higher alignment and other for lower.

- **Fluctuation Signals Reorganisation:** Adjunct roles exhibit greater fluctuations in confidence across training steps. These fluctuations may reflect ongoing representational reorganisation as the model adjusts the alignment of semantic roles across layers and training stages.
- **Semantic Acquisition Hierarchy:** The emergence pattern suggests a learning hierarchy, with core argument structure learned before modifiers, consistent with theoretical linguistics and prior neural acquisition work.
- **Alignment with Representational Dynamics:** Confidence dips at intermediate steps (e.g., 10k–25k) align with known transitions in intrinsic dimensionality and Hessian curvature, reflecting internal restructuring as the model shifts from memorisation to abstraction.

C.3.2 Intrinsic Dimensionality Analysis

Input: Hidden states h from decoder layers 0 and 1, sampled at regular training intervals (500 in this experiment).

Output Example: Figure 5 and Table 3 show a sample of the ID analysis output, the figure shows the overall ID over the whole course of training steps, while the table shows a sample of the results. They illustrate the evolution of representational complexity.

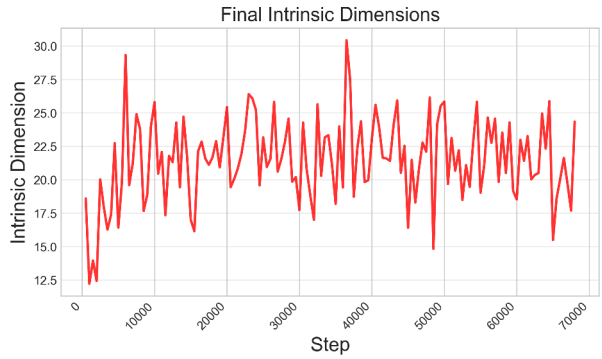


Figure 5: Intrinsic dimensionality across training steps.

Training Step	Avg. ID
500	18
2,000	12
5,000	16
10,000	25
25,000	23

Table 3: Sample averaged Intrinsic dimensionality over layers across training steps.

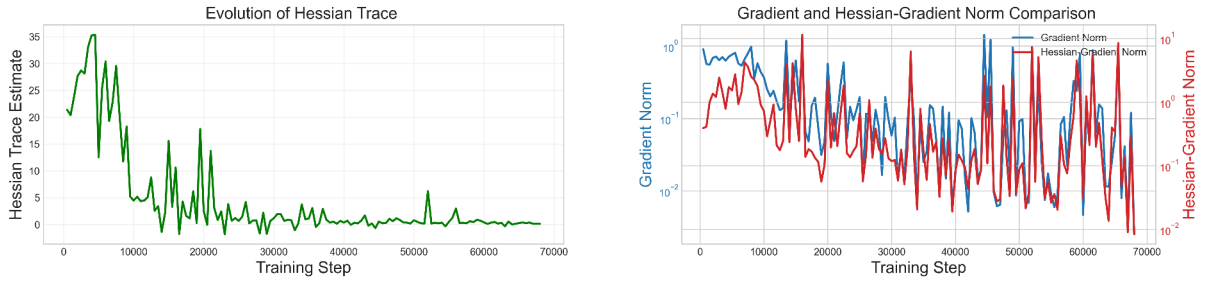
Interpretation:

- **Initial Compression and Expansion Pattern:** Early and sharp drop, 18 to 12 (steps 500-2,000) followed by and expansion to 25, patterns indicating transitioning from simple pattern matching to more complex linguistic understanding.
- **Dynamic Training Regime:** Persistent oscillations between ID 17-25 throughout training suggest active representational restructuring rather than static convergence, reflecting adaptive complexity based on linguistic content.
- **Cross-Metric Transitions:** Notable jumps in intrinsic dimensionality coincide with decreases in Hessian trace and fluctuations in probe confidence. These temporal alignments suggest coordinated signals between them.

C.3.3 Hessian Landscape Analysis

Input: Loss gradients and second-order information calculated by our tool.

Sample output: Figure 6 demonstrates the evolution of Hessian-based metrics throughout training, revealing characteristic patterns of curvature dynamics and spectral properties.



(a) Hessian trace over training steps.

(b) Gradient and Hessian norm dynamics.

Figure 6: Evolution of curvature and norm characteristics during training.

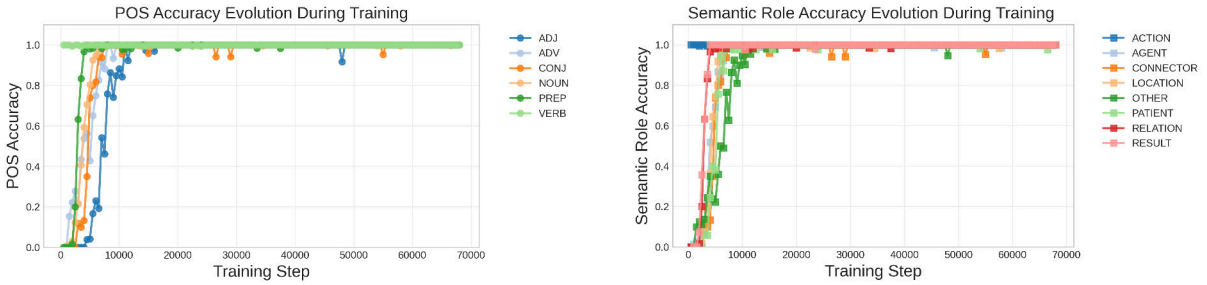
Interpretation:

- **Landscape Evolution Dynamics:** The Hessian trace exhibits a pronounced early peak followed by steady decrease, reflecting the model's transition from highly curved regions associated with memorisation to flatter regions indicating abstraction and generalisation. This is common in randomly initialised networks.
- **Curvature–Dimensionality Duality:** An interesting pattern is observed between curvature and intrinsic dimensionality: periods of rising dimensionality coincide with drops in the Hessian trace, suggesting that representational expansion is facilitated by local flattening in the optimisation landscape.
- **Structural Transition Markers:** Isolated spikes in Hessian curvature, e.g. around 55k steps, serve as indicators of local representational restructuring. These events temporally co-occur with inflection points in probing metrics and effective rank changes, underscoring the diagnostic value of second-order geometry for tracking internal phase shifts.

C.3.4 Output Quality Monitoring

Input: Model predictions at regular training intervals, evaluated against gold-standard semantic role and POS annotations generated by ABSynth.

Output Example: Progressive analysis of model predictions across POS and semantic role categories reveals distinct learning trajectories for different linguistic abstractions, different than earlier illustrated differences. Figure 7 demonstrates category-specific performance evolution, with concrete grammatical categories achieving rapid convergence while abstract semantic roles exhibit more gradual acquisition patterns.



(a) POS tagging accuracy across training.

(b) Semantic role prediction accuracy across training.

Figure 7: (Left) POS accuracy across training steps. (Right) Semantic role prediction accuracy.

Interpretation:

- **Semantic Role Acquisition Spectrum:** Roles such as `ACTION`, `RELATION`, and `RESULTS` are acquired early and maintain stable, high accuracy throughout training, indicating reliable identification of core predicate-argument relations. In contrast, roles like `LOCATION` and `PATIENT` converge more slowly and exhibit very few lingering fluctuations, reflecting their higher contextual variability and integration complexity. Notably, once core roles are learned (10K steps), the model rarely misclassifies them, even during intermediate representational reorganisations, suggesting firm grammatical competence, even if confidence (as seen in probing) continues to be refined.

- **POS Category Learning Hierarchy:** Among POS tags, `PREPOSITION` reaches high accuracy early, likely due to its limited variability and syntactic rigidity. `VERB` predictions stabilise quickly, note that the plotted category captures base forms rather than diverse verb types but that is also available in the tool. `NOUN` accuracy improves more gradually, reflecting greater lexical variability, but still outpaces more abstract modifiers like `ADJECTIVE` and `ADVERB`, which remain the most challenging throughout training.

- **Stable Predictions vs. Ongoing Representation Refinement:** Once acquired, role predictions remain stable throughout training and exhibit few regressions. This contrasts with the confidence patterns observed in probing layers, where internal representation alignment continues to shift, suggesting external outputs stabilise before internal semantics fully consolidate, which highlights a gap between external performance and internal semantic stability.

C.4 Summary

This walkthrough illustrates TRACE's ability to surface rich interpretability signals across diverse analytical lenses. By combining probing, dimensionality tracking, loss landscape analysis, and output performance monitoring, TRACE enables multi-faceted insight into model development over time. In this controlled ABSynth setting, TRACE reveals coherent trends in linguistic acquisition, such as early emergence of core roles, delayed abstraction in modifiers, and coordinated dynamics across representation, curvature, and output quality. Notably, we observe a divergence between surface-level grammatical accuracy and internal representational confidence, highlighting that models may produce correct outputs even as their internal abstractions continue to evolve. Such findings underscore the need for tools like TRACE to go beyond external metrics and expose latent dynamics during learning. These capabilities generalise across architectures and datasets, making TRACE a practical framework for probing both model behaviour and training dynamics in applied or research contexts.

MathBuddy: A Multimodal System for Affective Math Tutoring

Debanjana Kar[*1,2], **Leopold Böss**[*1], **Dacia Braca**[*1],
Sebastian Dennerlein[1], **Nina Hubig**[1], **Philipp Wintersberger**[1], **Yufang Hou**[1]

[1]IT:U Interdisciplinary Transformation University Austria
[2]IBM Research India

Correspondence: debanjana.kar1@ibm.com, {leopold.boess, dacia.braca}@it-u.at

Abstract

The rapid adoption of LLM-based conversational systems is already transforming the landscape of educational technology. However, the current state-of-the-art learning models do not take into account the student's affective states. Multiple studies in educational psychology support the claim that positive or negative emotional states can impact a student's learning capabilities. To bridge this gap, we present *MathBuddy*, an emotionally aware LLM-powered Math Tutor, which dynamically models the student's emotions and maps them to relevant pedagogical strategies, making the tutor-student conversation a more empathetic one. The student's emotions are captured from the conversational text as well as from their facial expressions. The student's emotions are aggregated from both modalities to confidently prompt our LLM Tutor for an emotionally-aware response. We have evaluated our model using automatic evaluation metrics across eight pedagogical dimensions and user studies. We report a massive 23 point performance gain using the win rate and a 3 point gain at an overall level using DAMR scores which strongly supports our hypothesis of improving LLM-based tutor's pedagogical abilities by modeling students' emotions. Our dataset and code are available at: https://github.com/ITU-NLP/MathBuddy.[1]

1 Motivation

The integration of AI tutors into educational platforms has rapidly evolved with the emergence of large language models (LLMs) (Chu et al., 2025), enabling natural and interactive support for learners across subjects, such as mathematics (Macina et al., 2023), and mitigating the issue of lack of qualified personal tutors.[2] Several studies (Kumar et al., 2024; Li et al., 2024; Wang et al., 2024) claim that

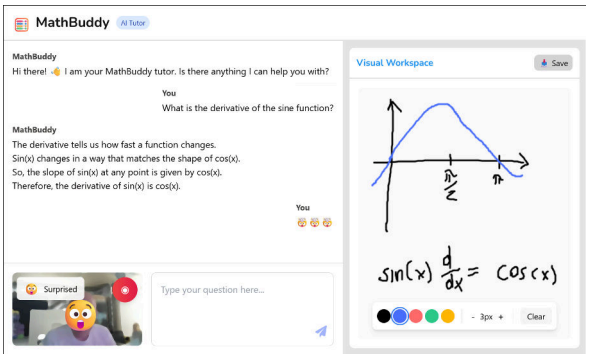


Figure 1: MathBuddy's user interface showcasing an example interaction using all functionality.

the integration of LLM-powered tutors facilitates formative self-regulated learning, where students learn and assess themselves at an independently decided pace.

Despite significant progress in instructional capabilities, most existing LLM-based tutoring systems lack emotional intelligence (Liu et al., 2024; Azerbayev et al., 2024), treating all learners homogeneously without accounting for their affective states. This can lead to disengagement, frustration, and ultimately suboptimal learning outcomes, especially in subjects such as mathematics, where learner anxiety is well-documented (Daker et al., 2023; Maki et al., 2024; Schoenherr et al., 2025).

Research in educational psychology has long emphasized the role of emotions in learning (Pekrun et al., 2002; Graesser et al., 2012). Negative affective states, e.g., confusion and frustration, can inhibit problem-solving ability and motivation, while positive emotions, such as engagement and curiosity, are positively correlated with learning gains (Pekrun, 2006; Tan et al., 2021; Zheng et al., 2023).

At the same time, recent advances in NLP and computer vision have opened up new avenues for multimodal affect recognition. Transformer-based models can infer affective cues from text with increasing accuracy (Chutia and Baruah, 2024; Ali

[*]Authors contributed equally to this work.
[1]Link to demo video: https://youtu.be/ZUjgmOw9GM0
[2]https://unesdoc.unesco.org/ark:/48223/pf0000385723

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 821–838
November 4-9, 2025

et al., 2024), while facial expression recognition using deep learning has reached practical levels of robustness (Makhmudov et al., 2024). However, existing LLM-based tutoring systems do not yet combine these modalities in real time to support accurate, empathetic adaptation in mathematics tutoring.

To address this gap, we present our AI-based math tutor, *MathBuddy* (see Figure 1), that models student emotions using both textual and visual modalities. More specifically, given a student's last utterance and facial expressions, *MathBuddy* extracts the student's emotions from both modalities and aggregates them into one of three classes (*Positive, Negative, or Neutral*). The emotion is then used to direct the LLM Tutor in one of the following ways: if the student is in a positive or neutral affective state, the student is challenged by *MathBuddy*, while a negative affective state causes the system to try to motivate the student.

We have evaluated the effectiveness of our system across eight different pedagogical dimensions (Maurya et al., 2025) through both automatic evaluation (Section 3) and real-time user studies (Section 4). The results from both the evaluation strategies emphasize the importance of modeling student emotions in tutoring strategies.

Our work's main contributions are as follows:

1. We present the first emotionally-aware LLM tutoring system, *MathBuddy* that is grounded in education theory and adapts its response based on the student's affective state.

2. We develop an automatic evaluation system to evaluate the framework thoroughly on a recent math tutor benchmark across eight different pedagogical dimensions based on the concepts introduced in (Maurya et al., 2025).

3. We have annotated 224 student utterances with emotion labels and polarity to support the development of emotion recognition models from text.

Our code, annotated dataset and a link to the system demo video are publicly available at: https://github.com/ITU-NLP/MathBuddy.

2 *MathBuddy*: Design & Implementation

In real-life tutoring scenarios, tutors often adapt their pedagogical strategies to the student's present emotional state (Lin et al., 2022). However, hardly any of the state-of-the-art AI-powered tutoring systems consider modeling students' emotions in this process. Our proposed system, *MathBuddy*, aims to address this gap by leveraging multimodal interaction channels. *MathBuddy* models one-on-one interactions between a student and a tutor in the domain of Mathematics. In this section, we provide a detailed breakdown of the system's implementation process.

2.1 Emotion Recognition from Text and Webcam Data

Since *MathBuddy* is a conversational system, we try to gauge the student's affective state by extracting the emotion from the student's turn in the conversation. However, a major bottleneck we encountered in this task is the lack of such an annotated student-tutor conversational dataset. To overcome this challenge, we annotated the student turns in the hard version of the MathDial-Bridge dataset (Macina et al., 2025). Inspired by D'Mello and Graesser (2012) work on affect in learning contexts, we assumed that students' emotions can be multifaceted and vary in intensity. Hence, we modeled emotion extraction as a multi-label classification task with three target states: *Boredom*, *Engagement*, and *Neutral* (D'Mello and Graesser, 2012). To capture the intensity of each student's emotional state at every turn, we assigned a polarity score on a scale from 0 to 2, where 0 corresponds to low intensity and 2 to high, thus serving as an indicator for arousal (Posner et al., 2005). Four annotators with academic backgrounds, aged between 27 and 38, participated in the annotation process. After an initial round, which yielded a Cohen's Kappa inter-annotator agreement score of 40%, three annotators took part in the conflict adjudication process. Finally, 224 unique student turns were annotated ensuring an equal distribution across the labels. We reserve the manually annotated dataset as our test set and generate a noisy student-tutor annotated dataset using DeepSeek-R1-Distill-LLama-70B (DeepSeek-AI, 2025) as our training dataset. We fine-tune multiple BERT-based models on the silver-labeled training data. The best model (Sanh, 2019) reports an accuracy and F1-score of 61.8% and 57.9% respectively. The detailed results are available in Appendix A.

In addition to text-based emotion recognition, the system processes webcam data to extract a student's affective states based on the student's facial expression.

We use the *face-api.js* package (Mühler, 2024) for the task of detecting students' emotions through their facial expressions. The *face-api.js* package represents a JavaScript library that builds on *tensorflow.js* to provide functionality related to human faces, such as face recognition, face landmark detection, and face expression recognition. The system utilizes its lightweight and fast in-browser face expression recognition. It is configured to recognise the emotions *Happy*, *Sad*, *Angry*, *Surprised*, *Fearful*, *Disgusted*, and *Neutral*, with *Neutral* being the default when no face is being detected. With our current configurations and considering the mapping explained later in Subsection 2.2, *face-api.js* reports an accuracy of 76% and an F1-score of 71% on our test set. The test set comprises 151 images from FERAC (Das Chaiti and Afrin, 2024) and 38 images from the Facial Emotion Recognition Dataset (Data, 2023), combined to ensure diversity across age groups as well as the inclusion of individuals diagnosed as non-neurotypical.

Given that the system requires all emotional states to be attributed to the specific timestamp of a message, the face emotion samples have to be aggregated over the interval bounded by two messages. For this step, the system considers only the changes in recognised emotion, grouping equal consecutive emotion states. This way, each group can be assigned a duration as well as an age—the time from now to the last sample within the group. Aggregation computes a weighted sum of the durations of all groups associated with one specific emotion, choosing the emotion with the highest sum as the result. The required weights are derived from a half-life decay function employing a group's age and a fixed half-life of 120 seconds. See Algorithm 1 for a specific description.

Algorithm 1 Temporal emotion aggregation.

Require: Sequence of emotion samples $S = \{(e_1, t_1), \dots (e_n, t_n)\}$ sorted by time t
Initialize group list $G \leftarrow [\,]$
Current emotion $e_c \leftarrow e_1$, start time $t_s \leftarrow t_1$
for $i = 2$ to n **do**
  if $e_i \neq e_c$ **then**
    Add new group (e_c, t_s, t_{i-1}) to G
    $e_c \leftarrow e_i$, $t_s \leftarrow t_i$
  end if
end for
Add final group (e_c, t_s, t_n)
Initialize emotion score map $S_e \leftarrow \{\}$
Half-life constant $\lambda \leftarrow \ln(2)/120$
for each group $(e, t_s, t_e) \in G$ **do**
  duration $d \leftarrow t_e - t_s$
  age $a \leftarrow \text{now} - t_e$
  weight $w \leftarrow \exp(-\lambda \times a)$
  $S_e[e] \leftarrow S_e[e] + d \times w$
end for
$e^* \leftarrow \arg\max_e S_e[e]$
return e^*

2.2 Multimodal Emotion Aggregation

Users are assumed to feel only one unique emotion at any given time. Therefore, the system aims to approximate this emotion by compiling all sampled emotional states into a single one. Each modality ideally increases the accuracy of this aggregation.

A straightforward approach is to project the emotions sampled from both modalities to a common value space to allow them to be merged. This value space holds three primitive emotion states: *Positive*, *Neutral*, and *Negative*.

The emotion mapping applied by the system can be expressed as follows:

$$m : E \to P$$

$$m(e) = \begin{cases} \text{Positive}, & \text{if } e \in E_{\text{positive}} \\ \text{Neutral}, & \text{if } e \in E_{\text{neutral}} \\ \text{Negative}, & \text{if } e \in E_{\text{negative}} \end{cases}$$

where E denotes the set of all recognizable emotions across modalities, and P denotes the set of primitive categories: *Positive*, *Negative*, and *Neutral*. Note that P is a subset of E used for simplified downstream processing. The specifics of how each emotion detected from either modality is mapped to one of the above primitive emotion states is detailed in Appendix G. Each emotion state produced by either emotion recognition method can be directly mapped to these primitives, including their confidence values. Notably, states produced via face-based recognition are mapped before aggregation. Merging the resulting values considers the following rule: *Positive* and *Negative* are always preferred over *Neutral*; confidence is used as a secondary criterion if both values are not equal to *Neutral*. For example, a text-based positive emotion with a confidence of 50% and a contradicting facial-based negative emotion with a confidence of 70% are reconciled into a negative emotion. Hence, the system assumes that any hint of non-neutral

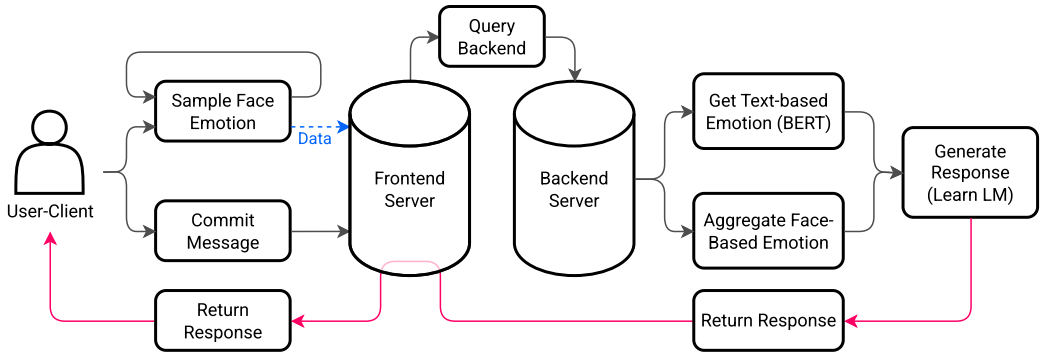


Figure 2: An overview of the system's logical flow, showcasing the core components.

emotion implies that neutrality is not given. This behavior appears beneficial as a non-neutral emotional state should enable better adjustment of the currently applied pedagogical strategy.

2.3 Emotion-Aware LLM Tutor

Inspired by educational psychology literature (Pekrun et al., 2002; Graesser et al., 2012), we design our system to map dynamically extracted student emotions to relevant pedagogical strategies in order to enhance the learning experience for the student. After extracting and aggregating multi-modal student emotions, we finally map it to relevant pedagogical strategies such that:

- *Positive* student emotion, the LLM tutor is prompted to *challenge* the student;
- *Neutral* or a *Negative* emotion, the tutor is prompted to *motivate* the student.

Please refer to Appendix C for detailed prompts.

2.4 System Implementation

The system is divided into two core components: Frontend and Backend. The Frontend represents a web server implemented in TypeScript that serves the web-based client and a public API, providing all functionality needed by the said client. This way, all requests must go through the Frontend, while all other components remain hidden from the outside. These requests mainly include the tutor response generation, when the user submits a message, and applying the face based emotion recognition to the webcam input. The later can be configured to be handled in browser by *face-api.js* or by the backend via our own model. Additionally, the Frontend also represents the main storage, holding a primitive in-memory database.

The client employs a simple design consisting of two parts: a chat-based interface and a visual workspace. The chat-based interface visualizes the tutor-student conversation similar to established LLM UIs. In addition to a text field, this interface also features an optional webcam input. The client uses this webcam input to query the user's emotional state in regular intervals; it can be configured to give live feedback on the recognised emotion. The visual workspace offers space to take notes or sketch ideas via mouse input. This feature is an artifact of the idea to submit handwritten notes to the tutor, reserved for future research, owing to implementation constraints.

The Backend component acts as a REST API, implementing all functionality related to inference via Python. It both runs local models and employs external APIs to respond to queries made by the Frontend. This structure allows for adapting models or configurations at the Backend without necessitating changes to the Frontend. Its key endpoints include the tutor response generation based on conversation history and face-based emotion states, as well as face-based emotion recognition based on an image. An overview of the system's logical flow is illustrated in Figure 2.

3 Automatic Evaluation

For a comprehensive evaluation of the system, we evaluate *MathBuddy* through automatic evaluation strategies and a user study. In this section, we detail the automatic evaluation strategies along with an analysis of the quantitative results. We evaluate *MathBuddy* with different backend LLMs on an existing math tutor benchmark (Maurya et al., 2025) with emotion information from the input text only. We use the hard version of the MathDial Bridge dataset (Macina et al., 2023) containing 327 human-annotated conversations using two different metrics:

Table 1: Comparison study using Win Rate and DAMR across eight different pedagogical dimensions. The plus (+) version of each baseline represents our models. Best reported metrics are highlighted in bold. The overall score is a combined mean of the DAMR scores across the eight dimensions.

Model	Win Rate	Mistake Identification	Mistake Location	Answer Disclosure	Providing Guidance	Action-ability	Human-likeness	Coherence	Tutor Tone	Overall Score
QSLM	0.30	0.71	0.92	0.41	0.22	0.88	0.56	0.58	0.86	0.64
QSLM+	**0.37**	0.71	0.92	0.41	0.22	0.88	0.56	0.58	0.86	0.64
LlemmaMM	0.38	0.22	0.93	**0.53**	**0.35**	0.75	0.65	0.70	0.92	0.63
LlemmaMM+	**0.38**	**0.27**	**0.96**	0.50	0.30	**0.90**	**0.72**	**0.83**	**0.97**	**0.68**
LearnLM	0.59	0.96	1.00	0.65	0.32	0.99	0.93	0.91	1.00	0.85
LearnLM+	**0.82**	**1.00**	**1.00**	**0.69**	**0.37**	**0.99**	**0.98**	**0.98**	**1.00**	**0.88**

Table 2: Comparison study using Win Rate and DAMR across eight different pedagogical dimensions with a more complex prompt.

Model	Win Rate	Mistake Identification	Mistake Location	Answer Disclosure	Providing Guidance	Action-ability	Human-likeness	Coherence	Tutor Tone	Overall Score
QSLM	0.33	0.52	**0.94**	0.49	0.26	**0.86**	0.61	**0.68**	**0.92**	0.66
QSLM+	**0.35**	**0.57**	0.93	**0.49**	**0.28**	0.84	**0.61**	0.67	0.91	**0.66**
LlemmaMM	0.41	**0.86**	**1.00**	**0.53**	**0.23**	**0.99**	0.66	0.62	**1.00**	**0.74**
LlemmaMM+	**0.42**	**0.22**	**0.95**	0.49	**0.32**	**0.82**	**0.66**	**0.72**	**0.95**	**0.64**
LearnLM	0.46	0.87	0.99	0.51	0.21	1.00	0.65	**0.63**	1.00	0.73
LearnLM+	**0.46**	**0.87**	**0.99**	**0.57**	**0.24**	**1.00**	**0.67**	**0.59**	**1.00**	**0.74**

- **Win Rate**: Rate at which reward model (Macina et al., 2025) prefers the LLM Tutor response over the ground truth response.
- **Desired Annotation Match Rate (DAMR)**: % of labels assigned to the tutor responses matching the desiderata described in (Maurya et al., 2025).

Following (Maurya et al., 2025), to calculate the DAMR scores, we have assigned labels *"Yes", "No", "To some extent"* to the tutor responses using a round table of LLMs as Judges (LaaJ) (see Appendix F for the prompt details). A response gets a high DAMR score if the assigned label matches the desired label for the given pedagogical dimension as defined in (Maurya et al., 2025). The authors also report a poor correlation coefficient with two LLMs as judges across eight pedagogical dimensions. In our work, we have considered four different LLMs as judges, namely, Llama-3-3-70b-instruct (Grattafiori et al., 2024), Deepseek-V3 (DeepSeek-AI et al., 2025), Mixtral-8X22b-Instruct-v0.1 (Jiang et al., 2024), Phi-4 (Abdin et al., 2024) and a fifth ensemble model through majority voting for the same pedagogical dimensions. To establish the reliability of the LaaJ models, we use the mentioned LLMs to generate the labels for each tutor utterance on the annotated dataset[3] (Maurya et al., 2025) and report the Spearman Correlation, Pearson Correlation and Accuracy for the same (Appendix Table 8). We find that the ensemble LaaJ model serves as the best evaluator with the highest correlations across four out of the eight dimensions and comparable results for the other dimensions. We use this model to report our DAMR scores in Table 1. We have used the current state-of-the-art tutoring models (Liu et al., 2024; Azerbayev et al., 2024; Modi et al., 2024) as baselines for comparison.

Through Table 1, we can observe that our feature enhanced models perform consistently better compared to their respective baseline versions (for prompt used, check Appendix D). In the smaller models like Qwen2.5-7B-Socratic-LM (Liu et al., 2024) and Llemma-7B (Azerbayev et al., 2024), we do observe some variations, especially with regard to the *Mistake Identification* and *Mistake Location* dimensions. However, with much larger models like LearnLM 2.0, we see that the feature enhanced version produces responses with more desirable pedagogical qualities in six out of the eight dimensions and is comparable for the rest of the dimensions. We found consistent results (Table 2) when we replicated this experiment with a more complex prompt (prompt is attached in Appendix E, intended to test the model's pedagogy instruction following skills). This indicates the importance of modeling student's emotions across all the peda-

[3] https://github.com/kaushal0494/UnifyingAITutorEvaluation/blob/main/MRBench/MRBench_V2.json

Table 3: Ablation study results with emotions and education theory features. Prompt 1 and Prompt 2 refer to the simple and complex prompts respectively. QSLM = Qwen2.5-7B-Socratic-LM; ET = Education Theory

Model	Prompt1	Prompt2
QSLM	0.28	0.33
QSLM + emotion	0.21	0.12
QSLM + emotion + ET	**0.37**	**0.35**

gogical dimensions in the learning models. We occasionally observe variability in results when using smaller models — particularly with more complex prompts. This suggests that smaller models may struggle or become overwhelmed when presented with an excessive amount of information.

3.1 Ablation Studies

Table 3 highlights the importance of student emotion features mapped to education theory in enhancing the tutor's pedagogical capabilities. Across both the simple and complex prompts, prompting the model with relevant pedagogical strategies based on the detected emotion of the student results in a steep performance gain (by 9 points and 2 points through simple and complex prompts respectively). For the ablation study, we have used the Qwen2.5-7B-Socratic-LM (Liu et al., 2024) model. Since it is a smaller model, providing only the emotional information misguides the model. However, when instructions are grounded in educational theory, for instance, directing that if a student exhibits boredom, the instructional response should aim to enhance the learner's motivation, the model demonstrates a markedly stronger performance.

In general, across all dimensions, the emotion feature seems to have resulted in a performance gain particularly in *humanlikeness*, *actionability*, *coherence* and *tutor tone*. Each of these dimensions is related to a human's empathetic side, again indicating the significance of modeling student emotions in the models. Since we only have an annotated textual dataset, the quantitative results reported in Table 1 only employed emotions extracted from textual conversation. We evaluate the multimodal system in real-time through a user study discussed in the following section (Section 4) using the best model (LearnLM+).

4 User Study

To evaluate the impact of emotion-aware adaptation on learning performance, we conducted a within-subject user study using *MathBuddy*, our AI-based math tutoring system. The study aimed to assess whether integrating predicted student emotions leads to improved learning outcomes. A total of 30 participants (aged 15–55), representing diverse genders, nationalities, linguistic backgrounds, and educational profiles, took part in the study. Further details are provided in Appendix J.

4.1 User Study Analysis

The collected user study comprises a diverse set of metadata and multimodal data: (1) learning outcomes, measured through multiple-choice questionnaires on select math topics (serving as a pre- and post-test); (2) interaction data, in the form of chat logs from both tutoring conditions (Emotion ON/OF modalities); (3) affective states, derived from emotion predictions based on textual and facial inputs; (4) learning experience, measured with a 15-item post-interaction questionnaire including engagement and perceived support, rated on a 5-point Likert scale; (5) overall satisfaction, captured through a final rating-scale survey; and (6) user-generated ground truth for the aggregated emotion, through participants' gold annotations and corrections of the Emotion ON conversations.

Empathy is a desired quality in a tutor across pedagogical dimensions. Figure 3 reports the mean score of 30 participants across 15 questions, each of them mapped to different pedagogical dimensions, the same discussed in Section 3 (Table 9 in Appendix). Here, we observe that the average satisfaction scores are generally higher when emotions are used to guide *MathBuddy*'s pedagogical strategies (Emotion ON condition). This suggests that participants reported a more positive perception of this interaction mode across the questionnaire dimensions, which we consider as indicators of their overall learning experience.

Empathetic responses enhances user learning. Comparing participants' facial emotional dynamics over time, those in the Emotion ON condition displayed a higher frequency of positive expressions than those in the Emotion OFF condition (see Figure 8 in the Appendix).

Looking at the kernel density estimates (KDE) for the facial emotion duration distributions across conditions (Figure 4), we can see that all three emotion classes (*Negative*, *Neutral*, *Positive*) exhibit highly skewed distributions, with most durations clustered under 2 seconds.

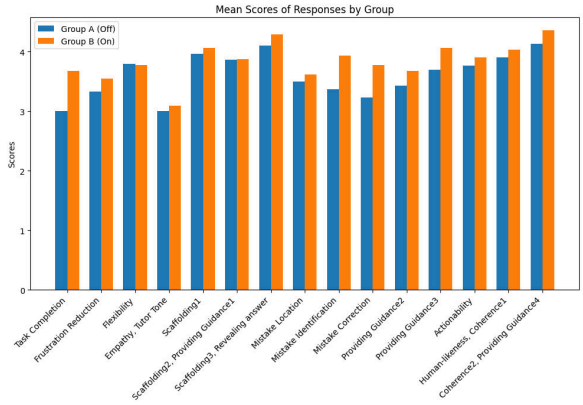


Figure 3: Average scores of participants with and without the emotion-aware condition

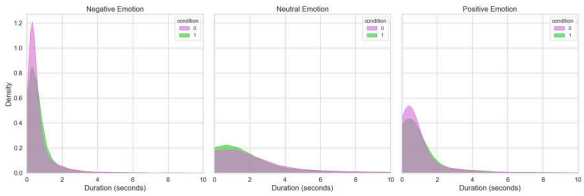


Figure 4: Kernel density estimates of facial emotion durations per class and condition.

We calculated the mean duration per participant for each of the three facial emotional states and conducted Wilcoxon paired t-tests among the conditions; the results are summarized in Table 4. We can appreciate that positive expressions showed a statistically significant difference ($p = .046$), with longer average durations under the Emotion ON condition (1.14s vs. 0.93s). This result indicates that participants experience longer positive emotional states under the Emotion ON condition.

These findings suggest that modeling student emotions and providing adaptive emotional feedback may have contributed to a more engaging and satisfying user experience during the tutoring sessions, potentially enhancing the overall learning process. Nonetheless, facial expressions commonly associated with happiness, such as smiling, can sometimes reflect more complex states, e.g., sarcasm or frustration. Expanding our multimodal channels could help capture emotional nuances more accurately, especially in technical domains such as mathematics, where affective signals are often subtle and context-dependent.

The detected emotions align with user emotions. As the last step of the experiment, the participants reviewed their own conversations in the Emotion ON condition, adjusting the tutor's emotion labels when necessary. To assess the system's ability to correctly detect the participant's emotions, we compare the aggregated text and facial emotions detected by the different modalities with the participant reviewed emotions. As reported in Table 5, the system achieves an overall accuracy of 60% in real-time usage. However, we observe a sharp fall in the recall scores for the *Neutral* class. This highlights the non-robust nature of our aggregation method where we try to suppress neutrality by preferring the modality that reports a non-neutral emotion. This can be improved by adopting a more sophisticated emotion aggregation method but remains to be explored as part of our future works.

Table 4: Wilcoxon paired test on mean emotion durations per user and emotion (in seconds).

Emotion	p-value	(Emo OFF) Mean ± Std	(Emo ON) Mean ± Std
Negative	0.537	0.72 ± 0.56	0.65 ± 0.42
Neutral	0.248	12.73 ± 30.41	7.07 ± 14.7
Positive	**0.046**	0.93 ± 0.87	1.14 ± 0.87

Table 5: Comparison of system aggregated multimodal emotions to participant gold annotations.

Emotion	Precision	Recall	F1-score
Negative	0.57	0.98	0.72
Neutral	0.71	0.15	0.25
Positive	0.79	0.41	0.54

5 Conclusion

In this paper, we introduced *MathBuddy*, an emotionally aware LLM-based math tutor that uses both text and facial expressions to model student emotions and deliver empathetic, pedagogically informed responses. By bridging affective cues with adaptive tutoring strategies, *MathBuddy* enhances student engagement and learning effectiveness by a massive 23 points using win rate (Macina et al., 2025) and 3 points at an overall level using DAMR scores as reported in Table 1. The usefulness of our approach is also supported through our carefully designed user studies as reported in Section 4.

Our evaluations highlight the benefits of integrating emotional awareness into LLM-driven education. We envision *MathBuddy* as a step toward more human-centered, emotionally intelligent learning systems.

Limitations

Generally, the development and evaluation of tutoring systems is limited by the lack of a golden standard or straightforward performance metrics. Qualitative analysis relies on feedback from users, who, however, may not be aware of a tutor system's ideal behavior—for example, instead of rating the educational prowess, they might rate user experience. Moreover, the user study described in this paper captured an age group devoid of children, whose idea of ideal tutoring may be vastly different and feature concepts our evaluation misses entirely. We also introduce a dataset with this work which currently contains coarse-grain emotion labels for student utterances. However, we feel that finer emotion labels capturing micro facial expressions may greatly benefit in capturing a much more nuanced student affective state. We leave this non-trivial task for future work. Based on the data and user feedback, it appears that the modalities currently used (text-based and face-based) may not be informative enough to accurately capture a user's emotional state. In technical education contexts, emotional expression in text is often subtle, unlike in other genres such as poetry (Hou and Frank, 2015). This insight motivates future investigation into additional modalities, e.g., spoken audio, handwritten notes, or biometric sensors, which ideally considerably increase emotion recognition accuracy. Further, spoken audio and handwritten notes more accurately represent the classical learning environment people might be more familiar with than a digital chat interface. We also want to address the aggregation strategy that we use for merging the emotion classes from different modalities in our system. The strategy can be further improved to handle neutral emotional states better. We leave this improvement for future work.

Acknowledgments

We would like to acknowledge the contributions of Yesica Yanina Duarte who helped us design the user study and the interface of the system. We also want to acknowledge Prof. Daniel Klotz who helped us with his deep insights and interesting critiques on this project. We extend a hearty vote of thanks to all the enthusiastic participants of the user study who are mostly members of the IT:U Interdisciplinary Transformation University Austria.

Ethical Considerations

This work acknowledges the ethical implications of our system's design and the data it collects. All conducted studies adhered to the GDPR (Goddard, 2017), upholding all included principles, e.g., purpose limitation, data minimization, confidentiality, as well as lawful, fair, and transparent processing.

Before data collection, informed consent was obtained, clearly outlining the purpose of the research, the voluntary nature of participation, and the right to withdraw at any time without consequence. All data were anonymized to protect participant identities. Special attention was given to exclusively collecting the information necessary to fulfill the functionality of real-time interaction. Although the system processes identifiable data such as image recordings of a participant's face, this is solely used for emotion recognition and never stored beyond model inference. The entire process followed the institutional ethical guidelines set by the IT:U Interdisciplinary Transformation University Austria.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Sara Ali, Bushra Naz, Sanam Narejo, Sahil Ali, and Jitander Kumar Pabani. 2024. Transformers Unveiled: A Comprehensive Evaluation of Emotion Detection in Text Transcription. In *2024 Global Conference on Wireless and Optical Technologies (GCWOT)*, pages 1–7. IEEE.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An Open Language Model for Mathematics. In *The Twelfth International Conference on Learning Representations*.

Zhendong Chu, Shen Wang, Jian Xie, Tinghui Zhu, Yibo Yan, Jinheng Ye, Aoxiao Zhong, Xuming Hu, Jing Liang, Philip S Yu, and 1 others. 2025. LLM agents for education: Advances and applications. *arXiv preprint arXiv:2503.11733*.

Tulika Chutia and Nomi Baruah. 2024. A review on emotion detection by using deep learning techniques. *Artificial Intelligence Review*, 57(8):203.

Richard J Daker, Sylvia U Gattas, Elizabeth A Necka, Adam E Green, and Ian M Lyons. 2023. Does anxiety explain why math-anxious people underperform in math? *npj Science of Learning*, 8(1):6.

Rajasree Das Chaiti and Mahzuzah Afrin. 2024. FERAC Dataset. https://www.kaggle.com/datasets/rajasreechaiti/ferac-dataset. Accessed: 2024-07-01.

Training Data. 2023. Facial Emotion Recognition Dataset. https://huggingface.co/datasets/username/dataset-name. Accessed: 2024-07-01.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. DeepSeek-V3 Technical Report. *Preprint*, arXiv:2412.19437.

Sidney K D'Mello and Art Graesser. 2012. Language and discourse are powerful signals of student emotions during tutoring. *IEEE Transactions on Learning Technologies*, 5(4):304–317.

Michelle Goddard. 2017. The EU General Data Protection Regulation (GDPR): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705.

Arthur C. Graesser, Mark W. Conley, and Andrew Olney. 2012. *Intelligent tutoring systems.*, pages 451–473. APA handbooks in psychology®. American Psychological Association, Washington, DC, US.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 Herd of Models. *Preprint*, arXiv:2407.21783.

Yufang Hou and Anette Frank. 2015. Analyzing Sentiment in Classical Chinese Poetry. In *Proceedings of the 9th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 15–24, Beijing, China. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of Experts. *Preprint*, arXiv:2401.04088.

Harsh Kumar, Ruiwei Xiao, Benjamin Lawson, Ilya Musabirov, Jiakai Shi, Xinyuan Wang, Huayin Luo, Joseph Jay Williams, Anna N. Rafferty, John Stamper, and Michael Liut. 2024. Supporting Self-Reflection at Scale with Large Language Models: Insights from Randomized Field Experiments in Classrooms. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, L@S '24, page 86–97, New York, NY, USA. Association for Computing Machinery.

Belle Li, Curtis J. Bonk, Chaoran Wang, and Xiaojing Kou. 2024. Reconceptualizing Self-Directed Learning in the Era of Generative AI: An Exploratory Analysis of Language Learning. *IEEE Transactions on Learning Technologies*, 17:1489–1503.

Jionghao Lin, Shaveen Singh, Lele Sha, Wei Tan, David Lang, Dragan Gašević, and Guanliang Chen. 2022. Is it a good move? Mining effective tutoring strategies from human–human tutorial dialogues. *Future Generation Computer Systems*, 127:194–207.

Jiayu Liu, Zhenya Huang, Tong Xiao, Jing Sha, Jinze Wu, Qi Liu, Shijin Wang, and Enhong Chen. 2024. SocraticLM: Exploring Socratic Personalized Teaching with Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pages 85693–85721. Curran Associates, Inc.

Jakub Macina, Nico Daheim, Sankalan Chowdhury, Tanmay Sinha, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2023. MathDial: A Dialogue Tutoring Dataset with Rich Pedagogical Properties Grounded in Math Reasoning Problems. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5602–5621, Singapore. Association for Computational Linguistics.

Jakub Macina, Nico Daheim, Ido Hakimi, Manu Kapur, Iryna Gurevych, and Mrinmaya Sachan. 2025. MathTutorBench: A Benchmark for Measuring Open-ended Pedagogical Capabilities of LLM Tutors. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China. Association for Computational Linguistics.

Fazliddin Makhmudov, Alpamis Kultimuratov, and Young-Im Cho. 2024. Enhancing Multimodal Emotion Recognition through Attention Mechanisms in BERT and CNN Architectures. *Applied Sciences*, 14(10):4199.

Kathrin E. Maki, Anne F. Zaslofsky, Robin Codding, and Breanne Woods. 2024. Math anxiety in elementary students: Examining the role of timing and task complexity. *Journal of School Psychology*, 106:101316.

Kaushal Kumar Maurya, Kv Aditya Srivatsa, Kseniia Petukhova, and Ekaterina Kochmar. 2025. Unifying AI Tutor Evaluation: An Evaluation Taxonomy for Pedagogical Ability Assessment of LLM-Powered AI Tutors. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1234–1251, Albuquerque, New Mexico. Association for Computational Linguistics.

Abhinit Modi, Aditya Srikanth Veerubhotla, Aliya Rysbek, Andrea Huber, Brett Wiltshire, Brian Veprek, Daniel Gillick, Daniel Kasenberg, Derek Ahmed, Irina Jurenka, James Cohan, Jennifer She, Julia Wilkowski, Kaiz Alarakyia, Kevin R. McKee, Lisa Wang, Markus Kunesch, Mike Schaekermann, Miruna Pîslar, and 26 others. 2024. LearnLM: Improving gemini for Learning.

Vincent Mühler. 2024. face-api.js: JavaScript API for Face Recognition and Face Detection. https://github.com/justadudewhohacks/face-api.js. Accessed: 2025-07-01.

Reinhard Pekrun. 2006. The control-value theory of achievement emotions: Assumptions, corollaries, and implications for educational research and practice. *Educational psychology review*, 18:315–341.

Reinhard Pekrun, Thomas Goetz, Wolfram Titz, and Raymond P Perry. 2002. Academic emotions in students' self-regulated learning and achievement: A program of qualitative and quantitative research. *Educational psychologist*, 37(2):91–105.

Jonathan Posner, James A Russell, and Bradley S Peterson. 2005. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology*, 17(3):715–734.

V Sanh. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Proceedings of Thirty-third Conference on Neural Information Processing Systems (NIPS2019)*.

Johanna Schoenherr, Stanislaw Schukajlow, and Reinhard Pekrun. 2025. Emotions in mathematics learning: a systematic review and meta-analysis. *ZDM–Mathematics Education*, pages 1–18.

Jing Tan, Jie Mao, Yizhang Jiang, and Ming Gao. 2021. The influence of academic emotions on learning effects: A systematic review. *International journal of environmental research and public health*, 18(18):9678.

Chaoran Wang, Zixi Li, and Curtis Bonk. 2024. Understanding self-directed learning in AI-Assisted writing: A mixed methods study of postsecondary learners. *Computers and Education: Artificial Intelligence*, 6:100247.

Juan Zheng, Susanne Lajoie, and Shan Li. 2023. Emotions in self-regulated learning: A critical literature review and meta-analysis. *Frontiers in psychology*, 14:1137010.

Appendix

A Trained BERT Evaluation

Table 6 depicts the evaluation results of all text-based emotion recognition models investigated.

Table 6: Text emotion classification model performance comparison based on accuracy and F1-score on silver-labeled test set.

Model	Accuracy	F1-Score
bert-base-uncased	0.611	0.573
distilbert-base-uncased	0.618	0.579
roberta-base	0.576	0.538
distilroberta-base	0.617	0.579
roberta-base-go_emotions	0.600	0.534

B Trained Face Recognition Models Evaluation

We trained two models using a custom Convolutional Neural Network (CNN) architecture with 3 convolutional layers, one baseline and one with an attention mechanism. In addition, we evaluated three pre-trained Vision Transformer (ViT) models: `google-vit-base-patch16-224` models.[4]

For fine-tuning, we considered two dataset configurations obtained by merging samples from three existing datasets. The first, *small*, includes only data from FERAC (Das Chaiti and Afrin, 2024), and FER (Data, 2023). The second, *large*, also incorporates additional samples from AffectNet YOLO dataset. We trained one version of each model on both dataset configurations and evaluated them on their corresponding test sets. Results shown in the Table 7.

Table 7: Comparison of model performance on validation and test sets

Model	Small Test Set		Large Test Set	
	Accuracy	F1-Score	Accuracy	F1-Score
CNN	0.646	0.487	0.614	0.540
CNN (+ Att)	0.667	0.503	0.610	0.540
ViT emo cls	0.794	0.740	0.793	0.787
ViT emo det	**0.809**	**0.756**	**0.797**	**0.793**
ViT face rec	0.799	0.747	0.789	0.783

C System Prompt Template

The exact prompt template employed to elicit the tutor model's response is as follows:

```
Task:
Below is an instruction that describes a task,
paired with an input that provides further
context. Write a response that appropriately
completes the request.
```

[4] https://huggingface.co/JamesJayamuni/emotion_classification_v1.2
https://huggingface.co/jayanta/google-vit-base-patch16-224-cartoon-emotion-detection
https://huggingface.co/jayanta/google-vit-base-patch16-224-cartoon-face-recognition

```
Instruction:
1. You are an experienced math teacher and you
are going to respond to a student in a useful
and caring way.
2. Gently nudge the student towards the correct
 answer using guiding questions as your
response.
3. Also consider the student's emotional state.
- Positive emotions include engagement and joy.
- Neutral emotions include neutral and surprise.

- Negative emotions include angriness, boredom,
 confusion, contempt, disgust, fear,
frustration, and sadness.
4. If the student's last response indicates:
- negative emotion, please motivate the student
 as a teacher.
- If the student's last response indicates
positive emotion or neutral emotion, please
challenge the student as a teacher.

Full Conversation:
{}

Sentiment based on Student's Facial Expression
and Text Input (out of Positive, Neutral,
Negative):
{}

Tutors Response:
{}
```

D Simple Prompt Template

This is an example of the simple prompt template that we tested our model with. This is an adapted version of (Macina et al., 2025)'s scaffolding generation prompt.

```
Task:
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:
1. You are an experienced math teacher and you
are going to respond to a student in a useful
and caring way.
2. Gently nudge the student towards the correct
 answer using guiding questions as response.
Also consider the student's emotional state.
3. If the student's last response indicates:
- boredom, please motivate the student as a
teacher
- engagement, please challenge the student as a
 teacher.
4. The student is trying to solve the following
 problem.

Full Conversation:
{}

Tutors Response:
{}
```

E Complex Prompt Template

This is an example of the complex prompt template that we also tested our model with. The complex prompt is more verbose containing specific pedagogical instructions to follow. This is an adapted version of (Macina et al., 2025)'s pedagogy instruction following prompt.

```
Task:
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:
1. Be a friendly, supportive tutor.
2.Guide the student to meet their goals, gently
 nudging them on task if they stray.
3. Ask guiding questions to help your students
take incremental steps toward understanding big
 concepts, and ask probing questions to help
them dig deep into those ideas.
4. Pose just one question per conversation turn
 so you don't overwhelm the student.
5. Also consider the student's emotional state.
 If the student's last response indicates
boredom, please motivate the student as a
teacher.
6. If the student's last response indicates
engagement, please challenge the student as a
teacher.
7. Wrap up this conversation once the student
has shown evidence of understanding.

Full Conversation:
{}

Tutors Response:
{}
```

F Evaluation Prompt Template

The exact prompt template is employed for the LLM as Judge evaluation is as follows:

```
Instructions:
1. Following is the solution to the given math
problem, the conversation history between the
student and the tutor as the student tries to
```

```
solve the given math problem, and the tutor's
response to the student's last utterance.
2. Evaluate the tutor's response on the defined
 paradigms.
3. Also provide your reasoning for the
evaluation.

Math Solution:
{}

Conversation History:
{}

Tutor Response:
{}

Response (Stick to the given format, output it
as a json only):

"Mistake identification": "Yes/No",
"Mistake location": "Yes/No",
"Revealing of the answer": "Yes/No",
"Providing guidance": "Yes/No"
"Actionability": "Yes/No"
"Coherence": "Yes/No",
"Tutor tone": "encouraging/neutral/offensive"
"Human-likeness": "Yes/No"
"Reasoning": "The tutor's response is evaluated
 as follows..."
```

G Emotion Mapping

The emotion mapping applied by the system can be expressed as follows:

$$
\begin{aligned}
m &: E \to P \\
E_{\text{positive}} &= \{\text{Engaged, Happy}\} \\
E_{\text{neutral}} &= \{\text{Neutral, Surprised}\} \\
E_{\text{negative}} &= \{\text{Angry, Boredom, Disgusted,} \\
&\qquad \text{Fearful, Sad}\} \\
m(e) &= \begin{cases} \text{Positive}, & \text{if } e \in E_{\text{positive}} \\ \text{Neutral}, & \text{if } e \in E_{\text{neutral}} \\ \text{Negative}, & \text{if } e \in E_{\text{negative}} \end{cases}
\end{aligned}
$$

where E denotes the set of all recognizable emotions across modalities, and P denotes the set of primitive categories: *Positive*, *Negative*, and *Neutral*. We labeled *surprise* as neutral due to dataset limitations: since it can indicate both positive and negative emotions and no contextual information was available, this choice ensured consistency in the educational strategy. E_{positive} and E_{negative} denote the the positive and negative emotions detected through both modalities respectively. The emotions detected through text include *Engagement, Neutral and Boredom* while facial expression emotions include *Happy, Sad, Surprised, Angry, Disgusted and Fearful*. Note that P is a subset of E used for simplified downstream processing.

H LLM as a Judge Evaluation

Please refer to Table 8.

I User Study Questions

Please refer to Table 9.

J User Study Design

A total of 30 participants (aged 15–55), representing diverse genders, nationalities, linguistic backgrounds, and educational profiles, took part in the study. Each participant was assigned a unique user ID to ensure anonymity. Demographic distribution is reported in Figure 5.

Each participant completed two tutoring sessions with *MathBuddy*, solving one geometry problem and one probability problem. In both sessions, the system captured facial expressions via webcam and analyzed textual responses to infer the student's emotional state. However, only in the **Emotion ON** condition were these emotional predictions actively used to guide the tutor's communication and instructional strategy. In the **Emotion OFF** condition, emotion detection was passive and did not influence the tutor's behavior. The two math problems and the order of conditions were randomized across participants to control for order and content effects. Participants were aware of the existence of two different system configurations but were blind to their order. Each session lasted a maximum of 10 minutes. Participants could use a digital whiteboard, chat freely with the AI tutor, and decide whether to explore the solution after solving the problem or end the session early. Before and after the two sessions, participants completed a 6-question multiple-choice test to assess baseline and post-interaction knowledge in geometry and probability. Each test was independently completed within a 5-minute time limit.

After each tutoring session, participants filled out a 15-item Likert-scale questionnaire evaluating their experience in terms of engagement, clarity, frustration, emotional alignment, and perceived helpfulness of the tutor. At the end of the experiment, they completed a final satisfaction survey (4 closed and 4 open-ended questions), where they

Table 8: LLM as Judge results

Dimensions	Llama3-70B	DeepseekV3	Mixtral-22B	Phi4	Ensemble
Tutor_Tone_spearman	0.2965	**0.5791**	0.2484	0.1808	0.4779
Tutor_Tone_pearson	0.297	**0.5789**	0.2485	0.1819	0.4779
Tutor_Tone_accuracy	0.5019	**0.7907**	0.6522	0.4006	0.7025
Humanlikeness_spearman	**0.3605**	0.2377	0.0717	0.3548	0.3506
Humanlikeness_pearson	0.4101	0.2703	0.0757	0.4494	**0.4264**
Humanlikeness_accuracy	0.8758	0.8832	0.582	0.8857	**0.887**
Mistake_Identification_spearman	**0.6218**	0.508	0.1351	0.3317	0.5914
Mistake_Identification_pearson	**0.6442**	0.5311	0.1396	0.3579	0.6147
Mistake_Identification_accuracy	**0.8571**	0.8193	0.5615	0.8112	0.8516
Mistake_Location_spearman	0.3199	0.3756	0.169	0.1981	**0.4019**
Mistake_Location_pearson	0.319	0.3997	0.1704	0.2169	**0.4268**
Mistake_Location_accuracy	0.5155	0.5988	0.5205	0.6373	**0.628**
Revealing_of_the_Answer_spearman	0.8195	0.7925	0.5543	0.7854	**0.821**
Revealing_of_the_Answer_pearson	0.8195	0.794	0.5582	0.7891	**0.8206**
Revealing_of_the_Answer_accuracy	0.9379	0.9373	0.8957	0.9385	**0.9516**
Providing_Guidance_spearman	0.3824	0.3674	0.1575	0.3923	**0.4302**
Providing_Guidance_pearson	0.4037	0.4278	0.1689	0.4384	**0.4858**
Providing_Guidance_accuracy	0.6031	0.5807	0.4547	0.6199	**0.6217**
Actionability_spearman	0.3122	0.307	0.2058	0.382	**0.3937**
Actionability_pearson	0.3081	0.3477	0.2097	0.4081	**0.4162**
Actionability_accuracy	0.6224	0.5894	0.5366	0.6354	**0.6503**
Coherence_spearman	**0.4696**	0.392	0.1058	0.3975	0.4165
Coherence_pearson	**0.5102**	0.4591	0.1215	0.4744	0.4739
Coherence_accuracy	**0.8453**	0.8354	0.5652	0.8354	0.8398

could share general feedback about the system, their experience, and any perceived limitations.

Additionally, participants were shown their chat transcript from the Emotion ON session and asked to highlight tutor responses they found particularly helpful—either in terms of mathematical support or emotional alignment. They were also invited to review and correct the system's predicted emotions, enabling us to gather feedback on the accuracy of emotion recognition.

The entire experiment lasted between 30 and 45 minutes. All participants were presented with the same math problems and knowledge test items, carefully selected based on a pre-study survey. In this preliminary phase, 25 respondents (aged 24–45, primarily from academic environments) identified which mathematical topics they had found most difficult during their education. While calculus-related topics (e.g., derivatives, integrals) were the most frequently cited, geometry and probability followed closely and were chosen for their conceptual richness and suitability for short, interactive sessions.

K Participant Details

Please refer to Figure 5.

L User Study Analysis Details

We examine the following hypotheses:

- **H1**: The Emotion ON condition will elicit higher levels of engagement and lower levels of boredom in students' *textual responses*, compared to the Emotion OFF condition.
- **H2**: The Emotion ON condition will be associated with a higher frequency of positive *facial expressions* (e.g., happy) and a lower frequency of negative expressions (e.g., sad, angry, fearful etc.) during the tutoring session.
- **H3**: The system's emotion predictions will match user reports with moderate to high agreement.

L.1 Emotion Dynamics in Textual Responses

We analyzed 460 student utterances (235 and 225 in the Emotion OFF and ON conditions, respectively), reviewed by the participants themselves for the veracity of the machine-assigned labels. Each utterance was labeled with an intensity score from 0 (not present) to 2 (strong) for each of three emotional categories: *engagement*, *boredom*, and *neutral*.

Table 9: Mapping of User Study Questions to relevant Pedagogy Dimensions and metrics.

Question	**Pedagogy Dimension /Metric**
I completed the math problem on time.	Task Completion
The tutor helped me feel less stuck or frustrated.	Frustration Reduction
I felt the tutor adapted to my needs.	Flexibility
The tutor seemed aware of how I was feeling.	Empathy, Tutor Tone
The tutor guide me toward the solution.	Scaffolding1
The tutor helped me think through the problem step by step.	Scaffolding2, Providing Guidance1
The tutor guided me without simply giving away the answer.	Scaffolding3, Revealing answer
The tutor helped me understand where I made mistakes.	Mistake Location
The tutor helped me identify what my mistake was.	Mistake Identification
The tutor helped me to correct my mistakes.	Mistake Correction
The tutor helped me understand math concepts better.	Providing Guidance2
I was able to connect the explanations to what I already knew.	Providing Guidance3
The tutor's feedback was useful and helped me know wat to do next.	Actionability
The tutor's responses were coherent with what I asked.	Human-likeness, Coherence
The explanations were easy to follow.	Coherence, Providing Guidance4

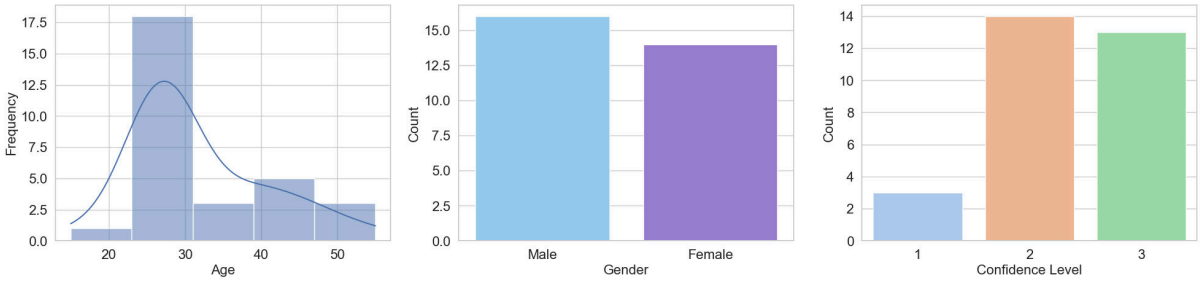


Figure 5: Dataset presentation

Further statistical tests are required to confirm the significance of these differences. Negative emotion is excluded from these as it is never detected by the system in either condition.

Figure 6 displays the temporal quantile plots for engagement and neutral predictions across the two conditions. For each emotion, we report the median intensity (solid line) and quantile bands (shaded areas) over interaction steps.

- Engagement. Both conditions — Emotion OFF and ON — show fluctuating median engagement levels around intensity 1 throughout the session. The Emotion ON condition (upper right) presents slightly more late-session variability, including peaks in the upper quantiles near the final third of the interaction. This

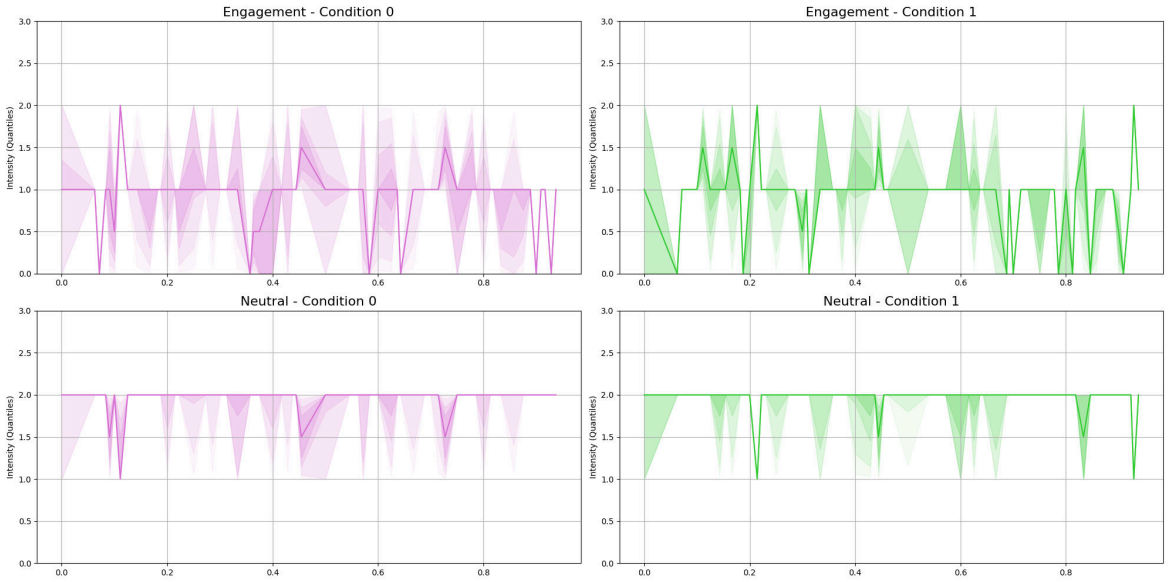


Figure 6: quantile trajectories of textual emotion intensities within interaction sessions.

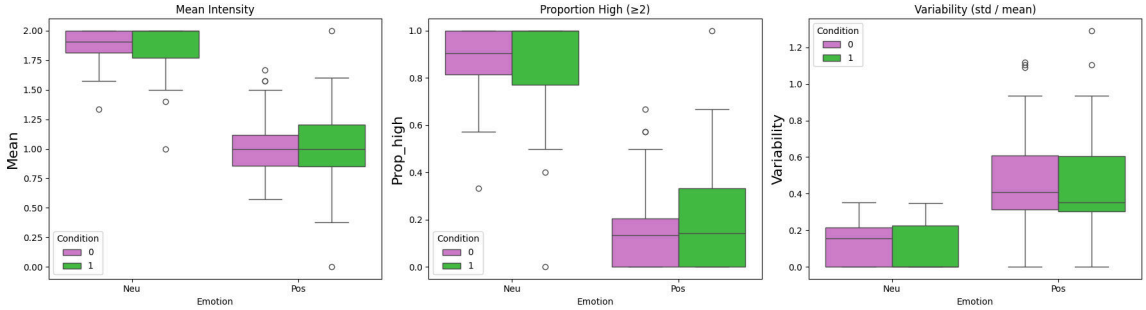


Figure 7: Text emotions statistical properties comparison by condition

may reflect individual differences in how students responded to emotionally adaptive tutoring strategies. The Emotion OFF condition (upper left) shows more symmetric variability distributed throughout the session, without any marked upward trend.

- Neutrality. In both conditions, neutrality maintains high median intensity across the whole session. The Emotion ON one shows slightly more fluctuation in lower quantiles during mid-session relative steps, but the overall pattern suggests a dominant neutral tone throughout, with low between-participant variation.

- Boredom. Predictions were consistently scored as 0 across all utterances and conditions, and are thus omitted from the plot.

To complement the trajectory analysis, we ran Wilcoxon signed-rank tests comparing the two conditions (Emotion OFF vs. ON) across three textual emotion metrics: *mean intensity*, *proportion of high-intensity values* (≥ 2), and *intra-session variability*. Analyses were conducted separately for *neutral* and *positive* emotional classes (Figure 7). Results revealed no statistical significant differences between conditions across any metric, suggesting that the Emotion ON condition did not produce measurable aggregate changes in emotional expression within student text responses. This suggests that textual input alone may not provide sufficiently informative signals to accurately detect participants' emotions, and that integrating additional multimodal inputs could represent a valuable resource for future improvements.

L.2 Face Emotion Details

We analyzed the temporal distribution of students' facial expressions changes during the tutoring sessions. Each emotion *positive*, *neutral*, and *negative*, is treated as a discrete categorical value, ignoring the confidence scores.

Given the session lengths and user-specific ex-

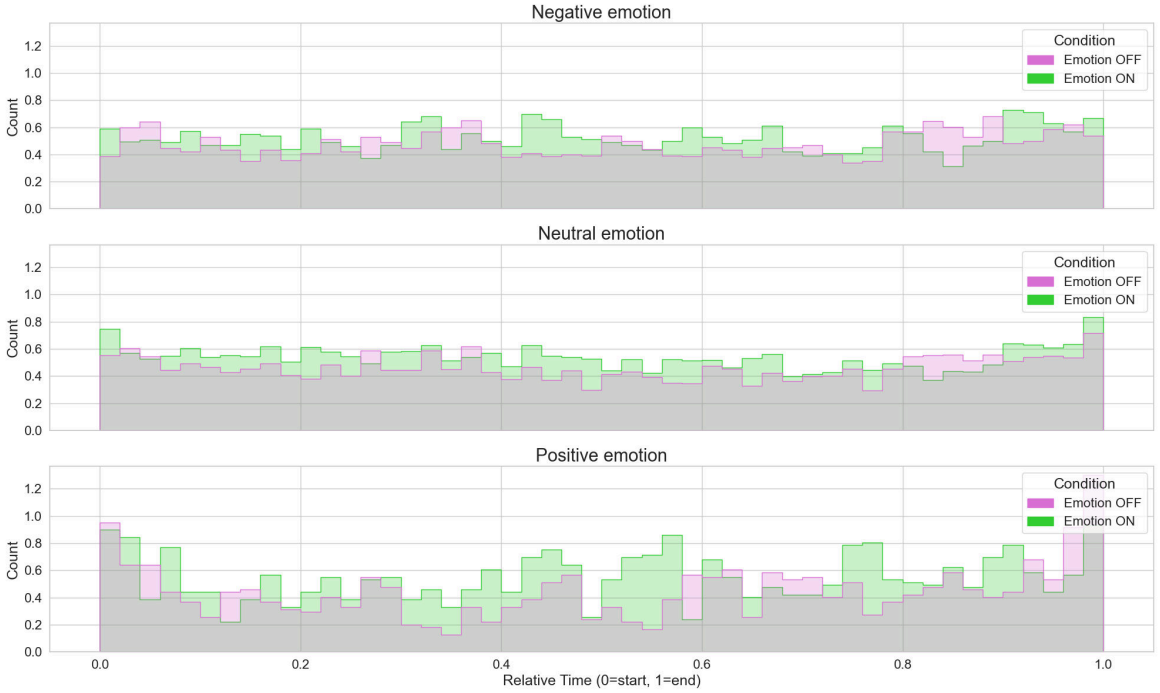


Figure 8: Histogram of facial emotion frequency over normalized session time. Each row shows one emotion category. Color-coded by condition (Emotion OFF = violet, Emotion ON = green).

pressivity, raw emotion timestamps were first converted into discrete time intervals $\Delta' t_{ij}^{(u)}$, defined as the elapsed time between two consecutive detections t_i and t_j within a session for participant user u. Each $\Delta' t_{ij}^{(u)}$ was assigned to the second timestamp in the pair t_j, reflecting the time at which the corresponding emotion was detected. All timestamps were cumulated and then normalized by the total session duration $\Delta t_{\max}^{(u)}$ for each users:

$$\Delta t_{ij,(\text{rel})}^{(u)} = \frac{\Delta t_{ij}^{(u)}}{\sum_{ij} \Delta t_{ij}^{(u)}} = \frac{\Delta t_{ij}^{(u)}}{\Delta t_{\max}^{(u)}},$$

where $\Delta t_{ij}^{(u)}$ is the cumulative time from the start of the session up to the j-th detection. The result is a temporally standardized view of facial emotion dynamics across all participants. This normalization enabled comparison of emotion frequencies across participants on a common temporal axis ranging from 0 (session start) to 1 (session end).

Figure 8 shows the relative density of detected emotions aggregated across participants and grouped into temporal bins, while Figure 9 show the resulting mean time series with shaded areas representing one standard deviation, separately for Emotion OFF (blue) and Emotion ON (orange) conditions. These curves capture the general trends in emotion distribution over time and the degree of agreement among participants. A wide standard deviation band indicates higher variability—i.e., some users expressed the emotion frequently in that bin, others not at all. Overall, both conditions show overlapping trends, with slight increases in positive emotions at the end of the session under Emotion ON. However, variability remains high across participants, especially for positive expressions, limiting the strength of conclusions.

To statistically test for distributional differences between conditions, we computed two shape-based metrics per user and emotion class:

- **Centroid**: the weighted temporal center of the emotion distribution;
- **Skewness**: the temporal distribution asymmetry (positive = delayed, negative = early).

Results from Wilcoxon signed-rank tests revealed no significant differences in centroid position or skewness between Emotion ON and OFF across all emotion classes ($p > 0.05$). Descriptively, the centroid for positive emotions was slightly earlier in Emotion ON (mean = 26.2) than in OFF (mean = 27.9), and skewness slightly lower (2.42 vs. 2.79), suggesting a minor temporal anticipation of positive expressions.

To investigate the structural patterns of emotional flow during the sessions, we analyzed the

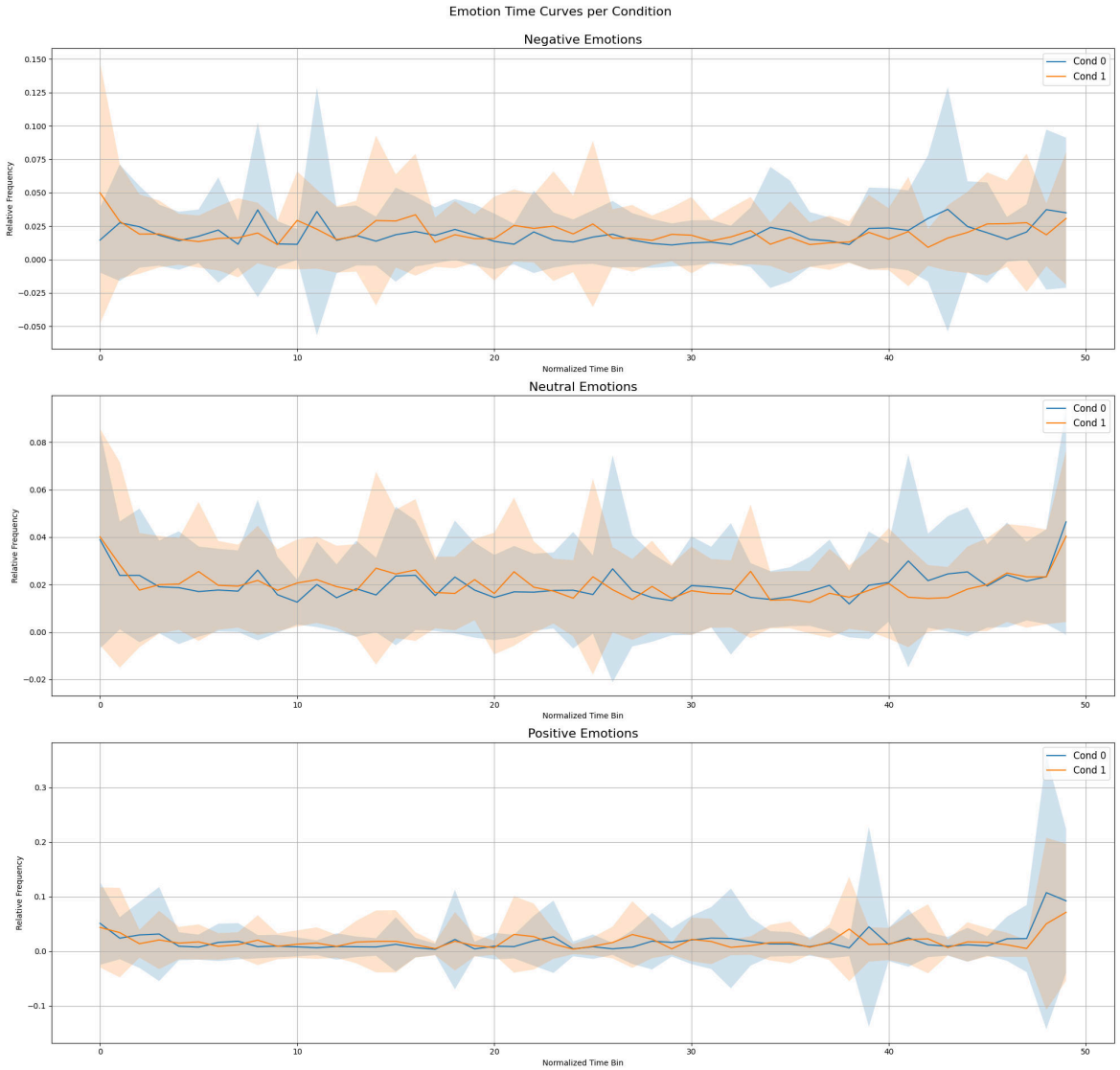


Figure 9: Average temporal curves of facial emotion frequency per participant, with standard deviation

transitions between facial emotion classes. Each session was modeled as a sequence of discrete emotional states sorted by cumulative timestamp. For each participant and condition, we computed a transition matrix capturing the frequencies of all possible emotion-to-emotion transitions (e.g., Pos→Neu). Transition counts were normalized by the total number of transitions per user, and then averaged across participants to obtain group-level transition probabilities. The resulting transition graphs are shown in Figure 10, separately for the Emotion OFF and ON conditions. Each node represents an emotional state, with node size indicating average state occupancy. Directed edges indicate transitions between states, with edge thickness and labels proportional to transition probability.

We further analyzed the average duration of each facial emotion under both conditions. For every emotion occurrence, duration was computed as the time elapsed until the next detection ($\Delta t = t_{i+1} - t_i$), and assigned to the first timestamp t_i, under the assumption that the emotion persisted until a new one was detected.

L.3 System's ability to detect emotions

To test last hypothesis, we evaluated the system's predictions against gold-standard annotations and computed standard classification metrics. As a final step, participants in the Emotion ON condition reviewed their own utterances at the end of the experiment and adjusted the tutor's emotion labels when necessary. This procedure provided a reliable basis for assessing the system's emotion recognition performance. We compared the system-generated emotion labels with the participant-corrected anno-

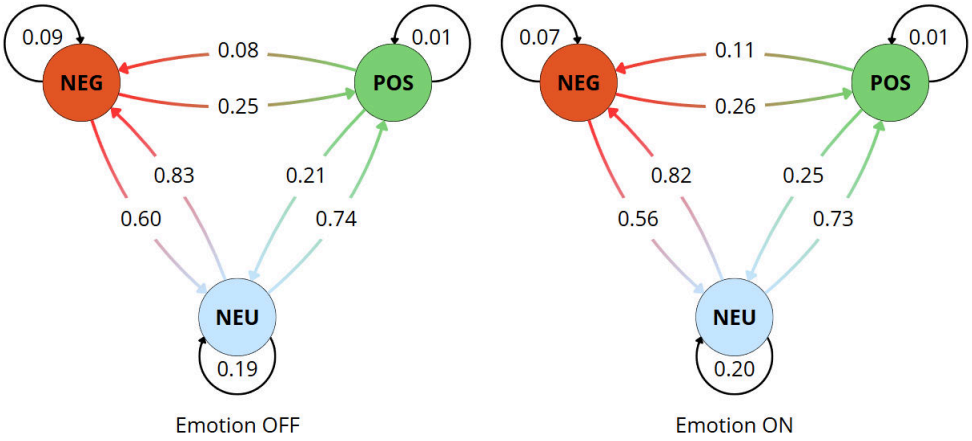


Figure 10: Transition graphs for facial expression emotion by condition.

tations and computed standard classification metrics. The system achieved an overall accuracy of 60%, with particularly high recall for the negative class (98%), but lower performance for neutral and positive classes, especially in terms of recall. This suggests that while the system is effective at detecting negative emotions, it struggles to consistently identify neutral or positive affect, which are more subtle or context-dependent. See table Table 5.

PLEDGETRACKER: A System for Monitoring the Fulfilment of Pledges

Yulong Chen, Zhenyun Deng, Andreas Vlachos
University of Cambridge
{yc632, zd302, av308}@cam.ac.uk

Michael Schlichtkrull
Queen Mary University London
m.schlichtkrull@qmul.ac.uk

David Corney, Nasim Asl, Joshua Salisbury, Andrew Dudfield
Full Fact
{firstname.lastname}@fullfact.org

Abstract

Existing methods simplify the pledge monitoring task into a document classification task, overlooking its dynamic temporal and multi-document nature. To address this issue, we introduce PLEDGETRACKER, a system that formulates pledge monitoring as structured event timeline construction. PLEDGETRACKER consists of three core components: (1) a multi-step evidence retrieval module; (2) a timeline construction module and; (3) a fulfilment filtering module, enabling us to capture the evolving nature of the task. We evaluate PLEDGETRACKER in collaboration with professional fact-checkers in real-world workflows, showing its superior effectiveness over Google search and `GPT-4o` with `web_search`.

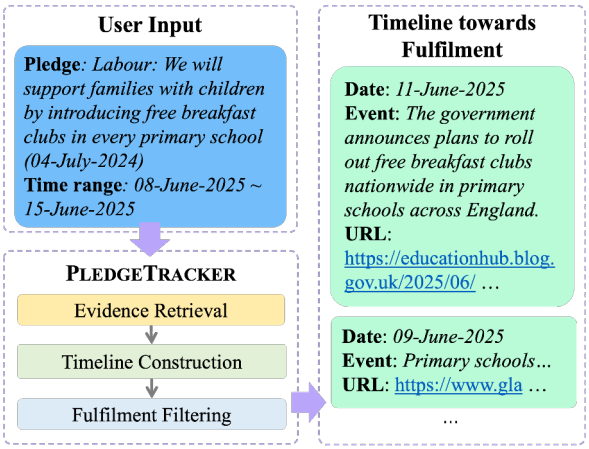


Figure 1: Overview of PLEDGETRACKER.

1 Introduction

Political pledges are commitments and governance plans made by political parties or candidates, especially during their election campaigns, which aim to promote their policies (Costello and Thomson, 2008; Dupont et al., 2019). Monitoring the fulfilment of pledges helps measure government performance, reinforcing transparency in democracy and accountability. However, this task typically requires fact-checkers to retrieve and analyse relevant documents regularly (e.g., daily or weekly) (Duval and Pétry, 2020; Fornaciari et al., 2021; Sahnan et al., 2025), which is resource-intensive, motivating the need for automated systems.

Recent work treats pledge monitoring as a *document-level* classification problem (Seki et al., 2024), by identifying whether a single article supports a pledge or not, overlooking the dynamic and long-term nature of pledge fulfilment. A political pledge is a strategic commitment, which is usually fulfilled via a sequence of actions and milestones (e.g., "*build 100 new schools in the UK by 2027*" materialises via local actions such as "*50 schools in England*" or incremental milestones like "*30 schools by 2025*"). Furthermore, the pledge status is temporal and dynamic in nature. It can evolve when new evidence emerges (e.g., the exit and re-entry into international agreements). Thus the task requires collecting and reasoning over temporally distributed evidence from multiple documents.

These requirements distinguish pledge monitoring from conventional fact-checking (Guo et al., 2022; Schlichtkrull et al., 2023; Iqbal et al., 2024). Although fact-checking also collects evidence from multi-document, it typically focuses on verifying whether a claim is supported by evidence *before* when the claim was made (Konstantinovskiy et al., 2021). Thus, the verdict is unlikely to change as those claims are about facts or knowledge that have already happened, except for corrections due to errors. In contrast, pledge monitoring aims to track how the fulfilment of a pledge evolves. Moreover, unlike the static labels of fact-checking output, pledge monitoring requires the output to reflect incremental progress over time. As such, the needs of end-users go beyond static labels, calling for structured, time-aware output.

To address these issues, we introduce PLEDGETRACKER, a retrieval augmented generation

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 839–850
November 4-9, 2025

(RAG)-based system for monitoring the fulfilment of political pledges by extracting timelines from online documents. As shown in Figure 1, PLEDGETRACKER consists of three core components in a multi-step framework: (1) an evidence retrieval module collects and identifies relevant documents through multi-step retrieval; (2) a timeline construction module identifies and extracts key event descriptions and their timestamps from multiple relevant documents; (3) a fulfilment filtering module determines relevant events, and assembles them into a temporally-structured timeline (Hu et al., 2024). For the development of the latter, we construct an annotated dataset covering 1,559 event descriptions across 50 pledges, where each event is labelled regarding its relevance to fulfilment.

To demonstrate the effectiveness, we evaluate PLEDGETRACKER in collaboration with professional fact-checkers from Full Fact, in their real-life workflows where evidence continuously evolves. Our system achieves 0.641 F_1 in identifying fulfilment events in a real-world evaluation. Moreover, our further analysis finds PLEDGETRACKER to be more accurate in retrieving useful evidence URLs (0.78 F_1) than Google Search (0.23 F_1) and `GPT-4o` with `web_search` (0.03 F_1), both of which are part of the modules in PLEDGETRACKER. Qualitative feedback suggests that PLEDGETRACKER brings useful events to the attention of the fact-checkers that would have otherwise been missed. We publicly release PLEDGETRACKER[1] and our annotation to facilitate the task of pledge monitoring.

2 Pledge Monitoring

Drawing inspiration from fact–checking organisations like Full Fact's Government Tracker[2], pledge monitoring refers to the task of fulfilling promises with actions, i.e., when, how, and to what extent those promises are being fulfilled. We formulate this task as constructing an event timeline that reflects the progress regarding a pledge.

Formally, given a pledge $p = (p_s, p_d, p_g, p_c)$, where p_s is the pledge speaker (e.g., a political party such as *Labour*), p_d is the pledge date (i.e., when it is made), p_g is the geographic scope (e.g., *the UK*), and p_c is the pledge claim (e.g., "*We will ban trail hunting*"), and a monitoring time range $r = (r_s, r_e)$, where r_s and r_e are the start date and end date, respectively, the system $\mathcal{S}$ is asked to generate a timeline T:

$$T = \mathcal{S}(p, r), \tag{1}$$

where T is the timeline (possibly empty if no progress has been made). For a non-empty $T = \{(e, t, url)\}$, each event description e_i is associated with a timestamp t_i and its source URL url_i, with the full set sorted in order, i.e., for all $i < j$, we have either $t_i \leq t_j$ (chronological) or $t_i \geq t_j$ (reverse chronological). Timeline T captures incremental progress and setbacks over time.

3 PLEDGETRACKER

As shown in Figure 2, PLEDGETRACKER is a RAG-based system consisting of three modules: an evidence retrieval module $\mathcal{R}$, a timeline construction module $\mathcal{T}$, and a fulfilment filtering module $\mathcal{F}$, i.e., $\mathcal{S} = \{\mathcal{R}, \mathcal{T}, \mathcal{F}\}$. Given a pledge and the time range, we first collect a set of documents using the evidence retrieval module: $D = \mathcal{R}(p, r)$. Then, based on the retrieved documents and the pledge, the timeline construction module extracts all possible events and their timestamp: $E = \mathcal{T}(D, p)$. Finally, the fulfilment filtering module selects the subset of events most useful to monitor the pledge, producing the final timeline: $T = \mathcal{F}(E, p, r)$. The subsequent subsections provide a detailed description of the corresponding modules.

3.1 Evidence retrieval

Following recent work on evidence retrieval that uses a multi-round retrieval strategy (Liao et al., 2023; Yang et al., 2024; Liu et al., 2024), PLEDGETRACKER's retrieval component is progressively expands and refines the document set D in multiple rounds of interaction and question-guided augmentation.

Given a pledge p and a target monitoring time range r, we first perform an initial web search using Google custom search API.[3] In particular, we construct a query string such as *"Labour: We will ban trail hunting (04-Jul-2024)"*, conditioned by the geographic scope p_g and the date range (r_s, r_e). As these results can often be sparse or incomplete, we further extract key noun phrases (e.g., "*trail hunting*") from the pledge content p_c using `spaCy`[4] as additional search queries. Given the retrieved

[1] https://huggingface.co/spaces/PledgeTracker/Pledge_Tracker

[2] https://fullfact.org/government-tracker/

[3] https://developers.google.com/custom-search/

[4] https://spacy.io/

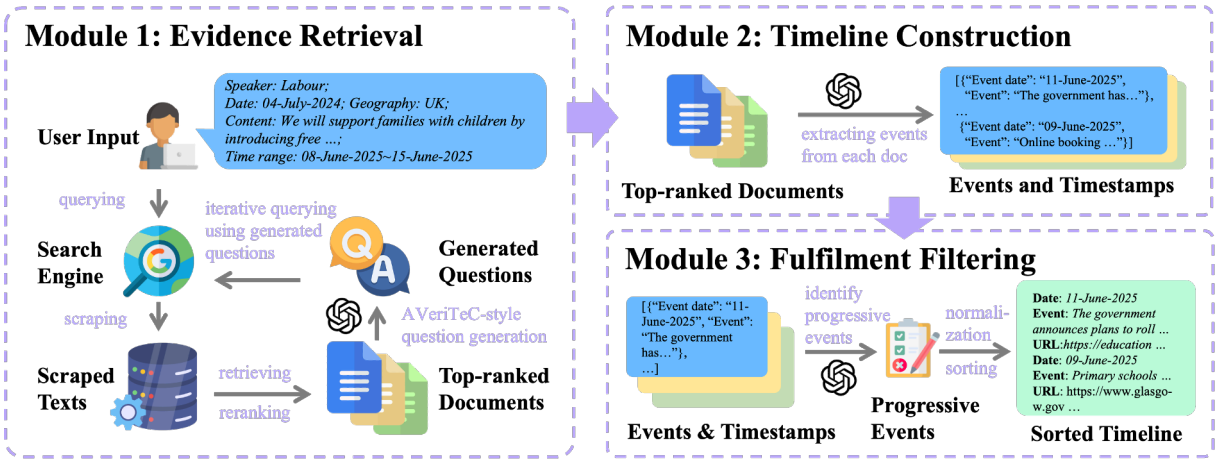


Figure 2: The architecture and workflow of PLEDGETRACKER.

URL results, we obtain the corresponding textual documents using `trafilatura` (Barbaresi, 2021), a library for web crawling and text extraction.

To guide deeper retrieval, we further incorporate question-driven augmentation based on retrieved evidence. Following Yoon et al. (2024), we first generate a set of hypothetical documents, which simulate possible evidence. We then use those hypothetical documents to retrieve sentence-level evidence from the scraped texts using `bm25`, and re-rank the evidence based on their semantic similarity computed with `SFR-Embedding-2_R`.[5] For each top-ranked evidence, we generate the corresponding clarification question that explicitly targets different aspects of the pledge (e.g., *"Is Labour planning to implement a central reporting mechanism for reporting potential animal welfare offences?"*). These questions are then used as new search queries for the next round of retrieval. Both the hypothetical document generation and question generation are performed by `Llama-3.1-8B-instruct` (Grattafiori et al., 2024) using in-context learning (ICL) examples from AVeriTeC (Schlichtkrull et al., 2023). The details can be found in Appendix A.1.

Finally, after multiple rounds of retrieval,the evidence retrieval module returns a set of top-ranked evidence. We then collect and deduplicate the document texts and corresponding URLs to construct the final D for the timeline construction module as described in the next subsection.

3.2 Timeline Construction

Rather than relying on predefined schemas (Minard et al., 2015), we adopt a generative extraction approach using `GPT-4o` (Hurst et al., 2024), which allows for more flexible identification of events (Gao et al., 2023; Chen et al., 2024a; Qorib et al., 2025).

In particular, we prompt the model using few-shot ICL examples consisting of document–event pairs that we annotated manually, and constrain the model output to follow the JSON format. Given a pledge p and each document $d_i \in D$, we construct a prompt that includes the document's metadata (e.g., title and publication date), the article body, and the pledge text, in order to generate relevant event descriptions (e.g., *"A petition is rejected because there is already a similar petition about banning trail hunting."*). Moreover, since event timestamps mentioned in the text may be expressed in various terms (e.g., publication date: `08-Jul-2024`, event temporal-related phrase: "`two days ago`"), we prompt the model through ICL to generate the corresponding absolute date (e.g., `06-Jul-2024`) if possible, or a relative date (e.g., `Last month (relative to 01-Jul-2024)`). The details can be found in Appendix A.2.

After processing all documents from D, we further sort the events by their dates. We normalise the timestamps using a rule-based parser that handles a wide range of temporal expressions (e.g., locating "`Autumn 2023`" into "`01-09-2023`"). Finally, this module returns a set of candidate events E.

3.3 Fulfilment Filtering

In practice, we find that not all events in E are informative or relevant to monitoring the fulfilment of the pledge. Although they are extracted from top-ranked documents, many events provide only contextual or background information (e.g., *What does the pledge mean?*), rather than concrete progress

[5] https://huggingface.co/Salesforce/

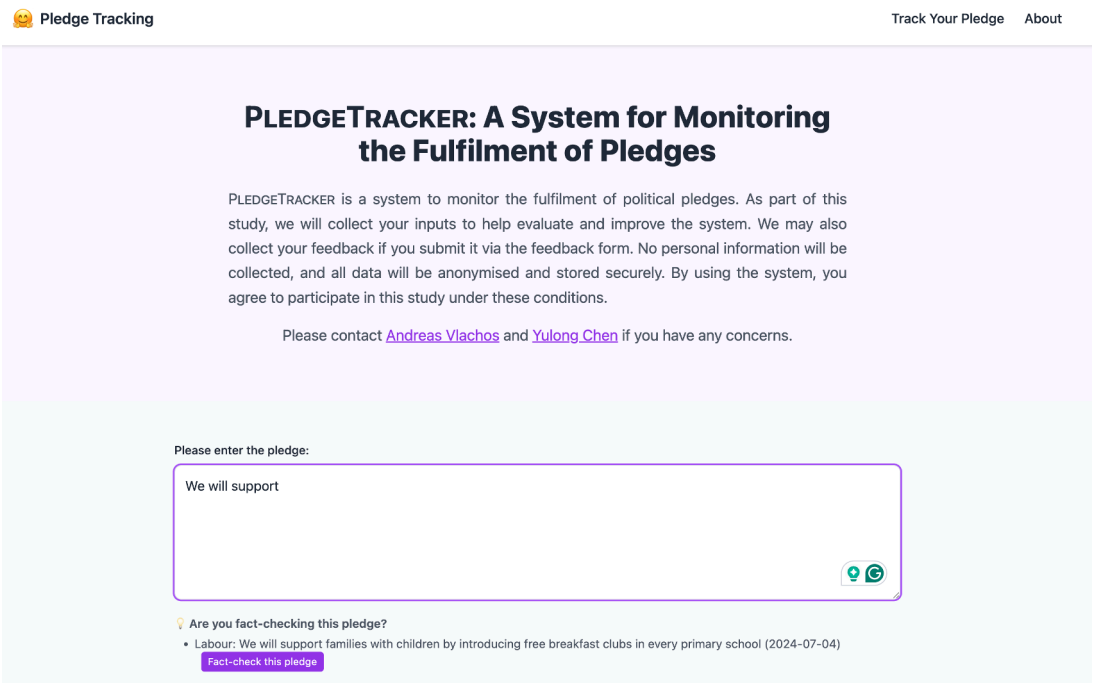


Figure 3: The user input interface of PLEDGETRACKER.

to fulfilling the pledge (*What progress has been made?*).[6] For example, while the event "*Critics claim trail hunting is being used as a 'smokescreen' for illegal fox hunting activities*" is related to "*trail hunting*", it does not contain any useful information about the actions that were taken. To address this, we developed the fulfilment filtering module $\mathcal{F}$ to filter the events to be included in the timeline.

To support the fulfilment filtering, we construct a dataset focusing on the task. We begin with a set of 50 pledges selected from FullFact government tracker, which are from the Labour Party's manifesto for the 2024 UK general election. We then use the PLEDGETRACKER (without fulfilment filtering) to retrieve all potentially related events from the time each pledge was made (starting on 4 July 2024) up to the time when the timeline was generated (March 2025). For each pledge, a professional fact-checker, who was familiar with it, examined the generated timeline and evaluated whether each event and its timestamp were useful or not, with the help of the corresponding URL. In particular, we define an event and its timestamp as *useful* in assessing fulfilment if it (1) is factually consistent with the source document, (2) contains a correctly inferred timestamp, and (3) contributes to the fulfilment of the pledge. If any of these criteria are not met, the event is labelled as *not useful*. In total, we collect 1,559 annotated instances, where each instance consists of a pledge, an event description, a timestamp, the original URL, and a binary usefulness label. In particular, our analysis shows that only 26.63% of them are useful in monitoring the fulfilment of the corresponding claims, which demonstrates the necessity of fulfilment filtering.

During testing, given each e_i from E, we ask `GPT-4o` to label each extracted event as either *useful* or *not useful* in assessing fulfilment using ICL examples from our annotation. The resulting timeline provides a clear and interpretable progression of pledge fulfilment over time. The details can be found in Appendix A.3.

4 User Interface Design

We build the PLEDGETRACKER demo system on Hugging Face Space (Nvidia A100) using Flask. Using the interface, users enter a pledge, specify the speaker, pledge date, and time range, and initiate the system by clicking the "*Let's track!*" button (Figure 3).

Once the input data is submitted, PLEDGETRACKER starts the multi-stage pipeline as detailed in §3. The system will start collecting evidence, generating the timeline, and identifying fulfilment events using ICL instances from our annotation, showing relevant status updates (Figure 4).

[6] https://fullfact.org/government-tracker/hillsborough-law-candour-duty/

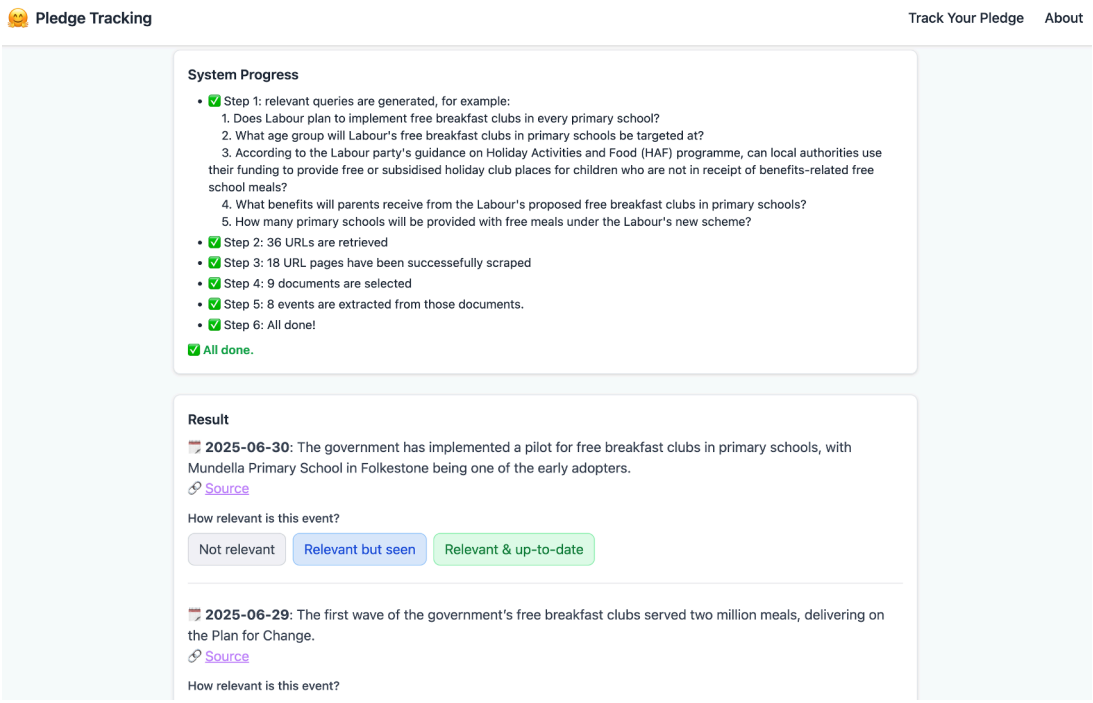


Figure 4: The output timeline interface of PLEDGETRACKER.

Finally, PLEDGETRACKER presents the timeline, where each event is associated with an event date, an event description, and the original source link. To support iterative refinement for analysis and future work, the demo system enables users to provide feedback on the usefulness of each event.

PLEDGETRACKER also supports matching pledges against previously checked ones. When the user enters a new pledge, the system automatically searches for similar pledges among the pledges already checked by the system, using `TF-IDF` and shows the top suggestions based on their similarities. For suggested pledges, the system will re-use previously retrieved results (from an initial web search) to accelerate the process and enable more accurate fulfilment filtering by selecting corresponding annotated data.

5 Experiments

We perform two kinds of quantitative evaluation: offline, using our annotated data, and in real-world use with professional fact-checkers. In particular, we first demonstrate offline the effectiveness of fulfilment filtering (§5.1), and then evaluate the full PLEDGETRACKER in real-world use (§5.2) and show its comparison with existing tools (§5.3). We further present qualitative analysis in §5.4.

	Train	Dev	Test
useful (%)	20.86	33.33	37.12
non-useful (%)	79.14	66.67	62.88
event/pledge	43.14	24.90	20.06

Table 1: Statistics for the fulfilment filtering annotation.

	P	R	F_1
ROBERTA	0.517	0.224	0.313
`Llama`	0.544	0.507	0.525
`GPT-4o`	0.509	0.836	0.633

Table 2: Results on fulfilment filtering.

5.1 Effectiveness of Fulfilment Filtering

As described in §3.3, we collect 1,559 instances for fulfilment filtering, which are divided into training (949), development (249) and test (361) sets based on pledges. Table 1 shows their statistics. We note the distribution difference across data splits due to fulfilment varying across pledges. We conduct experiments using three models: (1) ROBERTA-large (Liu et al., 2019) with full-parameter fine-tuning; (2) `Llama-3-8B` (Grattafiori et al., 2024) trained using instruction-based LoRA tuning (Hu et al., 2022) and; (3) `GPT-4o` with ICL prompting. Given a pledge and an associated event, each model is asked to assign a binary label indicating whether the event is useful in assessing fulfilment.

System	Pledge-level			URL-level			Novelty
	P	R	F_1	P	R	F_1	
PLEDGETRACKER	0.83	0.74	0.76	0.93	0.68	0.78	36
Google Search	0.32	0.08	0.12	0.50	0.15	0.23	5
`GPT-4o with web_search`	0.08	0.01	0.01	1.00	0.02	0.03	1

Table 3: Overall retrieval performance. Pledge-level: results first averaged per pledge, then averaged. URL-level: results averaged across all URLs. Novelty: the number of unique useful URLs retrieved by a system.

As shown in Table 2, `GPT-4o` achieves the best performance, with an F_1 score of 0.633. It suggests that, compared with ROBERTA and `Llama`, `GPT-4o` is better at capturing potential fulfilment signal. The main challenge lies in the imbalanced data distribution of the pledge monitoring data. As mentioned, the fulfilment events can be sparse in the real world, while most events lack concrete evidence of progress (c.f. Table 1).

5.2 Evaluation in Real-world Use

After deploying the full version of PLEDGETRACKER, we evaluate the system in a *real-world* setting with Full Fact fact-checkers. In particular, our evaluation was conducted from 12 June to 08 September 2025, monitoring 68 pledges from the Labour Party's 2024 UK election manifesto. Each timeline is generated over a time range of the past 7 days. As some pledges were monitored multiple times at different times in the evaluation period, we collected 113 timelines in total. Two professional fact-checkers (paper co-authors Nasim Asl and Joshua Salisbury), who were responsible for the corresponding pledges in their daily work, evaluate the usefulness of each event, using the criteria described in §3.3. We continue to present *all* candidate events, including both those retained and those filtered out, to the fact-checkers. This setup enables a direct comparison between the PLEDGETRACKER's filtering decisions and human judgments. During the evaluation, the fact-checkers select one of three labels: `not_relevant`, `relevant_seen`, and `relevant_update`. The label `relevant_update` indicates that the event is new to the fact-checkers and useful for fulfilment tracking, `relevant_seen` means that the event is useful and temporally appropriate, and meanwhile, fact-checkers already know about it. We therefore treat both `relevant_seen` and `relevant_update` as *useful* in our evaluation, since our goal is to assess whether the system can accurately surface relevant fulfilment evidence, regardless of whether the annotator had seen it from other sources. In total, 513 events were evaluated across 68 timelines.

Generally, PLEDGETRACKER achieves 0.764 precision, 0.553 recall and 0.641 F_1, demonstrating that it can identify fulfilment events with reasonably high performance in a real-world setting. Compared to the offline results in §5.1, the full system shows higher precision. This can be partly because the full system benefits from using the full annotation set for ICL prompting. Meanwhile, recall slightly decreases, which can be because the time range (past 7 days) is narrower, resulting in sparser fulfilment. In particular, for 513 events, we manually identify 152 fulfilment events (29.63%), which is lower than in the offline evaluation (37.12%).

5.3 Comparison with Existing Tools

We compare PLEDGETRACKER with two other tools that are often used for pledge monitoring: (1) Google Search and; (2) `GPT-4o` with `web_search`. In particular, we collect 13 pledge monitoring requests (from 12 June to 22 June 2025) from the evaluation in §5.2 that received at least one fulfilment event according to the fact-checker's judgment. We use the aforementioned two tools to return top-ranked evidence (Appendix C), and ask fact-checkers to evaluate them. Since they cannot directly return timelines, the evaluation focuses on *whether the retrieved URLs include events useful in assessing fulfilment*. For each pledge monitoring request, we first pool all URLs returned by the three systems, remove duplicates, and have professional fact-checkers label each URL. We take all *useful* URLs for a given request as the ground truth set and evaluate their performance as shown in Table 3.

Overall, PLEDGETRACKER retrieves 68% of all manually identified evidence with 0.93 precision and 0.78 F_1 at the URL level. It also contributes 36 unique, useful URLs that other systems fail to find. Compared to Google Search (0.15 recall), PLEDGETRACKER benefits from the question-driven iterative retrieval using question generation, which aligns with findings from the

ID	Pledge claim	Date	Event description, timestamp and URL
1	Labour will end the VAT exemption and business rates relief for private schools	2025-06-13	Private school families lost their High Court challenge against the Government over the VAT policy on fees. 2025-06-13. [URL]
2	Labour will capitalise Great British Energy with £8.3 billion, over the next parliament	2025-06-11	The government is delivering a new generation of publicly owned clean power. Great British Energy and Great British Energy–Nuclear will together invest more than £8.3 billion over the SR in homegrown clean power. 2025-06-11. [URL]

Table 4: Events that led to updates in Full Fact's pledge pages. The Date here refers to when the monitoring was requested. The time range is set to the past 7 days. We attach the hyperlink (URL) for reference.

AVeriTeC (Schlichtkrull et al., 2023, 2024). It is worth noting that PLEDGETRACKER has higher precision than Google Search (0.50), indicating the effectiveness of our other modules. Moreover, `GPT-4o` shows very poor performance in this task (0.03 F_1 at URL level). In our evaluation, we find that `GPT-4o` is less sensitive to temporal constraints. In particular, although `GPT-4o` returns 61 URLs in total, only 1 is within the correct time range.

5.4 Qualitative Feedback

The two Full Fact fact-checkers who conducted the human evaluation in §5.2 also provided some qualitative feedback. From their feedback and specific examples as shown in Table 4, we observe certain scenarios where the system has been helpful.

First, PLEDGETRACKER captures useful events that may otherwise be overlooked. In Table 4 case 1, it alerted fact-checkers to news that had not gained much coverage in the media, a High Court ruling. Although this event did not change the verdict of the pledge, it led to an update to the pledge page on the removal of the VAT exemption for private schools, as the page previously said the appeal was taking place. Second, PLEDGETRACKER assists in timely event identification. In Table 4 case 2, PLEDGETRACKER found that the investment would be split between Great British Energy and Great British Nuclear, on the same day the government's 2025 Spending Review was released. This early signal enabled them to update the pledge page promptly and contact the UK government for further clarification. Third, PLEDGETRACKER helps surface legislative and political signals that inform future developments. Fact-checkers found PLEDGETRACKER could highlight the names of bills and draft legislations associated with pledges, and trace their mentions across time in official communications. For example, it surfaces passing remarks by politicians, indicating when legislation or announcements could be expected, which was not previously captured through routine monitoring. Overall, they reported that PLEDGETRACKER greatly contributes to their workflow.

In addition to these strengths, fact-checkers also noted occasional hallucinations in the event descriptions, for example, the generated events can be inconsistent with the source documents. To mitigate this known limitation of LLMs (Zhang et al., 2023; Chen et al., 2024b), PLEDGETRACKER is designed to explicitly include source URLs for each event, allowing fact-checkers to verify the underlying evidence when necessary.

6 Conclusion and Future Work

We presented PLEDGETRACKER, the first end-to-end system that formulates pledge monitoring as the construction of temporally ordered timelines. By iteratively collecting evidence from online, with generative timeline construction and fulfilment filtering, PLEDGETRACKER captures incremental evidence and generates more interpretable outputs. We integrated the system into professional fact-checkers' real-life workflows, and found PLEDGETRACKER achieved an F_1 of 0.641 in identifying fulfilment events. Our further comparison with Google Search and `GPT-4o` with `web_search`, demonstrating the superior performance of PLEDGETRACKER for pledge monitoring.

Limitations

The limitations of PLEDGETRACKER can be stated from four perspectives. First, PLEDGETRACKER is built on the basis that pledges have already been identified and normalised, and therefore it does not address the task of automatically extracting and decontextualising pledges from manifestos (Deng et al., 2024; Panchendrarajan and Zubiaga, 2024). Second, our evaluation focuses on pledges from UK political parties. However, its effectiveness in

other linguistic or institutional contexts remains to be further explored (Zhang et al., 2024; Turk et al., 2025). Third, our evidence retrieval relies heavily on the Google Custom Search API, limiting its evidence coverage with potential ranking bias, and quota constraints. Fourth, due to the limited resources, we could not perform large-scale training and thus use small models and LLM APIs for implementing PLEDGETRACKER.

Ethical Considerations

Our work involves human annotation and evaluation as stated in §3.3, §5.2, and §5.3. These two annotators are professional fact-checkers and the co-authors of this paper. Their background information is provided in Appendix B.

We acknowledge that LLMs exhibit political biases (Chalkidis and Brandl, 2024); however, we mitigate these by using RAG (Lewis et al., 2020; Ram et al., 2023) and providing the URLs of the sources used for the timeline construction, so that users can verify the output themselves. Furthermore, we evaluated the system with fact-checkers from Full Fact, which is a signatory to the International Fact-Checkers Network code of principles (https://ifcncodeofprinciples.poynter.org/the-commitments) that stipulates that they need to be impartial in their work.

The release of our demo has been approved by the Ethics Review Committee[7] at the Department of Computer Science and Technology, University of Cambridge, under the CC-BY-NC license.

Acknowledgements

We appreciate the program chairs, area chair, and reviewers from EMNLP 2025 System Demonstration for their insightful feedback. We would like to thank Chenxi Whitehouse and Tom Stafford for their contributions to an earlier version of this project. This work is supported by the ERC grant AVERITEC (GA 865958). Full Fact's work is funded by the JournalismAI Innovation Challenge, supported by the Google News Initiative. Michael is further supported by the Engineering and Physical Sciences Research Council (grant number EP/Y009800/1), through funding from Responsible AI UK (KP0016).

[7] https://www.cst.cam.ac.uk/local/policy/ethics

References

Adrien Barbaresi. 2021. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. 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 122–131, Online. Association for Computational Linguistics.

Ilias Chalkidis and Stephanie Brandl. 2024. Llama meets EU: Investigating the European political spectrum through the lens of LLMs. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 481–498, Mexico City, Mexico. Association for Computational Linguistics.

Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. 2024a. Is a large language model a good annotator for event extraction? In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17772–17780. AAAI Press.

Yulong Chen, Yang Liu, Jianhao Yan, Xuefeng Bai, Ming Zhong, Yinghao Yang, Ziyi Yang, Chenguang Zhu, and Yue Zhang. 2024b. See what LLMs cannot answer: A self-challenge framework for uncovering LLM weaknesses. In *First Conference on Language Modeling*.

Rory Costello and Robert Thomson. 2008. Election pledges and their enactment in coalition governments: A comparative analysis of ireland. *Journal of Elections, Public Opinion and Parties*, 18(3):239–256.

Zhenyun Deng, Michael Schlichtkrull, and Andreas Vlachos. 2024. Document-level claim extraction and decontextualisation for fact-checking. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11943–11954, Bangkok, Thailand. Association for Computational Linguistics.

Julia C Dupont, Evelyn Bytzek, Melanie C Steffens, and Frank M Schneider. 2019. Which kind of political campaign messages do people perceive as election pledges? *Electoral Studies*, 57:121–130.

Dominic Duval and François Pétry. 2020. Citizens' evaluations of campaign pledge fulfillment in canada. *Party Politics*, 26(4):437–447.

Tommaso Fornaciari, Dirk Hovy, Elin Naurin, Julia Runeson, Robert Thomson, and Pankaj Adhikari. 2021. "we will reduce taxes" - identifying election pledges with language models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3406–3419, Online. Association for Computational Linguistics.

Jun Gao, Huan Zhao, Changlong Yu, and Ruifeng Xu. 2023. Exploring the feasibility of chatgpt for event extraction. *ArXiv preprint*, abs/2303.03836.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

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.

Qisheng Hu, Geonsik Moon, and Hwee Tou Ng. 2024. From moments to milestones: Incremental timeline summarization leveraging large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7232–7246.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *ArXiv preprint*, abs/2410.21276.

Hasan Iqbal, Yuxia Wang, Minghan Wang, Georgi Nenkov Georgiev, Jiahui Geng, Iryna Gurevych, and Preslav Nakov. 2024. Open-FactCheck: A unified framework for factuality evaluation of LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 219–229, Miami, Florida, USA. Association for Computational Linguistics.

Lev Konstantinovskiy, Oliver Price, Mevan Babakar, and Arkaitz Zubiaga. 2021. Toward automated factchecking: Developing an annotation schema and benchmark for consistent automated claim detection. *Digital Threats*, 2(2).

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Hao Liao, Jiahao Peng, Zhanyi Huang, Wei Zhang, Guanghua Li, Kai Shu, and Xing Xie. 2023. Muser: A multi-step evidence retrieval enhancement framework for fake news detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4461–4472.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Zheng Liu, Yujia Zhou, Yutao Zhu, Jianxun Lian, Chaozhuo Li, Zhicheng Dou, Defu Lian, and Jian-Yun Nie. 2024. Information retrieval meets large language models. In *Companion Proceedings of the ACM Web Conference 2024*, pages 1586–1589.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, and Rubén Urizar. 2015. SemEval-2015 task 4: TimeLine: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786, Denver, Colorado. Association for Computational Linguistics.

Rrubaa Panchendrarajan and Arkaitz Zubiaga. 2024. Claim detection for automated fact-checking: A survey on monolingual, multilingual and cross-lingual research. *Natural Language Processing Journal*, 7:100066.

Muhammad Reza Qorib, Qisheng Hu, and Hwee Tou Ng. 2025. Just what you desire: Constrained timeline summarization with self-reflection for enhanced relevance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25065–25073.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Dhruv Sahnan, David Corney, Irene Larraz, Giovanni Zagni, Ruben Miguez, Zhuohan Xie, Iryna Gurevych, Elizabeth Churchill, Tanmoy Chakraborty, and Preslav Nakov. 2025. Can llms automate fact-checking article writing?

Michael Schlichtkrull, Yulong Chen, Chenxi Whitehouse, Zhenyun Deng, Mubashara Akhtar, Rami Aly, Zhijiang Guo, Christos Christodoulopoulos, Oana Cocarascu, Arpit Mittal, and 1 others. 2024. The automated verification of textual claims (averitec) shared task. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 1–26.

Michael Schlichtkrull, Zhijiang Guo, and Andreas Vlachos. 2023. Averitec: A dataset for real-world claim verification with evidence from the web. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yohei Seki, Hakusen Shu, Anaïs Lhuissier, Hanwool Lee, Juyeon Kang, Min-Yuh Day, and Chung-Chi Chen. 2024. Ml-promise: A multilingual dataset

for corporate promise verification. *ArXiv preprint*, abs/2411.04473.

Nawar Turk, Eeham Khan, and Leila Kosseim. 2025. Clac at semeval-2025 task 6: A multi-architecture approach for corporate environmental promise verification. *arXiv preprint arXiv:2505.23538*.

Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. 2024. IM-RAG: multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 730–740. ACM.

Yejun Yoon, Jaeyoon Jung, Seunghyun Yoon, and Kunwoo Park. 2024. HerO at AVeriTeC: The herd of open large language models for verifying real-world claims. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 130–136, Miami, Florida, USA. Association for Computational Linguistics.

Caiqi Zhang, Zhijiang Guo, and Andreas Vlachos. 2024. Do we need language-specific fact-checking models? the case of chinese. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1899–1914.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *ArXiv preprint*, abs/2309.01219.

A Implementation Details

A.1 Evidence Retrieval

Following Yoon et al. (2024), we index the training data from AVeriTeC (Schlichtkrull et al., 2023) and retrieve the top-10 most similar question-evidence pairs to the input pledge from the training corpus using BM25. These top-10 question-evidence pairs are then used as the ICL examples. In particular, the prompt is as follow:

> Your task is to generate a question based on the given claim and evidence. The question should clarify the relationship between the evidence and the claim.
>
> `{ICL_examples}`
>
> Now, generate a question that links the following claim and evidence:
>
> Claim: `{pledge_claim}`
> Evidence: `{sentence_evidence}`

We use `Meta-Llama-3.1-8B-Instruct` with a temperature of 0.6 and top-p of 0.9. We generate one question per evidence sentence.

For Google Custom Search, we set the geographic scope to the UK due to our focus on the UK election pledges. In practice, we set the iterative evidence retrieval to two rounds, to balance a good result in practice and our budgets.

A.2 Timeline Construction

We use the below prompt for event description generation and timestamp identification:

> Please only summarize events that are useful for verifying the pledge, and their dates in the JSON format.
>
> `{ICL_examples}`
>
> Please only summarize events that are useful for verifying the pledge: `{pledge}`, and their dates in the JSON format.
>
> Input:
>
> Title: `{document_title}`
> Date: `{document_date}`
> Article: `{document_text}`
>
> Output:

Please note that we use `GPT-4o` for experiments, and constrain the output (including the outputs of the ICL pairs) in the JSON format, for example:

```
{
"events": [
{
"event": "Home Secretary Yvette
Cooper announces new measures to
boost Britain's border security,
including the recruitment of
up to 100 new specialist
intelligence and investigation
officers at the National Crime
Agency (NCA).",
"date": "2024-08-21"
},
{
"event":
"Announcement of a major surge
in immigration enforcement and
returns activity to achieve the
highest rate of removals of those
with no right to be in the UK
since 2018.",
"date": "2024-08-21"
},
{
...
}]
}
```

We use 2 ICL examples to balance the length constraint and model efficiency. We set the top-p and temperature to 0.

A.3 Relevant Event Identification

We use the below prompt for identifying relevant events:

> You are given a pledge, the pledge speaker, and the date of when the pledge is made, and a key event summarized from an online article along with the date of when the event happens. Your task is to determine whether this event summary is useful to track the fulfilment of this pledge.
>
> Yes: The summary presents developments or actions that demonstrate progress (or lack thereof) towards fulfilling the pledge. It helps evaluate whether the pledge is on track or not.
>
> No: The summary only provides background or contextual information, but no progress information for evaluating the fulfilment of the pledge; Or the summary is less than or not related to the pledge.
>
> Below are examples:
>
> {ICL_examples}
>
> Now, please assign a label to the below instance.
>
> Input:
>
> Pledge: {pledge}
> Event summary: {event}. (Event Date: {event_date})
>
> Output:

The model is expected to return `Yes` or `No`, and we also log the log-probability of the first predicted token to support confidence-based ranking.

We use at most 50 ICL examples. In particular, in our demo system, if we are checking a suggested pledge, we use their corresponding annotated data; otherwise, we randomly select instances from all annotated data. We set the top-p and temperature to 0.

B Fact-checkers' Background

Both of the fact-checkers involved in this study (Nasim and Josh) are native English speakers, educated to postgraduate level. One has worked as a fact-checker for two years and overall as a trained journalist for seven years, while the other has worked as a journalist for eight years and in fact-checking for several months.

C Setup of Google Search and GPT-4o for Real-world Evaluation

We use `GPT-4o` with the tool of `web_search`. We set the location as the UK (`GB` in `GPT-4o`), and the `search_context_size` as `high`. We use the same request for initial searching as the input, and use the below prompt to inform the model of the time range:

> Please find the recent online articles (from {time_start} to {time_end}) that can help monitor the fulfilment of the pledge. List only the article URLs, ordered by their usefulness and relevance (most useful and relevant first), one per line.
>
> {pledge}

Similarly, we use the same API for PLEDGETRACKER to call Google Search using the same parameters, and collect the top-10 retrieved results based on their prominence.

To ensure the retrieved URLs are within the correct time range, we further filter all URLs by examining their metadata, and use the useful URLs for evaluation.

Interactive Training: Feedback-Driven Neural Network Optimization

Wentao Zhang
University of Waterloo
w564zhan@uwaterloo.ca

Yang Young Lu
University of Wisconsin-Madison
ylu97@wisc.edu

Yuntian Deng
University of Waterloo
yuntian@uwaterloo.ca

Abstract

Traditional neural network training typically follows fixed, predefined optimization recipes, lacking the flexibility to dynamically respond to instabilities or emerging training issues. In this paper, we introduce Interactive Training, an open-source framework that enables real-time, feedback-driven intervention during neural network training by human experts or automated AI agents. At its core, Interactive Training uses a control server to mediate communication between users or agents and the ongoing training process, allowing users to dynamically adjust optimizer hyperparameters, training data, and model checkpoints. Through three case studies, we demonstrate that Interactive Training achieves superior training stability, reduced sensitivity to initial hyperparameters, and improved adaptability to evolving user needs, paving the way toward a future training paradigm where AI agents autonomously monitor training logs, proactively resolve instabilities, and optimize training dynamics.

1 Introduction

Traditional neural network optimization typically involves setting hyperparameters and defining training strategies before execution, after which practitioners passively observe the training process until it completes or fails (Bergstra and Bengio, 2012). Despite its widespread adoption, this static training paradigm lacks flexibility and responsiveness once training begins. In practice, unforeseen challenges often arise mid-training, such as unstable loss dynamics, underperformance on specific tasks, or vanishing gradients in certain network components, all of which necessitate human intervention (Takase et al., 2023; OLMo et al., 2024). Addressing these issues typically requires prematurely terminating the training job, manually adjusting hyperparameters or data configurations, and restarting the process (Zhang et al., 2022). On managed clusters, repeatedly resubmitting jobs exacerbates these inefficiencies, leading to wasted computational resources and delays due to job-queue overhead.

In this paper, we introduce *Interactive Training*, a framework enabling real-time feedback-driven optimization of neural networks, addressing the limitations of static training paradigms. Inspired by the intuitive act of adjusting a stove based on immediate sensory feedback during cooking, Interactive Training allows human experts or automated AI agents to dynamically intervene during training. Unlike traditional monitoring tools that only visualize training metrics, our approach transforms neural network optimization into an active and responsive process, enabling practitioners to continuously observe training progress, immediately react to emerging issues, and interactively guide the model toward improved outcomes.

Interactive Training enables users (human experts or automated AI agents) to dynamically adjust optimizer parameters, such as modifying learning rates in response to sudden spikes in loss. It supports mid-training updates to training data, allowing models to incorporate new data collected from real-world deployments without restarting training. Users can perform model-level interventions, such as reverting to previous checkpoints upon encountering unstable loss dynamics, or resetting specific parameters when invalid values are detected. The framework also provides gradient-level control, allowing users to dynamically set gradient clipping thresholds based on observed gradient norms, rather than relying on heuristic thresholds.

We implement Interactive Training as an open-source library built on Hugging Face Transformers' widely adopted Trainer class (Wolf et al., 2020). At the core of our implementation is a control server acting as an intermediary between human experts (or AI agents) and the ongoing training process. This control server continuously listens on a predefined network port for incoming commands is-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 851–861

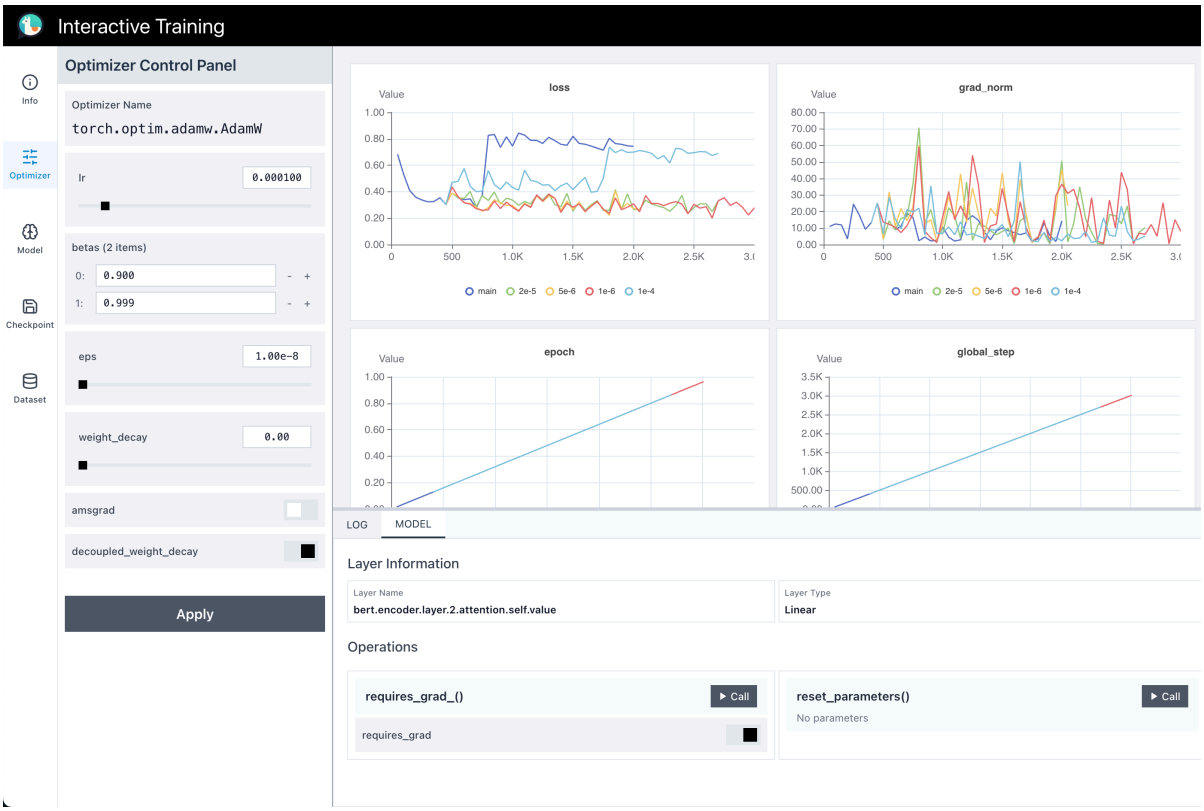


Figure 1: Interactive Training Frontend Dashboard. The left panel provides control tabs organized by Optimizer, Model, Checkpoint, and Dataset, allowing users to dynamically send intervention commands during training (e.g., adjusting the learning rate via the Optimizer panel shown). The right side displays real-time visualizations of training metrics, such as loss and gradient norm. Unlike traditional monitoring tools, this interface supports active two-way communication, enabling users to directly intervene and influence ongoing training processes in real-time.

sued by users. Upon receiving commands (e.g., "set learning rate to 1e-5"), it translates these instructions into corresponding updates to optimizer parameters, model components, dataloaders, or gradients via callback functions invoked after each gradient step. The control protocol exposes its API endpoints through FastAPI. To facilitate ease of use for human experts, we also developed a React-based visualization dashboard, conceptually similar to Weights & Biases (Biewald, 2020), which displays real-time training metrics across multiple plots (Figure 1). Crucially, unlike traditional monitoring tools, our frontend supports two-way communication: it not only visualizes training dynamics but also enables users to actively send control commands directly to the training loop.

We empirically validate Interactive Training through three case studies. First, we demonstrate that experienced human developers, leveraging real-time interactive adjustments, achieve superior optimization results compared to traditional static optimization methods on a language modeling task. Second, we showcase the potential for automated interventions by demonstrating that a general-purpose LLM-based AI agent, prompted with training logs, can autonomously correct suboptimal initial hyperparameters. Finally, we illustrate how our framework enables models to adapt in real-time to user-generated data collected during actual deployments (Albalak et al., 2023; Wettig et al., 2025), using a diffusion-based image generation application (Ho et al., 2020). These studies collectively show exciting potential for human- and AI-driven interactive training.

Interactive Training transforms neural network optimization from a passive, static task into an active and responsive process. We envision a future in which model training is fully interactive, with hyperparameters, training data, and even loss functions dynamically adjusted based on mid-training feedback. Such interventions could be performed by human developers or, with even greater potential, by specialized automated AI agents designed explicitly to monitor training dynamics and evaluate intermediate model checkpoints. By bridging mid-training feedback with dynamic interven-

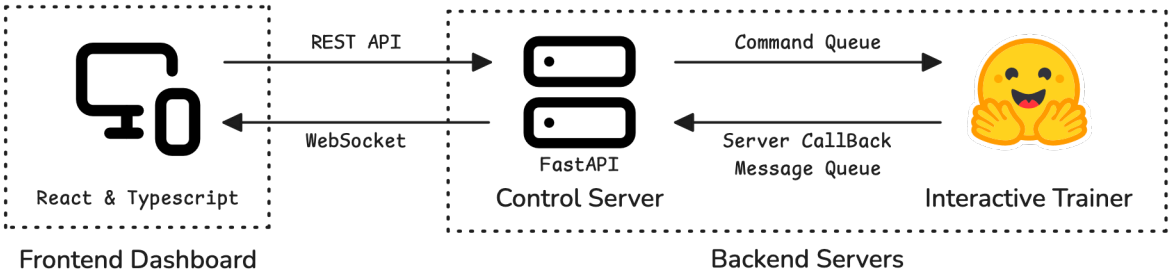


Figure 2: System Architecture. Users interact through a React-based Frontend Dashboard, which visualizes training metrics and sends control commands via REST API. The FastAPI-based Control Server mediates communication by forwarding user commands through command queues to the Interactive Trainer, implemented on top of Hugging Face's Trainer class. The trainer applies received commands via callback functions and sends real-time training updates back to the Control Server, which then broadcasts them to the Frontend Dashboard through WebSockets.

tions, Interactive Training represents a paradigm shift toward continually improving neural network training workflows. To facilitate this vision, we have made our implementation openly available at https://github.com/yuntian-group/interactive-training, with an online demo accessible at https://interactivetraining.ai.

2 Interactive Training Framework

Figure 2 provides an overview of our system architecture. At a high level, Interactive Training consists of three main components: a Control Server, which mediates communication between the trainer and users, managing commands, state updates, and training metrics; an Interactive Trainer, which performs model training and responds dynamically to intervention commands; and a Frontend Dashboard, which provides visualizations of training progress and enables users to issue real-time interventions.

2.1 Control Server

The Control Server acts as the central communication hub in Interactive Training, mediating interactions between the frontend dashboard and the interactive trainer. It serves two primary roles: receiving and dispatching user intervention commands, and broadcasting training updates back to clients.

Implemented using FastAPI, the Control Server exposes a set of APIs, allowing clients such as the frontend dashboard or automated AI agents to send intervention commands. Each command is represented as a JSON message specifying the action type (e.g., adjusting learning rates, checkpoint management) and its parameters (e.g., desired learning rate value). Upon receiving a command, the server enqueues it into command queues categorized by command type for asynchronous processing.

To enable real-time training updates, throughout training, the Interactive Trainer reports metrics such as loss values, gradient norms, and training status updates back to the server via event queues. The Control Server then broadcasts these updates to all subscribed clients, allowing users or automated agents to make timely intervention decisions.

Additionally, the server maintains state information such as training checkpoints, command history, and branched training logs. This state management not only supports reproducibility by logging each intervention but also enables interactive experimentation, such as reverting training to previous checkpoints or branching training trajectories.

The Control Server's modular design supports straightforward extensibility. We detail the currently supported intervention commands and discuss extensibility considerations in Appendix A.

2.2 Interactive Trainer

The Interactive Trainer performs the actual model training, dynamically responding to intervention commands relayed from the Control Server. It extends Hugging Face's widely used `Trainer` class, augmenting it with callback functions that enable real-time interactivity without requiring significant changes to existing training scripts.

At its core, Interactive Trainer is implemented using custom callback functions passed to Trainer:

- **InteractiveCallback**: Adjusts hyperparameters.
- **CheckpointCallback**: Saves/loads checkpoints.
- **LoggingCallback**: Captures training metrics.
- **RunPauseCallback**: Pauses/resumes training.

```python
from transformers import Trainer
from interactive_training import make_interactive # (1) Import helper

# (2) Wrap the standard Trainer class
InteractiveTrainer = make_interactive(Trainer)

# (3) Use them exactly as you would the original Trainer
trainer = InteractiveTrainer(...)

trainer.train()  # Training is now fully interactive!
```

Figure 3: Code changes required to enable Interactive Training.

These callbacks communicate directly with the Control Server via dedicated command and event queues. Upon receiving commands from the Control Server (e.g., "`set learning rate to 1e-5`"), the respective callback updates the trainer's internal state at the next available gradient step, ensuring minimal disruption to the ongoing training loop.

In addition to callbacks for trainer control and metric logging, our framework also supports dynamic training data updates. We provide a function `make_interactive_dataset`, which can wrap PyTorch's `Dataset` and `IterableDataset` classes to make them controllable through user instructions.

Furthermore, the Interactive Trainer supports branching training trajectories. When reverting to earlier checkpoints, it can automatically create new branches of the training state, allowing multiple parallel or sequential training experiments to be compared. Each branch maintains its own isolated training history and checkpoints, providing a clear and reproducible record of experimentation paths.

2.3 Frontend Dashboard

The Frontend Dashboard (Figure 1) provides a user-friendly interface that enables users to visually monitor training progress and intervene in real-time. Built using React and TypeScript, the dashboard displays visualizations of key training metrics updated continuously via WebSocket connections established with the Control Server.

Unlike traditional monitoring dashboards that offer only passive visualizations, our frontend supports two-way communication. Users can dynamically issue intervention commands using intuitive control panels organized by intervention type (Optimizer, Model, Checkpoint, Dataset). Upon issuing a command, the frontend sends structured requests through RESTful API calls to the Control Server, which then communicates with the Interactive Trainer to apply the interventions at runtime.

Furthermore, the dashboard supports branched training trajectories, visualizing multiple experiment paths originating from a common checkpoint, allowing users to compare results from different settings.

Additionally, the dashboard includes a log console at the bottom, which displays logs of each command issued, confirmation responses from the training process, as well as warnings and critical training events (e.g., "Gradient overflow detected").

2.4 Usage Example

To illustrate the simplicity of integrating Interactive Training into existing workflows, we highlight the minimal required modifications to a typical training script in Figure 3. With minor adjustments, users immediately gain interactive control over training.

3 Case Studies

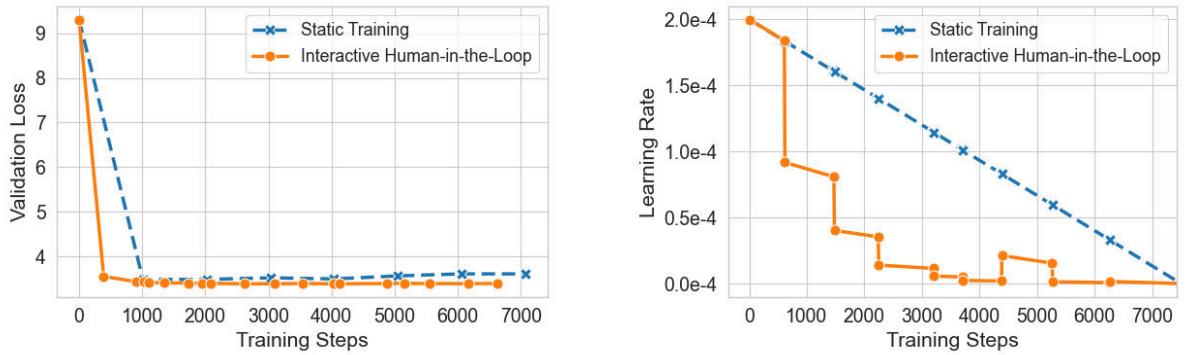
3.1 Human-in-the-Loop Intervention

We first demonstrate the benefits of human-in-the-loop interactive training by finetuning GPT-2 (Radford et al., 2019) on Wikitext-2 (Merity et al., 2017). Our goal is to evaluate whether human interventions could yield improved optimization results.

Baseline For the baseline, we trained the model using a fixed learning rate schedule, starting with an initial learning rate of 1×10^{-5} and linearly annealing it to zero over the entire training duration.

Human Intervention The interactive training setup mirrored the baseline, except that a human expert dynamically adjusted the learning rate based on training dynamics visualized in the dashboard.

Results Figure 4a compares the results of both approaches. The interactive method achieves lower validation losses than the static baseline. By inspecting the learning rate schedules (Figure 4b), we find that the human expert effectively responded to



(a) Validation loss curves

(b) Learning rate schedules

Figure 4: Comparison of human-in-the-loop interactive training versus traditional static training for finetuning GPT-2 on Wikitext-2. **(a)** Validation losses. Dynamic human interventions lead to improved optimization compared to the static baseline, which uses a fixed learning rate schedule. **(b)** Actual learning rates used over steps.

the model's real-time behavior. For instance, upon observing training loss oscillation resulting from an initially high learning rate, the expert reduced the learning rate, improving convergence.

3.2 LLM-in-the-Loop Intervention

Next, we investigate the feasibility of automating training interventions by leveraging an AI agent. Specifically, we evaluate whether an LLM, provided with training logs, can correct training instabilities caused by suboptimal hyperparameters.

Setup We follow the same setup as in the previous study but deliberately introduce instability by initializing the training with an excessively large learning rate (5×10^{-3}) and disabling the learning rate scheduler. This excessively high learning rate causes poor convergence of the training.

Instead of human interventions, we introduce an automated LLM-based agent, using OpenAI's o4-mini model (OpenAI, 2025). At every step, the LLM agent receives a textual summary of recent training logs—including current and historical training losses, validation losses, learning rates, and step counts—and is prompted to determine the next action regarding the learning rate (doubling, halving, or keeping it unchanged). The detailed prompt template is provided in Appendix B.

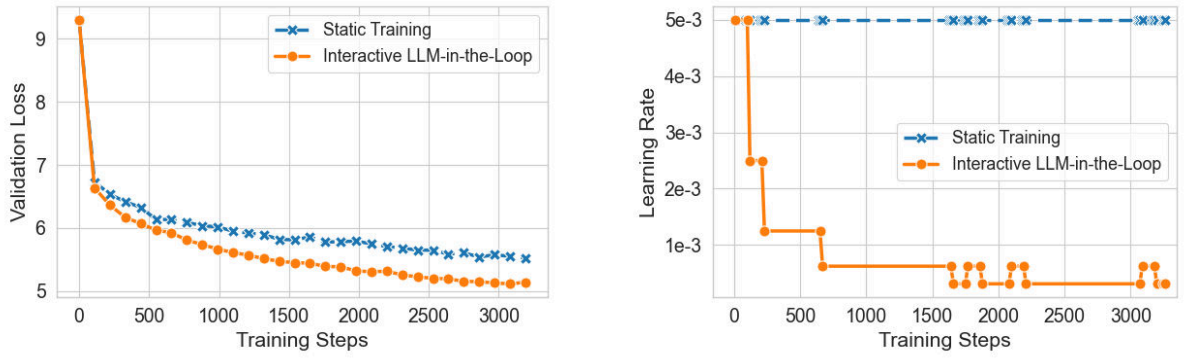
Results The results of this automated intervention approach are in Figure 5. LLM-in-the-loop training recovers from the initial suboptimal learning rate by recommending timely reductions. This study demonstrates the potential of using AI agents to automate interactive training interventions.

3.3 Real-time Training Data Updates

Finally, we demonstrate how Interactive Training enables continuous model improvement through dynamic updates to training data collected from real-world deployments. We apply our framework to NeuralOS (Rivard et al., 2025), which uses a diffusion model to simulate a real operating system by predicting the next screen frame given user mouse and keyboard inputs. After deploying the initial NeuralOS model online at `https://neural-os.com`, we continuously collected real user interactions.

Setup Initially, the NeuralOS model was trained for two months on a large synthetic dataset generated from scripted interactions. After deployment, we gathered 746 demonstration sequences (88K frame transitions) from real user interactions over a period of 14 days. Using the Interactive Training framework, we dynamically updated the training data of a continuously running finetuning process, incorporating newly collected data on-the-fly. Additionally, model checkpoints during finetuning were continuously uploaded, updating the deployed model to reflect improvements in real-time.

Results After dynamically finetuning NeuralOS, we observed substantial improvements, especially for tasks frequently performed by actual users, such as interacting with the Firefox browser and creating new folders. Representative examples illustrating the significant improvement achieved by incorporating real user data are shown in Figure 7 (Appendix C). This case study demonstrates that Interactive Training effectively enables deployed models to adapt to real-world usage patterns.



(a) Validation loss curves

(b) Learning rate schedules

Figure 5: Comparison of LLM-in-the-loop automated intervention versus static training with a fixed, excessively large learning rate. **(a)** Validation losses. LLM-based intervention effectively stabilizes optimization. **(b)** Learning rate trajectory. Initially high learning rate is reduced by the LLM agent in response to observed loss instabilities.

4 Limitations

Reproducibility Different experts or AI agents training the same model might perform different interventions, leading to different outcomes. While we acknowledge this inherent variability, it is worth noting that large-scale neural network training today already involves substantial expert intervention. For instance, Meta's OPT language model required at least 35 manual restarts due to hardware failures and involved manually selecting checkpoints to recover from loss divergences (Zhang et al., 2022). To mitigate reproducibility concerns, our implementation logs all interventions, enabling replay.

Expertise Requirement Interactive training requires human experts or automated agents to possess expertise to identify appropriate intervention points and apply effective corrective actions. AI agents may lack adequate demonstrations in their training data to reliably intervene, due to the novelty of this optimization paradigm. However, we view this limitation as an opportunity, motivating future research into specialized intervention agents.

5 Future Work

Feedback-Driven Data Adjustment Real-time interventions could enable new optimization strategies. Users or AI agents could periodically evaluate intermediate checkpoints to identify model weaknesses and then dynamically adjust training data accordingly, either by injecting targeted synthetic examples, or by adjusting data mixture weights to emphasize relevant existing examples.

Training Health Diagnostic Metrics Just as periodic checkups help humans address potential health issues, model training could benefit from analogous health monitoring metrics. One promising direction is to develop "health indicators" such as the standard deviation of hidden states across training examples to detect "dead" neurons (Ioffe and Szegedy, 2015). More sophisticated analyses (Hu et al., 2023) could also provide signals prompting human or AI interventions.

AI Agents for Training Intervention Finally, we envision a future where AI agents autonomously monitor health indicators and proactively intervene to improve training stability and efficiency. While this paper demonstrates simple log-based prompts to a general-purpose LLM, specialized intervention agents explicitly trained to detect anomalies and guide training represent a promising direction.

6 Related Work

Human-in-the-Loop Machine Learning A rich body of work has explored human interventions during model training. In active learning, the learning algorithm remains in control but queries human annotators for labels on selected examples (Mosqueira-Rey et al., 2023). Interactive machine learning goes further by allowing human feedback beyond just labeling, such as by correcting predictions or adjusting inputs (Fails and Olsen Jr, 2003). Another paradigm, machine teaching, gives human domain experts explicit control over the training process, such as by designing the sequence or structure of tasks to transfer knowledge to the model (Simard et al., 2017). These

approaches demonstrate the value of human insight during training; however, they often rely on predefined schedules or specific forms of input rather than truly real-time, open-ended intervention. Our Interactive Training framework aims to allow humans (or AI agents) to intervene training at any moment, which extends human-in-the-loop learning from static plans to live control.

Automated ML and Adaptive Optimization Orthogonal to human guidance, AutoML research has developed methods to automate hyperparameter tuning and training optimization. Traditional approaches include Bayesian optimization and bandit strategies that adaptively select hyperparameter configurations across trial runs (Li et al., 2018). More recent techniques seek to adapt within a single run: learning rate scheduling is routinely used to vary the step size during training, and researchers have even applied reinforcement learning to discover optimized scheduling policies automatically (Subramanian et al., 2023; Xu et al., 2019). Also related to our work, Population-Based Training (PBT) (Jaderberg et al., 2017) learns an automatic dynamic schedule of hyperparameters. Interactive Training complements AutoML by enabling both automated agents and human experts to adjust training trajectories in real time. Instead of treating training as a black-box process to tune from the outside, our framework opens the loop, so scheduling decisions and hyperparameter tweaks can occur on the fly, guided by live signals or human judgment.

AI Agents for Training Control and Debugging Researchers have started to consider AI agents as participants in the training loop. For example, Epperson et al. (2025) developed an interactive debugger for multi-agent AI workflows that allows a user to reset agents to earlier states and alter their messages mid-execution. Modern visualization platforms are beginning to integrate automated agents to monitor experiment runs, detect anomalies, and even suggest hyperparameter adjustments based on the accumulated training data (Relevance AI, 2025). However, such agents typically remain advisory tools; they do not directly plug into the training loop to enact immediate interventions. Our Interactive Training framework builds on this idea by permitting both humans and AI agents to not only analyze but also modify a running training job. This bridges a gap between AI-driven monitoring and actual training control, turning insights into on-the-fly actions.

7 Conclusion

We presented Interactive Training, a framework that reimagines neural network training as an interactive, feedback-driven process, with either humans or AI agents dynamically controlling training strategies mid-training. Through real-time interventions, Interactive Training enables adjusting optimization parameters, training data, and model components on-the-fly based on insights gained during training. Case studies show advantages over traditional static training paradigms: improved accuracy, reduced sensitivity to initial hyperparameters, and real-time adaptation to evolving application needs.

Interactive Training introduces a new dimension to training workflows: responsiveness. Just as modern software development evolved from rigid release cycles toward agile and continuous integration practices, we advocate for a parallel shift in neural network optimization. The training process need not remain a static black box where practitioners must passively await results. Instead, it can become an interactive process, allowing continuous monitoring and intervention based on emerging information and feedback. We have open-sourced our framework, inviting the community to provide feedback and contribute to further development.

Acknowledgements

Yuntian Deng acknowledges support from an NSERC Discovery Grant (RGPIN-2024-05178), a Starter Grant from the University of Waterloo, and research funding from Manulife. Wentao Zhang is supported in part by these sources and by the Dr. Derick Wood Graduate Scholarship, generously funded by Ms. Mary Chen.

References

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. 2023. Efficient online data mixing for language model pre-training. In *R0-FoMo:Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The journal of machine learning research*, 13(1):281–305.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Will Epperson, Gagan Bansal, Victor C Dibia, Adam Fourney, Jack Gerrits, Erkang Zhu, and Saleema

Amershi. 2025. Interactive debugging and steering of multi-agent ai systems. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Jerry Alan Fails and Dan R Olsen Jr. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Michael Y. Hu, Angelica Chen, Naomi Saphra, and Kyunghyun Cho. 2023. Latent state models of training dynamics. *Transactions on Machine Learning Research*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, and 1 others. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. 2023. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, and 1 others. 2024. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*.

OpenAI. 2025. OpenAI o4-mini. https://en.wikipedia.org/wiki/OpenAI_o4-mini. Released April 16, 2025.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Relevance AI. 2025. Weights & biases ai agents. https://relevanceai.com/agent-templates-software/weights---biases. Accessed: 2025-07-05.

Luke Rivard, Sun Sun, Hongyu Guo, Wenhu Chen, and Yuntian Deng. 2025. Neuralos: Towards simulating operating systems via neural generative models. *arXiv preprint arXiv:2507.08800*.

Patrice Y Simard, Saleema Amershi, David M Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and 1 others. 2017. Machine teaching: A new paradigm for building machine learning systems. *arXiv preprint arXiv:1707.06742*.

Shreyas Subramanian, Vignesh Ganapathiraman, and Aly El Gamal. 2023. Learned learning rate schedules for deep neural network training using reinforcement learning.

Sho Takase, Shun Kiyono, Sosuke Kobayashi, and Jun Suzuki. 2023. Spike no more: Stabilizing the pre-training of large language models. *arXiv preprint arXiv:2312.16903*.

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. 2025. Organize the web: Constructing domains enhances pre-training data curation. In *Forty-second International Conference on Machine Learning*.

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, and 3 others. 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.

Zhen Xu, Andrew M Dai, Jonas Kemp, and Luke Metz. 2019. Learning an adaptive learning rate schedule. *arXiv preprint arXiv:1909.09712*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Supported Interactive Commands

Interactive Training supports real-time intervention through structured commands. We first describe the general command message format, followed by details of supported commands grouped by category.

A.1 Command Message Format

All commands follow a uniform JSON message structure:

```
{
  "command": "[command_name]",
  "args": "[command_arguments_as_json]",
  "time": [unix_timestamp],
  "uuid": "[unique_identifier]",
  "status": "[status]"
}
```

- `command`: Type of command (as listed below).
- `args`: JSON-formatted arguments specific to the command.
- `time`: UNIX timestamp indicating when the command was issued.
- `uuid`: Unique identifier for tracking command status.
- `status`: Current execution state, which can be one of: `"requested"`, `"pending"`, `"running"`, `"completed"`, `"success"`, or `"failed"`.

A.2 Supported Commands

Interactive Training currently supports the following real-time intervention commands, organized by their intended use:

Optimizer Adjustment

- `update_optimizer`: Adjust optimizer hyperparameters (e.g., learning rates, momentum, weight decay) during training.

  Example:

```
{
  "command": "update_optimizer",
  "args": "{\"lr\": {\"value\": 1e-5}}"
}
```

Checkpoint Management

- `save_checkpoint`: Save the current model state as a checkpoint.
- `load_checkpoint`: Load a previously saved checkpoint and optionally branch training from that point.

  Example:

```
{
  "command": "load_checkpoint",
  "args": "{\"uuid\": \"[uuid]\"}"
}
```

Training Control

- `pause_training`: Pause the training loop.
- `resume_training`: Resume training after being paused.
- `stop_training`: Terminate the training process immediately.

Model Interventions

- `model_layer_operation`: Run a method of a layer such as resetting or reinitializing specified model parameters (e.g., upon detecting NaN values or activation collapse).
- `model_layer_parameter_update`: Update layer hyper-parameter, e.g. dropout value of a dropout layer.

Dataset Management

- `update_dataset`: Update training data mid-training, e.g., to incorporate newly collected user data.
- `update_dataset_runtime_hyperparameters`: Update dataset run time hyper-parameters, e.g. the mixing ratio of different part or subset of datasets.

Evaluation and Monitoring

- `do_evaluate`: Trigger a model evaluation step on the validation dataset.

A.3 Extensibility

Our framework is designed for easy extensibility. New commands and interactions can be added by defining new command types, registering handlers, and extending the control server and trainer callbacks. We encourage community contributions of new intervention commands and interaction patterns via our open-source repository.

```
You are an expert in language model tuning. You are going to adjust the learning rate of fine-tuning a GPT-2 model using the
WikiText-2 train data with a constant learning rate scheduler. Your goal is to minimize the final validation loss.

# Log History

current step: {{current_step}}

current learning rate:

{{current_lr}}

learning rate history:

{{lr_history}}

train loss history:

{{train_loss_history}}

validation loss history:

{{valid_loss_history}}


# Instruction

- Based on the log history, decide how to change the learning rate. The log history includes metrics like validation loss,
training loss, and epoch count.

- You MUST output choose between following three actions:
    1. "Double", double the learning rate
    2. "Half", reduce the learning rate to 50% of the learning rate.
    3. "Same", make no change.

- Respond in JSON format with an explanation within 100 words:

{
    "explanation": "<Explanation in 100 words>",
    "action": "<'Double', 'Half', or 'Same'>"
}

# Response
```

Figure 6: Prompt used for the LLM-based automated learning rate adjustment.

B Detailed LLM Prompt

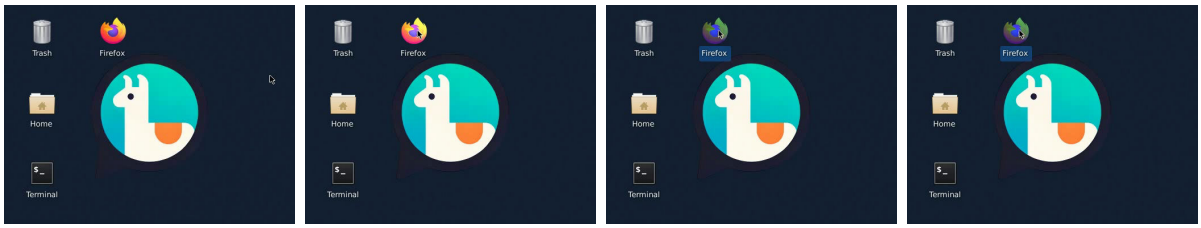
The exact textual prompt provided to the LLM-based intervention agent for automated learning rate adjustments is shown in Figure 6. At each intervention point, the placeholders such as `current_step`, `current_lr`, `lr_history`, `train_loss_history`, and `valid_loss_history` are dynamically replaced with the most recent training data and metrics before the prompt is sent to the LLM agent. The agent is explicitly instructed to respond with a clear JSON-formatted decision—choosing either to double, halve, or keep the learning rate unchanged—accompanied by a brief explanation within 100 words.

C Results for NeuralOS Case Study

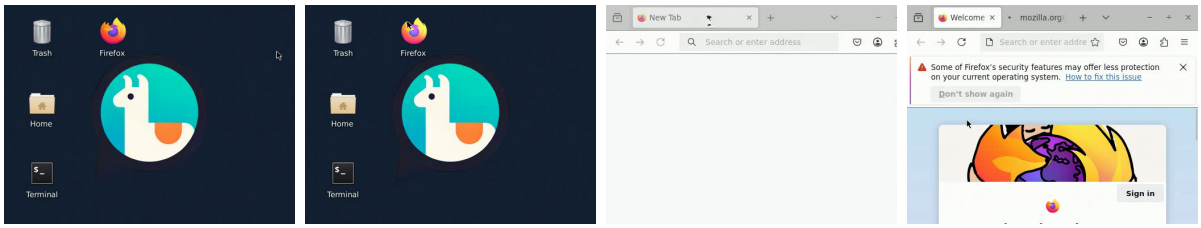
Figure 7 shows representative qualitative comparisons illustrating the improvements obtained by finetuning the NeuralOS model with real-time training data updates from actual user interactions collected through an online demo at `https://neural-os.com`. The figure consists of four rows: the top two rows depict model behavior when interacting with the Firefox browser, comparing performance before (first row) and after (second row) interactive finetuning. Before finetuning, attempts to open Firefox usually failed, leaving the screen on the desktop. This is because opening Firefox is challenging to predict, as the browser takes a longer time to launch compared to other applications (often more than 40 frames after clicking). After finetuning, however, opening Firefox is typically successful, due to frequent occurrences of Firefox interactions in the collected real user data. Similarly, the bottom two rows demonstrate the model's improved capability in creating new folders, comparing behavior before (third row, unsuccessful) and after (fourth row, successful) incorporating real user data. These examples highlight how Interactive Training effectively enables the model to naturally adapt to tasks frequently attempted by real users.

Pred. Frame (early) · · · Pred. Frame · · · Pred. Frame · · · Pred. Frame (late)

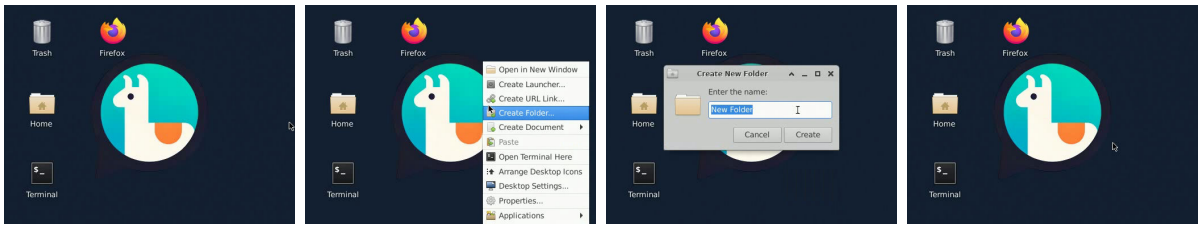
Firefox Interaction (Before Real-Time Training Data Updates, Unsuccessful)



Firefox Interaction (After Real-Time Training Data Updates, Successful)



Folder Creation (Before Real-Time Training Data Updates, Unsuccessful)



Folder Creation (After Real-Time Training Data Updates, Successful)

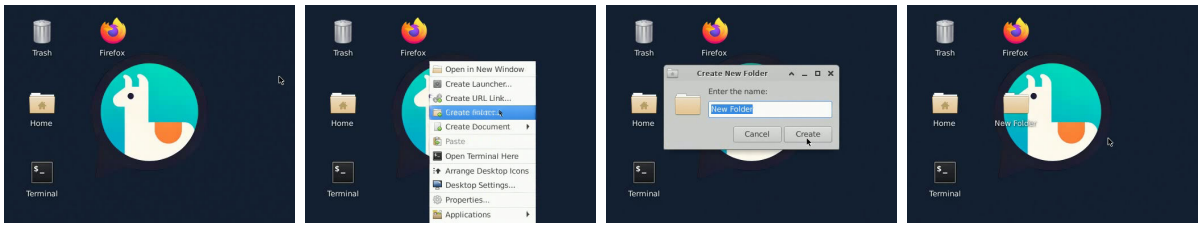


Figure 7: Qualitative comparison illustrating improvements in the NeuralOS model's behavior before and after real-time training data updates using data collected from real users. The first two rows demonstrate model predictions when launching the Firefox browser, while the last two rows demonstrate creating a new folder. For each task, the top row is before finetuning and the bottom row after finetuning.

Metamo: Empowering Large Language Models with Psychological Distortion Detection for Cognition-aware Coaching

Hajime Hotta[1], **Huu-Loi Le**[2], **Manh-Cuong Phan**[3], **Minh-Tien Nguyen**[3]*
[1] Hajime Institute, Kuala Lumpur, Malaysia.
hotta@hajime.institute
[2] AI Academy Vietnam, Vietnam.
loilh@aiacademy.edu.vn
[3] Hung Yen University of Technology and Education, Hung Yen, Vietnam.
cuongpm@spkt.edu.vn; tiennm@utehy.edu.vn

Abstract

We demonstrate Metamo, a browser-based dialogue system that transforms an off-the-shelf large language model into an empathetic coach for everyday workplace concerns. Metamo introduces a light, single-pass wrapper that first identifies the cognitive distortion behind an emotion, then recognizes the user's emotion, and finally produces a question-centered reply that invites reflection, all within one model call. The wrapper keeps the response time below two seconds in the API, yet enriches the feedback with cognitively grounded insight. A front-end web interface renders the detected emotion as an animated avatar and shows distortion badges in real time, whereas a safety layer blocks medical advice and redirects crisis language to human hotlines. Empirical tests on public corpora confirmed that the proposed design improved emotion-recognition quality and response diversity without sacrificing latency. A small user study with company staff reported higher perceived empathy and usability than a latency-matched baseline. Metamo is model-agnostic, illustrating a practical path toward cognition-aware coaching tools.

1 Introduction

Conversational coaching tools have begun to accompany performance reviews, career planning, and daily self-reflection in the workplace (Mitchell et al., 2022; Beinema et al., 2023; Arakawa and Yakura, 2024; Ong et al., 2024). Built on large language models (LLMs), such systems can deliver fluent encouragement; however, they typically operate with a single prompt that folds perception and advice into one turn. It leads to two limitations. First, the generated guidance is often generic, without empathy, because the prompt lacks an explicit representation of *why* the user feels distressed. Second, multi-step prompting or tool calling pushes response latency beyond the five-second window that humans perceive as natural.

Psychological research attributes many negative emotions to recurring thought patterns, known as *cognitive distortions* (Beck, 2020). Recent NLP studies have shown that prompting LLMs to label such distortions can sharpen downstream reasoning (Chen et al., 2023; Wang et al., 2024; Lim et al., 2024), but existing pipelines are usually designed with two separate steps: emotion recognition with shallow textual representation, and response generation. This makes ill-suited for interactive and empathetic coaching (Liong and Patel, 2024).

We argue that conversational coaching systems should be empowered by a deeper understanding of human thinking that fosters empathetic effective communication between humans and AI models. To address this argument, this study presents **Metamo**, a demonstration system that is empowered by psychological distortion detection. The system uses a compact prompt to first yield a ten-way distortion label, then emotion labels, and finally, a dialogue move that combines a Socratic follow-up question with a brief reframing suggestion. The design preserves latency while making cognitive analysis transparent: the web client maps the emotion to an avatar's facial expression and displays the distortion as a color-coded badge.

We evaluated Metamo under zero-shot conditions on EMPATHETICDIALOGUES (Rashkin et al., 2019) and a subset of EDOS (Welivita et al., 2021). The wrapper improves emotion recognition accuracy and response diversity over strong baselines, yet maintains real-time speed on both GPT-3.5 via API and a local Qwen-7B model. Human evaluation with corporate employees further indicated higher System Usability Scale scores and perceived empathy. The system is model-agnostic and requires no fine-tuning, offering a concrete blueprint for deploying cognition-aware coaching without compromising speed, safety, or user experience.

*Corresponding Author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 862–872

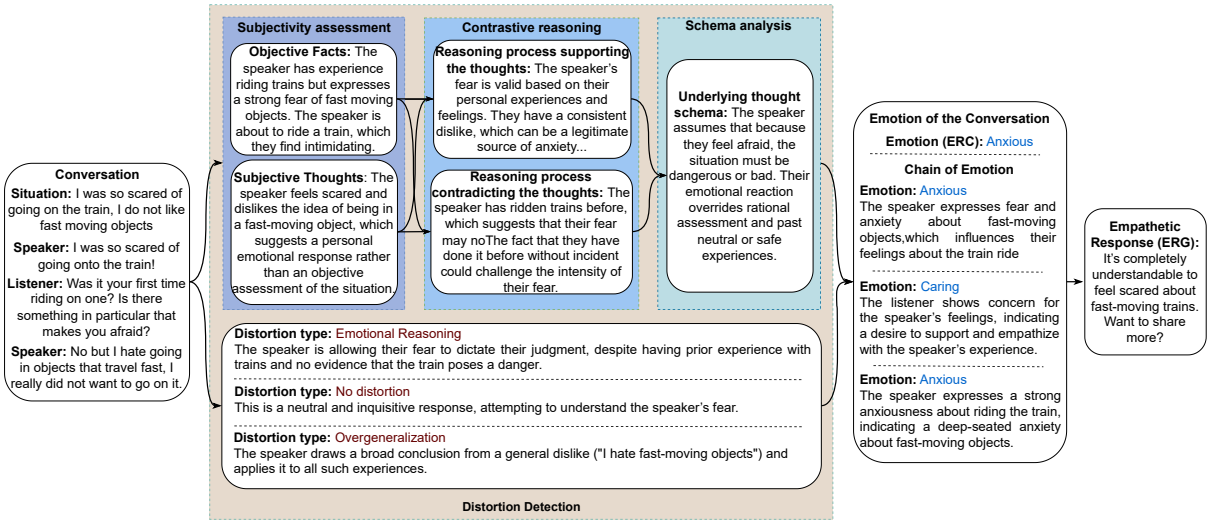


Figure 1: The Metamo model receives a conversation and its situation, and predicts distortions. It next recognizes emotions using cognitive indicators. It finally uses emotions and cognitive indicators for generation.

2 System Design

The system includes three main steps: distortion detection, emotion recognition in conversations, and empathetic response generation.

2.1 Distortion Detection

Cognitive distortion detection serves as a foundational procedure in cognitive behavioral therapy (CBT) (Beck, 2020) and has recently been investigated to enhance the reasoning capabilities of LLMs (Chen et al., 2023; Wang et al., 2024; Lim et al., 2024). According to CBT, mental disorders are closely associated to maladaptive thought patterns that manifest through emotional responses. For instance, individuals experiencing depression or anxiety often show negative thoughts, leading to negative emotions. Inspired by DoT (Dianogsis-of-Thought) (Chen et al., 2023), we introduce distortion detection to detect cognitive distortions of participants in a conversation. The detection includes three steps: subjectivity assessment, contrastive reasoning, and schema analysis. Distortion detection is a core module that establishes a cognitive model for subsequent reasoning processes.

Subjectivity assessment To detect cognitive distortions, the initial step involves evaluating the subjectivity of the speaker (or listener). This is because their utterances typically contain a mixture of objective information (factual content) and subjective interpretations or opinions. The subjectivity assessment is responsible for identifying and differentiating between these objective facts and subjective thoughts given the situation. This crucial step clarifies factual statements from personal opinions, providing clearer evidence for further analysis.

Contrastive reasoning Once the assessment has been established, contrastive reasoning offers two distinct reasoning processes to evaluate subjective thoughts using objective evidence. The first process establishes support for the thoughts, while the second constructs arguments that contradict them. By contrasting these opposing interpretations derived from the same set of facts, the method can more precisely uncover and characterize the speaker's underlying thought schemas.

Schema analysis This step elucidates the reasons behind the speaker's formation of specific reasoning processes. Drawing on psychological principles, we refer a "schema" as a cognitive structure that integrates an individual's knowledge, beliefs, and expectations to construct a coherent cognitive model. By identifying and analyzing these schemas, the method can effectively build a comprehensive cognitive model of the conversation to support subsequent reasoning tasks.

Given a situation of a conversation and the current utterance, the detection classifies distortions of the utterance into 10 types: personalization, mind reading, overgeneralization, all-or-nothing thinking, emotional reasoning, labeling, magnification, mental filter, should statements, and fortune-telling (Beck, 2020; Shreevastava and Foltz, 2021). These types are injected into the prompt and LLMs are required to predict the distortion which has the highest probability given an input conversation.

2.2 Emotion Recognition

Once the cognitive model has been built, the method recognizes emotions of a conversation. Given the situation and cognitive indicators, the method predicts the emotion of the conversation (or utterances) by prompting LLMs. Different from sentiment analysis that only uses shallow textual features, our method takes into account cognitive modeling in the form of distortions that are in a deeper level than the shallow textual level.

The chain of emotions We observe that emotional states have interconnections with distortions in utterances of a conversation. This is because the chain of emotions illustrates the emotional states engagement of both the speaker and listener in a conversation. Constructing this chain involves two important perspectives. From a psychological perspective, it reflects the transformation from cognitive modeling at a deeper level to emotional expression at a shallow level. By capturing individuals' thoughts throughout the dialogue, the chain of emotions enables the method to perform more nuanced and fine-grained reasoning. Regarding emotion expression, the overall emotion conveyed by a conversation should align with this emotional progression. For instance, in Figure 1, the conversation's emotion can be inferred as "Anxious" based on the chain "Anxious-Caring-Anxious". This chain of emotions is used for response generation.

2.3 Empathetic Response Generation

The final step is to generate empathetic responses using information from the situation, cognitive indicators, and the chain of emotions. In practice, the method uses a prompt that includes three components for emotion recognition and response generation. The empathy aspect is enhanced by using constraints in the prompt that guide LLMs to generate more empathetic responses.

3 Evaluation

The Metamo was applied to two tasks: emotion recognition in conversations (ERC) and empathetic response generation (ERG), which are critical components of an empathetic coaching system.

3.1 Settings

Datasets The evaluation uses two datasets. `EmpatheticDialogues` (Rashkin et al., 2019) is a text-only corpus. Each conversation contains a situation and a conversation created by the interaction between the speaker and listener. `EDOS` is a large-scale dataset for empathetic response generation (Welivita et al., 2021). It includes 1M dialogues from movie subtitles. Due to the limited budget, 500 dialogues from EDOS were randomly sampled. Table 1 shows the statistics of the datasets.

Table 1: Statistics of ERC and ERG datasets.

Dataset	train	dev	test	# labels
EmpatheticDialogues	19,533	2,770	2,547	18, 32
EDOS (500)	—	—	500	32

Baselines Baselines were selected with two categories: LLMs and task-specific models. For LLMs, the proposed method was tested with **GPT-3.5** (gpt-3.5-turbo-0125), **GPT-o3-mini**, **GPT-4o-mini** because of promising results in the zero-shot setting (Nori et al., 2023; Peng et al., 2023; Mozikov et al., 2024; Katz et al., 2024). We did not use GPT-4 due to the high cost. The method was further confirmed with **Qwen-7B** (Yang et al., 2024) and **Mistral-7B** (Jiang et al., 2023), small open models showing competitive results in various tasks. We also reimplemented **InsideOut** (Mozikov et al., 2024).

For task-specific models, **Transformer** uses the encoder-decoder architecture (Waswani et al., 2017) for ERG. **MoLE** (Lin et al., 2019) handles both ERC and ERG tasks by first detecting user emotions and then generating empathetic responses. **EmpDG** (Li et al., 2020) utilizes coarse-grained dialogue and token-level emotions. **CEM** (Sabour et al., 2022) takes into account the understanding of users' emotions and cognition for ERC and ERG. **KEMP** (Li et al., 2022) combines ConceptNet and emotional lexicons to enhance the representation of implicit emotions. **text-CNN**, **bcLSTM**, and **DiaRNN** uses the text modality for training ERC models (Poria et al., 2019). **MM-DFN** is a multimodal model, but we report the results using the text modality (Hu et al., 2022). **CFEG** (Chen et al., 2024) is a cause-aware ERG approach that uses Chain-of-Though prompts for fine-tuning LLMs.

Evaluation metrics Following previous studies (Cai et al., 2023; Mozikov et al., 2024), the evaluation uses accuracy for ERC; and ROUGE (Rouge, 2004), BLEU (Papineni et al., 2002), Distinct-1 (Li et al., 2016), and human evaluation for ERG.

3.2 Results and Discussion

3.2.1 Performance Comparison

Emotion recognition in conversations Table 2 summarizes the accuracy of LLM-based methods

for the ERC task. As observed, the proposed multi-agent framework achieves the best results in most of all cases. More importantly, the proposed method is significantly better than the baselines that directly use LLMs. A possible reason is that cognitive modeling in the form of distortion detection helps the method to simulate the thinking process and then to understand deeply the reasoning mechanism of humans. Interestingly, the gap between GPT-o3-mini and the proposed method is small, in which GPT-o3-mini outputs better performance on EmpatheticDialogues. A possible reason is that GPT-o3-mini is good at reasoning,[1] so adding more complex prompts seems to be unnecessary. The InsideOut method achieves low performance, as our argument in Section 1, in which InsideOut uses a shallow emotion analysis while the proposed model uses a deeper level of cognitive modeling.

Table 2: The performance of ERC. **Bold** text is the best. 18 classes were shared from Mozikov et al. (2024).

# Classes	LLMs	Baseline	InsideOut	Metamo
EmpatheticDialog				
32	GPT-3.5	35.89	36.59	**45.66**
	GPT-o3-mini	**48.49**	40.11	48.29
	GPT-4o-mini	42.87	40.28	**48.48**
	Qwen	39.07	30.43	**42.51**
	Mistral	37.38	34.35	**42.30**
18	GPT-3.5	55.71	39.97	**56.61**
	GPT-o3-mini	**62.35**	41.46	61.80
	GPT-4o-mini	56.69	45.70	**62.36**
	Qwen	45.94	40.16	**56.73**
	Mistral	46.09	43.35	**56.17**
EDOS				
32	GPT-3.5	24.60	27.40	**28.60**
	GPT-o3-mini	27.60	25.80	**28.40**
	GPT-4o-mini	25.80	25.05	**28.80**
	Qwen	25.40	22.00	**28.00**
	Mistral	21.40	21.00	**25.65**

Table 3: Comparison with strong task-specific methods. The results are derived from Mozikov et al. (2024).

EmpatheticDialogues (32 classes)						
LLMs	**Metamo**	**MoEL**	**MINE**	**EmpDG**	**CEM**	**KEMP**
GPT-3.5	**45.66**					
GPT-o3-mini	**48.29**					
GPT-4o-mini	**48.48**	31.74	30.96	31.65	36.84	36.57
Qwen	**42.51**					
Mistral	**42.30**					

Table 3 shows the comparison of Metamo with task-specific models of ERC. The proposed method outperforms task-specific models. There are two possible reasons. First, the proposed framework uses LLMs that have shown promising scores for ERC (Mozikov et al., 2024). Second, LLMs are empowered by cognitive modeling by using multiple agents. It allows the framework to better understand the thinking process of humans, leading to the improvements of emotion prediction. We could not compare the method with strong models on EDOS due to different settings.

Empathetic response generation Table 4 reports the comparison between the proposed framework and strong methods of ERG.

Table 4: The performance of ERG.

LLMs	Method	B-1	B-2	R-1	R-2	Dist-1
EmpatheticDialogues						
GPT-3.5	Baseline	11.49	2.05	12.78	**1.89**	0.989
	InsideOut	11.11	1.56	11.43	1.39	0.983
	Metamo	**13.56**	**4.04**	**13.30**	1.77	**0.991**
GPT-o3-mini	Baseline	9.26	0.95	**9.91**	**0.85**	**0.976**
	InsideOut	6.98	0.27	5.98	0.23	0.974
	Metamo	**9.29**	**1.47**	9.18	0.56	0.961
GPT-4o-mini	Baseline	11.52	2.13	13.17	**1.87**	**0.983**
	InsideOut	8.97	1.23	11.09	1.33	0.982
	Metamo	**13.06**	**3.97**	**13.74**	1.78	0.982
Qwen	Baseline	10.47	1.68	**13.34**	**1.56**	0.985
	InsideOut	9.47	1.29	11.62	1.35	0.982
	Metamo	**12.54**	**3.12**	12.32	1.36	**0.988**
Mistral	Baseline	8.74	2.06	**14.73**	**1.93**	0.924
	InsideOut	8.47	1.83	13.56	1.77	**0.989**
	Metamo	**11.64**	**3.23**	14.19	1.65	0.944
EDOS						
GPT-3.5	Baseline	13.71	2.56	9.14	0.75	**0.981**
	InsideOut	11.42	2.44	8.06	0.73	0.976
	Metamo	**13.80**	**4.92**	**10.64**	**2.11**	0.959
GPT-o3-mini	Baseline	9.14	1.21	7.00	0.37	0.968
	InsideOut	8.74	1.16	5.26	0.23	0.982
	Metamo	**9.74**	**1.68**	**7.05**	**0.66**	**0.984**
GPT-4o-mini	Baseline	11.29	2.13	8.52	0.62	**0.975**
	InsideOut	10.06	1.93	7.90	0.60	0.970
	Metamo	**12.46**	**2.45**	**9.14**	**0.65**	0.965
Qwen	Baseline	11.48	2.72	7.67	0.75	0.990
	InsideOut	10.22	1.97	6.73	0.70	0.991
	Metamo	12.25	1.95	**8.38**	**0.63**	**0.994**
Mistral	Baseline	**9.93**	**2.11**	**9.32**	0.75	0.934
	InsideOut	9.36	1.65	9.22	**0.96**	**0.977**
	Metamo	9.32	1.91	6.03	0.47	0.963

The scores share the trend with the ERC task in Table 2, in which Metamo achieves better scores than the methods that only use LLMs in almost all cases. The improvements come from cognitive modeling and multiple agents for response generation. The performance of the proposed framework is better than that of InsideOut and baseline. It validates our assumption that modeling cognition benefits the reasoning process.

We also compared the method with strong ERG models. Table 5 shows that there are large gaps between LLM-based methods and task-specific fine-tuned models. This is because LLM-based methods

[1] https://openai.com/index/openai-o3-mini/

work in the zero-shot setting, while task-specific models are fine-tuned with training data of the downstream task. Also note that the performance of the proposed method can still be improved by using few-shot learning.

Table 5: Comparison with task-specific methods on EmpatheticDialogues. The results of task-specific methods are from Mozikov et al. (2024) and Zhang et al. (2024).

Method	B-1	B-2	R-1	R-2	Dist-1
Transformer	18.07	8.34	17.22	4.21	0.36
MoEL	18.07	8.30	18.24	**4.81**	0.59
MINE	18.60	8.39	17.08	4.05	0.47
EmpDG	**19.96**	9.11	**18.02**	4.43	0.46
CEM	16.12	7.29	15.77	4.50	0.62
KEMP	16.72	7.17	16.11	3.31	0.66
CFEG	—	**10.54**	—	—	**2.96**
GPT-3.5	**13.56**	**4.04**	**13.30**	1.77	**0.991**
GPT-o3-mini	9.29	1.47	9.18	0.56	0.961
GPT-4o-mini	**13.06**	**3.97**	**13.74**	1.78	0.982
Qwen	12.54	3.12	12.32	1.36	**0.988**
Mistral	**11.64**	**3.23**	14.19	1.65	0.944

We conducted human evaluation because generating tokens with empathy challenges automatic evaluation for both meaning and linguistics. We randomly selected 100 samples for five annotators with four metrics: Fluency (**F**), Identification (**I**), Empathy (**E**), Suggestion (**S**), and Overall Effectiveness (**OE**) (Mozikov et al., 2024). Fluency measures the coherency and fluency of the generated response. Identification assesses the accurate identification of the problem that the speaker needs support from the listener. Empathy estimates the understanding empathy of the response of the listener given the emotion and feeling of the speaker. Suggestion measures the appropriateness and usefulness of advice or suggestions from the listener. Overall Effectiveness estimates effectiveness in providing information of the listener that supports the speaker's emotions. The annotators were trained with a guideline in three steps: (i) reading the situation of the conversation, (ii) understanding the conversation and the gold reference (the final utterance of the listener), and (iii) giving the score of the generated response based on steps (i) and (ii). After training, each annotator gave a score, ranging from 1 to 10, for each metric.[2]

The results in Table 6 confirm that the proposed framework obtains the best scores in almost all cases, followed by the baseline. This is because Metamo incorporates cognitive modeling in the form of multiple agents. Among the metrics, the fluency scores are the highest, followed by the empathy and identification scores. It is understandable that LLMs can generate fluency responses. The gap score for empathy between the proposed framework and the baselines is quite large, which confirms the contribution of cognitive modeling to enhance empathy in LLMs.

[2]The evaluation link: https://tinyurl.com/yupkp395

Table 6: Human evaluation of ERG.

LLMs	Method	F	I	E	S	OF	Average
EmpatheticDialogues							
GPT-3.5	Baseline	8.21	7.04	7.53	6.37	6.89	7.20
	InsideOut	8.13	6.88	7.10	6.76	6.30	7.03
	Metamo	**9.30**	**8.27**	**9.15**	**7.14**	**8.47**	**8.46**
GPT-o3- mini	Baseline	8.44	7.76	7.40	5.14	7.00	7.14
	InsideOut	7.99	7.46	7.12	5.46	6.53	6.77
	Metamo	**9.03**	**7.92**	**8.09**	**7.11**	**8.10**	**8.08**
GPT-4o-mini	Baseline	9.09	7.08	8.04	5.38	7.51	7.42
	InsideOut	8.09	7.87	7.15	6.64	8.06	7.56
	Metamo	**9.19**	**8.21**	**9.12**	**7.23**	**8.51**	**8.45**
Qwen	Baseline	7.78	7.11	7.03	5.42	6.60	6.70
	InsideOut	7.49	6.90	6.57	5.82	6.05	6.56
	Metamo	**9.11**	**8.15**	**8.08**	**7.23**	**8.00**	**8.11**
Mistral	Baseline	7.08	6.86	6.04	6.50	6.63	6.56
	InsideOut	7.63	6.94	6.56	5.95	6.45	6.70
	Metamo	**9.00**	**8.09**	**8.04**	**7.10**	**7.99**	**8.04**

3.2.2 Prompt Sensitivity

To assess the robustness of our framework to prompt design, we conducted a series of sensitivity experiments in which the original prompt was systematically perturbed. We considered three settings: **Ablation**, where a key portion of the instruction was omitted by removing two guiding questions in the diagnosis-of-thought step; **Terminology**, where domain-specific terminology was replaced with semantically similar but less technical expressions (Table 12); and **Rephrased**, where ChatGPT was instructed to rewrite the original prompt in its own words. The observation was done with 100 samples on EmpatheticDialogues.

Table 7: Accuracy of different prompt settings on the ERC (18 classes) task in EmpatheticDialogues.

Model	Metamo			
	Original	Ablation	Terminology	Rephrased
GPT-3.5	59.00	60.00	**61.00**	54.00
GPT-4o-mini	67.00	65.00	**68.00**	57.00
GPT-o3-mini	**62.00**	61.00	59.00	**62.00**

Table 7 reports ERC performance across models. The results indicate that all three settings yield comparable accuracy to the original prompt, with deviations of only one to three percentage points. Notably, Terminology did not substantially degrade accuracy, suggesting that the models can tolerate terminological shifts as long as semantic intent is preserved. In contrast, Rephrased showed a clearer drop in performance for GPT-3.5 and GPT-4o-mini. This suggests that when the model is asked to freely

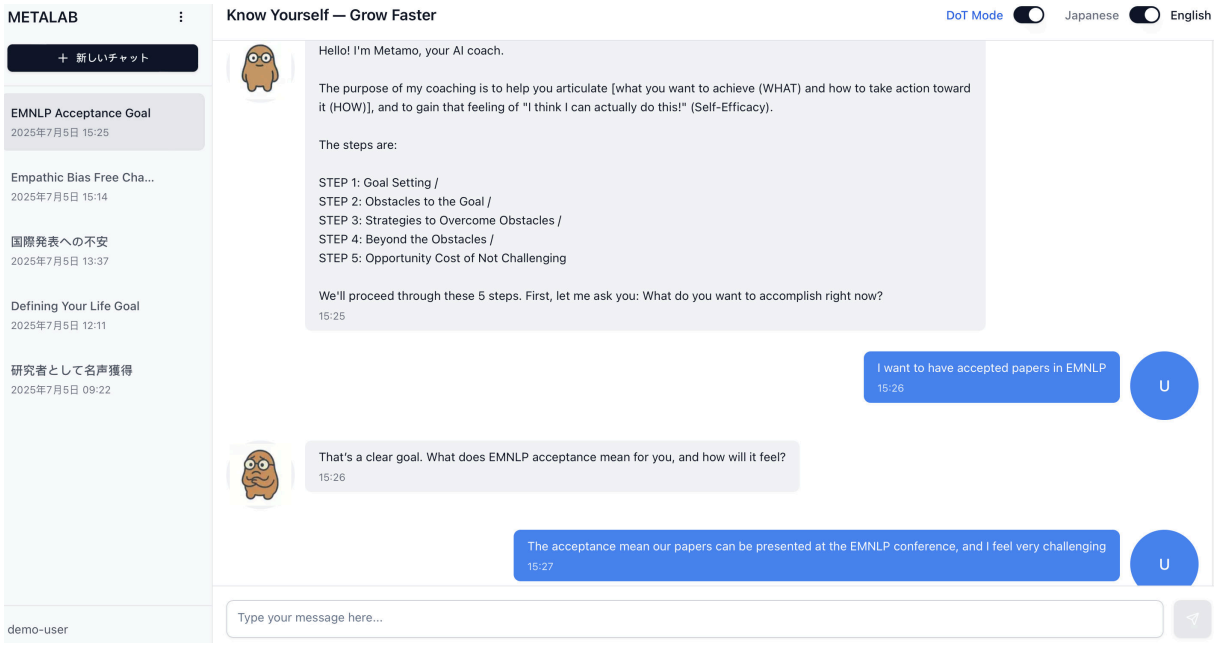


Figure 2: Metamo: the coaching system using Metamo. The system interacts with the user in the case that the user wants to have an accepted paper at EMNLP but he is not so confidence due to a very competitive conference.

rewrite the prompt, it tends to omit or soften certain instructions that are crucial but not obvious on the surface. As a result, the new prompt loses part of the structure that originally guided the reasoning process, leading to weaker results.

Table 8: Performance of different prompt settings on the ERG task in EmpatheticDialogues.

Model	Metamo	B-1	B-2	R-1	R-2	Dist-1
GPT-3.5	Original	13.91	3.35	13.14	**1.46**	**0.991**
	Ablation	14.07	2.87	13.51	0.71	**0.991**
	Terminology	13.63	3.31	13.51	1.12	0.986
	Rephrased	**14.15**	**3.37**	13.13	1.37	0.986
GPT-4o mini	Original	13.27	**4.18**	14.18	**2.04**	0.984
	Ablation	**14.52**	3.98	**14.98**	1.54	0.987
	Terminology	13.20	3.93	14.75	1.83	0.983
	Rephrased	12.17	3.27	12.21	1.54	**0.993**
GPT-o3 mini	Original	**8.44**	**1.99**	9.85	**0.99**	0.947
	Ablation	6.58	0.97	**10.79**	0.31	0.948
	Terminology	8.16	0.91	9.25	0.64	0.956
	Rephrased	7.77	1.51	9.68	0.62	**0.958**

Table 8 reports the performance of different prompt settings on the ERG task. Terminology appears relatively stable, with scores close to the original prompt, confirming that semantic intent rather than precise wording drives performance. Ablation yields mixed outcomes: for some models it slightly improves BLEU-1 or ROUGE-1, but at the cost of lower BLEU-2 and diversity, suggesting that removing diagnostic questions narrows responses to simpler patterns. In contrast, Rephrased consistently degrades results for GPT-4o-mini and GPT-o3-mini, while only marginally affecting GPT-3.5. This indicates that free-form rewriting can strip away the implicit structure of the original prompt, leading to less coherent and less cognitively grounded outputs. Overall, the results highlight that empathetic response generation is more sensitive to prompt design than emotion recognition, and that carefully engineered instructions remain crucial for sustaining quality.

3.2.3 Running Time and Cost

This section provides the investigation of running time and cost which are critical for actual applications. To do that, 100 samples were randomly selected from the EmpatheticDialogues dataset. Table 9 shows the average numbers computed in three runs. The baseline only using GPT-4o-mini is the best in terms of cost. Our method follows the baseline on the cost. For running time, the ratio also shows that our method is faster for ERG but slower for ERC. This is because ERG bases on ERC, in which ERC outputs distortion detection and chain of emotions for generating empathetic responses. In contrast, the baseline uses two different prompts for the two tasks. The InsideOut method takes a lot of time and cost due to the collaboration of multi-agents. The observation shows that Metamo is practical for actual cases.

4 Demonstration Scenario

This section shows the demonstration scenario of Metamo in two parts: system demonstration and human evaluation.

Table 9: The running time and cost of 100 samples. The time ratio was computed over the beseline.

Method	Task	Time (s)	Time ratio (%)	Cost (USD)
Baseline	ERC	0.966	—	0.002
	ERG	2.370	—	0.008
InsideOut	ERC	5.912	617.798	0.056
	ERG	9.094	403.998	0.102
Metamo	ERC	1.011	105,712	0.015
	ERG	1.787	79.3869	0.033

4.1 Metamo

The proposed prompt (Figure 1) was applied to build Metamo, an empathetic conversational coaching system that makes a huge impact (Mitchell et al., 2022; Arakawa and Yakura, 2024). We refer to "know yourself" in terms of understanding distortions and emotions. The system can be easily to be adapted to education or healthcare support.

Figure 2 shows the Metamo[3] system,[4] which offers two languages: Japanese and English. The central panel shows the main screen of conversations between users and the system, and the configuration. When users access the system and create a new conversation, the main screen in the center shows the guideline to setup the coaching in five steps shown in Figure 2. Metamo guides users to follow these steps to make the final advice. In conversations, Metamo detects distortions of users, predicts their emotions, shows appropriate avatars, and generates empathetic responses. It also utilizes a Socratic follow-up question combined with a brief reframing suggestion to interact with users.

4.2 Human Evaluation

Tasks We asked 38 people (English and Japanese, 32-57 years old) to judge the user experience of the system in three tasks: career change, English learning, and team relations. The scenario seed shown to the participants (summarized) is as follows. For career change, the scenario is "You feel anxious about a job transition." For English learning, the scenario is "You aim to raise your TOEIC score in 3 months." For team relation, the scenario is "You face conflict with a team-mate at work."

Settings Participants rate the system in three settings: (1) bare minimal prompt, no coaching steps, no avatar using ChatGPT, (2) the system with 5-step strategy prompt, neutral avatars, and (3) the proposed Metamo system.

[3]System: https://dot.hajime-institute.com/

[4]Video: https://www.youtube.com/watch?v=ui1NuSMFoxw

Procedure Each participant completed nine sections (three conditions: emotion classification, generation quality, and overall satisfaction) with the score of 1-5 (larger is better). The free chat is larger than or equal five round-trip within six minutes. Table 10 shows the instrumentation.

Table 10: The instrumentation of human evaluation.

Construction	Items	Timing
Facial empathy	3	post-session
Working alliance	10	post-session
Session satisfaction	8	post-session

The instrumentation is shown in Appendix A.2.

Human evaluation results Table 11 shows the results of human evaluation of Metamo. It indicates that our method obtains the highest scores of user satisfaction thanks to cognitive distortion detection. It is understandable that Metamo integrates cognitive distortion into the reasoning process, leading to more human-like responses. The STEP method obtains the second-best overall score. The possible reason is the use of neutral avatars compared to the dynamic facial expressions of Metamo. ChatGPT produces the lowest scores because the simple prompt does not use cognition-aware coaching.

Table 11: User experience of the system.

Method	Classification	Generation	Overall
Bare (ChatGPT)	3.16	3.73	3.82
STEP without Bare	3.15	4.24	4.57
Ours	**4.34**	**4.33**	**4.79**

5 Conclusions

This paper introduces Metamo, an empathetic conversational coaching system. Empowered by psychological distortion detection, Metamo can analyze thinking patterns of users in a deeper level, and then predict their emotions for empathetic response generation. The dialog move is also combined with a Socratic follow-up question with a brief reframing suggestion. Metamo is evaluated on two tasks: emotion recognition in conversations and empathetic response generation on two benchmark datasets. Promising experimental results leverage Metamo for actual coaching scenarios.

Future work will apply Metamo to other domains, e.g. education, and utilize multimodal data to improve the model's performance. In addition, human evaluation with larger, more diverse populations and longitudinal deployments will be important to further validate these findings.

Limitations

Although achieving promising results, the proposed method has the following limitations. First, it models cognition using distortion detection. It allows us to deeply understand the thought of the participants in a conversation. As a result, the framework obtains competitive performance when the conversation has distortion indicators. In other cases, understanding emotion states requires other cognitive aspects. Note that in mental health or health care treatment, distortion detection should be confirmed by psychologists. Second, the method is designed to handle text-only data; therefore, multimodal datasets may challenge the method and require other designs. However, the method can be easily extended for working with multimodal data by adding new agents, e.g., visual agents. Finally, Metamo requires more guardrails for actual user deployment. This helps Metamo avoid LLM jailbreaking by using prompt-based attacks.

Human evaluation, while valuable, is based on a relatively small number of samples and participants. The human evaluation of empathetic response generation in Section 3.2.1 only uses 100 samples with five annotators. The participant pool in Section 4.2 (38 users, English and Japanese, aged 32–57) is relatively narrow, which may limit the generalizability of the results to broader populations. In addition, the scenarios (career change, English learning, team conflict) and short interaction time (six minutes) simplify real-world conditions, and thus may not fully reflect long-term or diverse use cases. Furthermore, the study relies on self-reported satisfaction and alliance measures, which, while standard in user studies, can be influenced by expectations or novelty. Despite these constraints, the consistent improvements across different tasks and settings suggest that the observed benefits of cognition-aware responses are not accidental but stem from systematic design choices.

Ethics Statement

The authors confirm that this study does not have ethical issues. Thanks to the sharing of prompts from InsiteOut and DoT authors, we can successfully run the two methods. People participated in the human evaluation process with a clear guideline and observation to avoid human bias. We do not use any personal information from annotators for experiments. Code and datasets are collected from public GitHub links from original papers. All experiments do not have specific parameter tuning to maintain a fair comparison among methods.

References

Riku Arakawa and Hiromu Yakura. 2024. Coaching copilot: blended form of an llm-powered chatbot and a human coach to effectively support self-reflection for leadership growth. In *Proceedings of the 6th ACM Conference on Conversational User Interfaces*, pages 1–14.

Judith S Beck. 2020. *Cognitive behavior therapy: Basics and beyond*. Guilford Publications.

Tessa Beinema, Harm Op den Akker, Hermie J Hermens, and Lex van Velsen. 2023. What to discuss?—a blueprint topic model for health coaching dialogues with conversational agents. *International Journal of Human–Computer Interaction*, 39(1):164–182.

Hua Cai, Xuli Shen, Qing Xu, Weilin Shen, Xiaomei Wang, Weifeng Ge, Xiaoqing Zheng, and Xiangyang Xue. 2023. Improving empathetic dialogue generation by dynamically infusing commonsense knowledge. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7858–7873.

Xinhao Chen, Chong Yang, Man Lan, Li Cai, Yang Chen, Tu Hu, Xinlin Zhuang, and Aimin Zhou. 2024. Cause-aware empathetic response generation via chain-of-thought fine-tuning. *arXiv preprint arXiv:2408.11599*.

Zhiyu Chen, Yujie Lu, and William Wang. 2023. Empowering psychotherapy with large language models: Cognitive distortion detection through diagnosis of thought prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4295–4304.

Dou Hu, Xiaolong Hou, Lingwei Wei, Lianxin Jiang, and Yang Mo. 2022. Mm-dfn: Multimodal dynamic fusion network for emotion recognition in conversations. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7037–7041. IEEE.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. 2024. Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270):20230254.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Qintong Li, Hongshen Chen, Zhaochun Ren, Pengjie Ren, Zhaopeng Tu, and Zhumin Chen. 2020. Empdg: Multi-resolution interactive empathetic dialogue generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4454–4466.

Qintong Li, Piji Li, Zhaochun Ren, Pengjie Ren, and Zhumin Chen. 2022. Knowledge bridging for empathetic dialogue generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 10993–11001.

Sehee Lim, Yejin Kim, Chi-Hyun Choi, Jy-yong Sohn, and Byung-Hoon Kim. 2024. Erd: A framework for improving llm reasoning for cognitive distortion classification. *arXiv preprint arXiv:2403.14255*.

Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, and Pascale Fung. 2019. Moel: Mixture of empathetic listeners. 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 121–132.

Kyrin Liong and Nadya Shaznay Patel. 2024. Empathetic coaching conversations: Being emotionally present and connected with students. In *Coaching Students in Higher Education*, pages 53–69. Routledge.

Elliot Mitchell, Noemie Elhadad, and Lena Mamykina. 2022. Examining ai methods for micro-coaching dialogs. In *Proceedings of the 2022 CHI conference on human factors in computing systems*, pages 1–24.

Mikhail Mozikov, Nikita Severin, Maria Glushanina, Mikhail Baklashkin, Andrey Savchenko, and Ilya Makarov. 2024. Insideout: Unifying emotional llms to foster empathy. In *ECAI 2024*, pages 4499–4502. IOS Press.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. 2023. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*.

Qi Chwen Ong, Chin-Siang Ang, Davidson Zun Yin Chee, Ashwini Lawate, Frederick Sundram, Mayank Dalakoti, Leonardo Pasalic, Daniel To, Tatiana Erlikh Fox, Iva Bojic, et al. 2024. Advancing health coaching: A comparative study of large language model and health coaches. *Artificial Intelligence in Medicine*, 157:103004.

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.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. Meld: A multimodal multi-party dataset for emotion recognition in conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 527–536.

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. Association for Computational Linguistics.

Lin CY Rouge. 2004. A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization of ACL, Spain*, volume 5.

Sahand Sabour, Chujie Zheng, and Minlie Huang. 2022. Cem: Commonsense-aware empathetic response generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11229–11237.

Sagarika Shreevastava and Peter Foltz. 2021. Detecting cognitive distortions from patient-therapist interactions. In *Proceedings of the Seventh Workshop on Computational Linguistics and Clinical Psychology: Improving Access*, pages 151–158.

Ruiyi Wang, Stephanie Milani, Jamie Chiu, Jiayin Zhi, Shaun Eack, Travis Labrum, Samuel Murphy, Nev Jones, Kate Hardy, Hong Shen, et al. 2024. Patient-: Using large language models to simulate patients for training mental health professionals. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12772–12797.

A Waswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. 2017. Attention is all you need. In *NIPS*.

Anuradha Welivita, Yubo Xie, and Pearl Pu. 2021. A large-scale dataset for empathetic response generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1264.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Yiqun Zhang, Fanheng Kong, Peidong Wang, Shuang Sun, SWangLing SWangLing, Shi Feng, Daling Wang, Yifei Zhang, and Kaisong Song. 2024. Stickerconv: Generating multimodal empathetic responses from scratch. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7707–7733.

A Appendix

A.1 Prompt Sensitivity

Table 12 shows the Terminology setting used for the observation of prompt sensitivity.

Table 12: Replacement of technical terms in the Terminology setting.

Original term	Replacement term
Cognitive distortion	Thinking error
Diagnosis of thought	Thought analysis
Reasoning process	Logic pathway
Underlying cognition mode	Core belief
Objective facts	Verifiable facts
Subjective opinions	Personal beliefs

A.2 Instrumentation

This section shows the instrumentation in Tables 13, 14, and 15 for human evaluation in Section 4.

A.3 Prompt Details

Table 16 provides detailed information of the prompts used in the baseline (LLMs) and proposed method.

Table 13: Metamo facial-empathy short scale.

No.	Instrumentation
1	The AI coach's character expression accurately reflected how I felt.
2	The AI coach's expression made me feel understood and at ease.
3	The expression shown by the AI coach was appropriate for the situation.

Table 14: Working alliance inventory.

No.	Instrumentation
1	I believe the AI coach likes me.
2	The AI coach and I respect each other.
3	I feel confident working with the AI coach.
4	The AI coach and I trust each other.
5	The AI coach and I agree on the goals for this session.
6	The AI coach and I are working toward mutually agreed-upon goals.
7	The goals we set are important to me.
8	The AI coach and I agree on what tasks will help me reach my goals.
9	I understand what the AI coach wants me to do during the session.
10	I believe the tasks we worked on will help me.

Table 15: Session satisfaction questionnaire.

No.	Instrumentation
1	How would you rate the overall quality of this AI-coaching session?
2	Did you receive the kind of support you wanted from the AI coach?
3	To what extent did the session meet your needs?
4	If a friend had a similar problem, would you recommend this AI coach?
5	How satisfied are you with the amount of help you received?
6	Has the session helped you deal more effectively with your problem?
7	Overall, how satisfied are you with the AI coach's support?
8	If you needed help again, would you return to this AI coach?

Table 16: The prompts using in the proposed method.

Method	Prompt
Baseline	You will be provided with a conversation and the following situation with the emotion of the speaker and listener. Your task is to create a response to the conversation naturally, as if you were part of the conversation. Your response should reflect understanding, care, and emotional connection.
Metamo	Given a conversation, your task is to: 1) Finish a few diagnose of thought questions to analyze the thought patterns of the participants. Then based on the diagnose of thought analysis, 2) identify if there is cognitive distortion in the conversation. 3) Recognize the specific types of the cognitive distortion. These are the following diagnosis of thought questions: a) What is the situation? Find out the facts that are objective; what are the participants thinking or imagining? Find out the thoughts or opinions that are subjective. b) What makes the participants think the thought is true or is not true? Find out the reasoning processes that support and do not support these thoughts. c) Why do the participants come up with such reasoning processes supporting the thought? What's the underlying cognition mode of it? Based on the diagnosis of thought, for each speech made by the speaker and listener, identify if there is cognitive distortion, specify the type of distortion and provide an explanation. Here we consider the following common distortions: (followed by the descriptions and examples of all ten prompts included in the dataset metadata) <utterance: speaker or listener>: < speech >[<cognitive distortion type >or No distortion] <explanation > Based on the diagnosis of the speaker's and listener's thoughts and the situation of the conversation, your task is to determine the emotions of each utterance in the conversation. <utterance: speaker or listener>: <speech>[emotion] <explanation> The situation of the conversation is following: {situation} The conversation is as follows: {conversation} Based on the above analysis, generate an empathetic response.

InTriage: Intelligent Telephone Triage in Pre-Hospital Emergency Care

Kai He[1], Qika Lin[1], Hao Fei[1],
Chng Eng Siong[2], Dehan Hong[3], Marcus Eng Hock Ong[*4,5], Mengling Feng[*1],

[1]National University of Singapore, Singapore
[2]Nanyang Technological University, Singapore
[3]Singapore Civil Defence Force, Singapore
[4]Duke-NUS Medical School, Singapore
[5]Singapore General Hospital, Singapore

Abstract

Pre-hospital Emergency Care (PEC) systems are critical for managing life-threatening emergencies where rapid intervention can significantly impact patient outcomes. The rising global demand for PEC services, coupled with increased emergency calls and strained emergency departments, necessitates efficient resource utilization through Telephone Triage (TT) systems. However, existing TT processes face challenges such as incomplete data collection, communication barriers, and manual errors, leading to high over-triage and under-triage rates. This study proposes InTriage, an AI-driven multilingual TT system to provide decision support for triage. InTriage enhances accuracy by transcribing emergency calls, extracting critical patient information, prompting supplementary, and providing real-time triage decisions support. We conducted an evaluation on a real-world corpus of approximately 40 hours of telephone data, achieving a word error rate of 14.57% for speech recognition and an F1 score of 73.34% for key information extraction. By improving communication efficiency and reducing triage errors, InTriage offers a scalable solution to potentially help address the growing demands on PEC systems globally[1].

1 Introduction

Pre-hospital Emergency Care (PEC) refers to medical assistance provided to patients outside hospital settings, typically followed by their transfer to the nearest medical facility (Mohammadi et al., 2022). PEC plays a vital role in managing life-threatening emergencies where every second is crucial, as delays can determine the outcome between survival, permanent disability, or death. Rapid patient assessment and timely intervention by PEC personnel are therefore indispensable. Globally, the demand for PEC services has been rising, as evidenced by increasing emergency calls, longer ambulance response times, and overcrowded Emergency Departments (EDs) (Brady, 2020; Inokuchi et al., 2022; Singapore Civil Defence Force, 2023). These trends highlight the strain on PEC systems and underscore the need for innovative strategies to maintain their efficiency and effectiveness.

Figure 1: Yearly data growth of emergency calls from 2000 to 2023 in Singapore. The data illustrates a quick upward trend with peaks in 2021 and 2022, reflecting significant need for more efficient triage systems.

Telephone Triage (TT) is a crucial process in PEC systems, enabling call-takers to systematically gather patient information and assess clinical severity through structured questioning. This process, which begins when an emergency call is answered and concludes with determining dispatch priority or redirecting low-acuity cases to alternative care pathways (ACPs), plays a pivotal role in optimizing PEC resource utilization. In Singapore, the number of emergency calls has grown by 400% over the past two decades, with a marked acceleration during the COVID-19 pandemic (see Figure 1). This rapid growth has posed substantial challenges to existing TT systems, which heavily rely on manual processes and call-taker expertise.

Under such conditions, a robust TT system is

[1]The introduction video is at https://www.youtube.com/watch?v=a-XnPJ4dlyw, and the demonstration video at https://www.youtube.com/watch?v=dVYonyK5-cY. The symbol (*) denotes co-corresponding authors.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 873–885
November 4-9, 2025

essential for prioritizing high-acuity, time-sensitive cases while redirecting low-acuity cases to appropriate healthcare services. This reduces the burden on EDs and ensures that critically ill patients receive prompt and focused care (Vicente et al., 2013). However, TT performance has been shown to be suboptimal in many countries. A study found an over-triage rate of 26.0% and an under-triage rate of 4.9% in Thailand, suggesting a higher prevalence of over-triage compared to under-triage (Huabbangyang et al., 2023). Another study found that over-triage rates ranged from 9.9% to 87.4%, while under-triage rates varied from 1.6% to 72.0% in North America, depending on the population and methodology (Lupton et al., 2023). In Singapore, our investigation indicates that over/under-triage rates are approximately 45% and 5%, underscoring the urgent need for improvements.

Three primary challenges contribute to these inefficiencies. First, emergency call-takers often struggle to gather essential information and assess case acuity, even with established protocols. Key data, such as medical history, clinical risk factors, and previous call records, are frequently underutilized (Wu et al., 2024; He et al., 2019). Second, communication barriers between callers and call-takers delay responses and may lead to adverse outcomes. For example, obtaining accurate location information in a multilingual country like Singapore is challenging due to diverse accents and variations names. Third, manual data entry during the triage is time-consuming and prone to human error, reducing both efficiency and accuracy.

To address these issues, we propose an artificial intelligence-based system ***InTriage***, as shown in Figure 2, which aims to assist call-takers and improve TT performance. *InTriage* leverages a multilingual Automatic Speech Recognition (ASR) agent capable of transcribing emergency calls in English, Chinese, and Malay simultaneously. Using Hotword boosting technology (Yang et al., 2024), the ASR enhances the recognition of location names. The transcriptions from the ASR agent enable a Natural Language Understanding (NLU) agent to extract critical patient information, primary complaints, and assess the caller's stress levels. The system also highlights pertinent medical history and recommends subsequent questions, ensuring a streamlined and accurate triage process. A Dialogue Management (DM) agent continuously calculates a confidence score, and once a predefined threshold is met, a Natural Language Generation (NLG) agent delivers the final triage recommendations and emergency instructions.

By automating several aspects of the triage process, *InTriage* enhances decision-making accuracy, reduces over-triage and under-triage rates, and ensures efficient use of PEC resources. Furthermore, its integration of advanced AI capabilities, addresses the critical need for precise and timely triage in high-pressure emergency settings.

To the best of our knowledge, our study presents the first intelligent multimodal TT system, which designed to meet the demands of PEC in a multilingual context. *InTriage* has been tested by the Singapore Civil Defence Force[2] (SCDF), achieving satisfactory results and offering a scalable solution for wider adoption. This work pioneers a intelligent TT systems in the following aspects.

- *InTriage* introduces a multimodal AI pipeline that seamlessly integrates five different agents. InTriage enables precise and real-time processing of emergency calls, reducing the call-taker's burden under high-pressure PEC conditions.
- *InTriage* is specifically designed to operate in multilingual environments, utilizing advanced Hotword boosting to address one of the most critical and challenging issues.
- *InTriage* integrates stress-level analysis, medical history retrieval, and emergent instructions generation, providing more comprehensive information to avoid potential risks.

2 Related Work

Different Goals and Scopes. Many previous studies on triage systems are designed for mass casualty incidents (MCIs), which are involving various approaches, including wearable technologies, electronic tagging, and telemedicine solutions. For instance, Adler et al. (2011), Greiner-Mai and Donner (2010), and Park (2021) introduced IT-supported and IoT-based e-triage systems that monitor vital signs and provide real-time updates. Besides, some triage systems are designed for disease-specific scenarios to improve diagnosis and care during outbreaks. For respiratory infections like COVID-19, Villafuerte et al. (2023) developed a

[2]The Singapore Civil Defence Force (SCDF) is a uniformed government organization under the Ministry of Home Affairs. It provides essential national services such as firefighting, rescue operations, and emergency medical services.

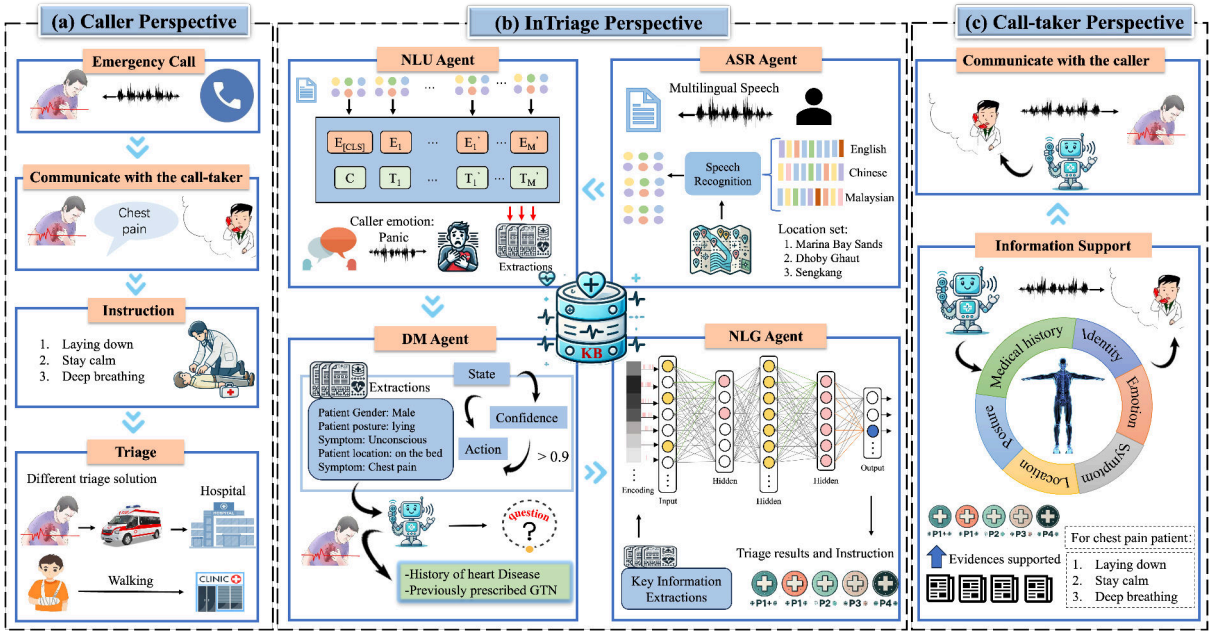


Figure 2: The architecture of InTriage, enhancing efficiency and ensuring accurate triage. We illustrate the functions of *InTriage* from three different perspectives, including the caller, InTriage, and call-taker.

telemedicine virtual assistant to diagnose conditions based on real-time vital signs and symptom evaluations. Khanna et al. (2023) and Soltan et al. (2021) employed ML models to predict COVID-19 severity using clinical biomarkers and structured data for identifying critical patients. These systems have different goals and scopes compared with broad-spectrum emergency triage *InTriage*, which is designed to serve general city residents in PEC, addressing diverse emergency scenarios without additional hardware dependencies.

Different Technologies. There are many different technologies are involved (He et al., 2025; Lin et al., 2025). In CV-based approaches, Lu et al. (2023) introduced an unmanned aerial vehicle (UAV) triage system using OpenPose and YOLO, highlighting the role of visual recognition. In ML-based solutions, Elhaj et al. (2023) conducted a comprehensive comparison of nine supervised algorithms to predict EDs outcomes, with Random Forest achieving the highest performance in ICU admission predictions, while Chen et al. (2023b) employed eXtreme Gradient Boosting (XGB) for dynamic risk stratification using ECG and chest X-ray data. In DL domain, Xiao et al. (2023) proposed TransNet and TextRNN models that integrate structured and unstructured medical data using attention mechanisms, whereas Chen et al. (2023a) developed a BiLSTM-based system that leverages clinical narratives to predict critical outcomes in EDs. In contrast, our *InTriage* system utilizes LLMs to dynamically process multimodal inputs, including real-time text, speech, and medical data, enabling more precise triage results and offering auxiliary functions like sentiment analysis and medical information retrieval to enhance decision-making in diverse emergency scenarios.

3 Preliminary Data

In TT systems, call-taker performance can vary significantly based on experience and training, leading to inconsistent triage results and potentially adverse outcomes. To address this challenge, *InTriage* utilizes the Patient Acuity Category Scale (PACS) protocol standardize the assessment process, which was introduced by Singapore's Ministry of Health (Fong et al., 2018). As shown in Table 1, PACS categorizes patients into one of five urgency levels, from P1+ (the most critical) to P4 (non-urgent). This structured approach minimizes variability in call-takers' decisions.

Under the PACS protocol, call-takers first conduct a preliminary assessment by asking general questions about the patient's information and symptoms. Based on this initial information, the case is classified into one of 30 Chief Incident Types (CITs), such as BLEEDING/LACERATION, INHALATION, or CHEST PAIN. After identifying the appropriate CIT, call-takers proceed to ask CIT-specific questions to gather more detailed informa-

PACS	Definition	Response	Example
P1+	Life-threatening emergencies	Highest priority, fastest response (extra resources deployed)	Cardiac arrest
P1		Highest priority, fastest response	Head injury
P2	Emergencies	High priority, fast response	Abdominal pain
P3	Minor Emergencies	Lower priority, slower response	Persistent Diarrhea
P4	Non-emergencies	No response	Cough

Table 1: Definitions of Patient Acuity Category Scale (PACS).

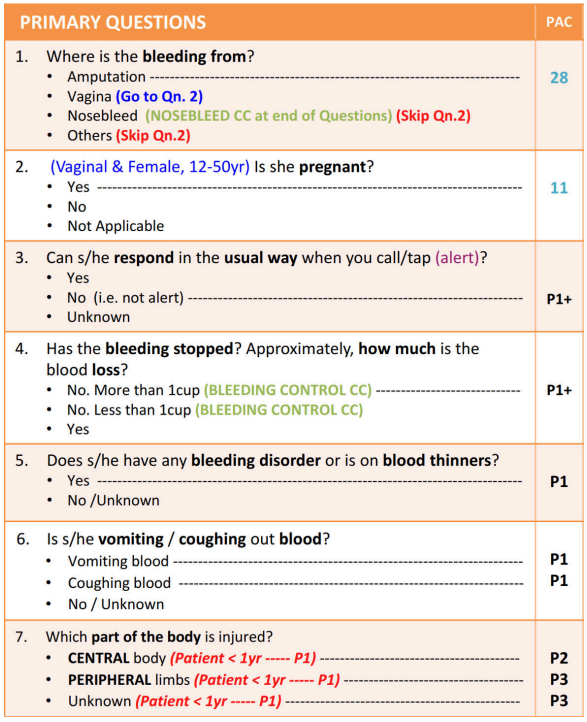


Figure 3: An example of BLEEDING/LACERATION incident type using PACS to guide call-takers for an emergency calls. The numbers in the PACS column indicate when it should be reassigned to another incident type, while P1+, P1, P2, and P3 represent triage levels.

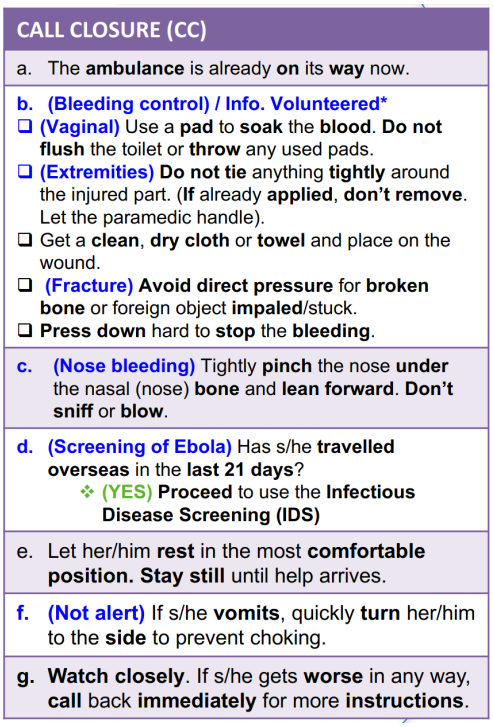


Figure 4: An example of BLEEDING/LACERATION incident type to guide call takers give necessary emergency instructions.

tion and assign the case to the appropriate PACS category for dispatch purposes. For example, in the case of BLEEDING/LACERATION, the call-taker would ask pre-defined questions such as, "Where is the bleeding from?" or "Has the bleeding stopped?" Figure 3 illustrates a sample decision tree for this specific CIT. Based on the caller's responses, the call-taker assigns the patient to a PACS category.

For assigning triage categories, call-takers must deliver CIT-specific instructions to callers to ensure the patient's safety while waiting for medical assistance. These instructions are tailored to each incident type, aiming to prevent deterioration and stabilize the patient. Figure 4 shows an example of instructions for BLEEDING/LACERATION, such as advising the caller to keep the patient still and elevate the injured area to reduce bleeding.

4 Different Perspectives from Caller, Call-taker, and InTriage

Caller Perspective – System Significance. For callers, they will no realize that any AI system is in place. As in Figure 2 (a), when they contact the emergency hotline, they are greeted by a human call-taker who responds professionally and promptly. They can also get emergency instructions, and obtain emotional comfort.

Call-taker Perspective – System Functions. For call-takers, our system provides an intelligent assistant that eliminates the need for additional manual operations to structure the content of current call. Besides, triage requires call-takers to follow strict guidelines, but the variety of emer-

gencies and question sets can overwhelm new staff and cause errors, even among experienced operators. To mitigate this issue, *InTriage* automatically recognizes the type of emergency and prompts the call-taker with the appropriate questions to ask. Additionally, our system can provide extra medical history to ensure call-takers do not overlook potential risks. Finally, *InTriage* recommends necessary emergency instructions and triage results with supporting evidence. This feature reduces the call-takers' workload and helps correct inappropriate triage decisions, ensuring better outcomes and increased efficiency in handling emergency calls.

InTriage Perspective – System Implementation. As shown in Figure 2 (b), *InTriage* consists of five agents of ASR, NLU, DM, KB, and NLG.

ASR Agent. Our ASR agent employs WeNet (Yao et al., 2021) as the backbone. We collect extra training data from audiobooks, podcasts, YouTube, and SCDF to further fine-tune WeNet, as shown in Appendix Table 3. One principle of data selection is that we have chosen audio that fits the specific accents of Singapore, including Singaporean-English, Mandarin-English, and Malay-English. At present, *InTriage* is multilingual, which can support English, Chinese, and Malay simultaneously.

NLU Agent. Our NLU agent utilizes Llama-3.2-1B. We manually annotate 2008 real cases from SCDF for training extracting key information from emergency calls. The annotated data are formatted as a QA task. For example, when an emergency call is going, the NLU agent will keep raising question like "What is the gender of the patient/caller?", "Where is the patient/caller?", or "How is the patient's mood? Describe the patient's mood and rate it on a scale of 1 to 10", NLU agent will answer the questions to automatically fill the slots.

DM Agent. Our DM agent performs three core functions. First, based on the incident type identified by the NLU agent, the DM agent provides real-time prompts to call-takers, guiding them on which questions to ask in accordance with SCDF's established protocols (see Figure 3). These prompts adjust dynamically as the NLU agent continuously refines its assessment of the incident type. Second, the DM agent retrieves data from the EHR system to obtain the caller's relevant medical history by matching extracted identity information. NLU agent further processes this data to extract key details for display. Finally, DM agent continuously calculates a confidence score. Each caller response that matches a predefined question from SCDF's protocols contributes to an incremental score.

Languages		Kaldi	Whisper	Ours	RTF
Singaporean-English	MSF Call Center	**18.25**	22.13	26.12	0.48
	MSF-DHL	26.53	24.12	**20.94**	0.42
	NTU Inhouse (1)	17.09	19.40	**10.31**	0.40
	NTU Inhouse (2)	31.60	25.85	**17.68**	0.48
	SCDF testset (1)	14.40	17.44	**10.60**	0.39
	SCDF testset (2)	28.25	24.98	**14.76**	0.46
	IMDA testset (1)	10.54	13.41	**8.60**	0.38
	IMDA testset (2)	21.95	17.43	**7.12**	0.34
	IMDA testset (2) Hotword Boosting	-	-	**4.56**	0.38
	SCDF-Mandarin	23.47	19.89	**15.29**	0.37
Malay-English	SCDF-Malay	24.16	22.65	**14.25**	0.34
	-	21.62	20.73	**14.57**	0.40

Table 2: The table compares word error rates across models. Kaldi (Povey et al., 2011) and Whisper (Radford et al., 2022) serve as baseline models. RTF (Real-Time Factor) is reported to assess our system's real-time performance on an RTX 4090 GPU.

KB Agent. KB Agent consists of an EHR database and predefined emergency instructions, which can be used for retrieval.

NLG Agent. When the score calculated by the DM agent exceeds the predefined threshold, the NLG agent generates the final triage decision and provides corresponding instructions. The NLG agent shares a common LLM with the NLU agent.

5 Evaluation and Showcases

To quantitatively evaluate the performance of the proposed *InTriage*, we conducted evaluations of both the ASR agent and NLU agent. Table 2 compares Kaldi, Whisper (small version, with larger size than ours) and our ASR agent across various cases, for Singaporean-English, Mandarin-English, and Malay-English speech recognition. All data are sampled from real-world data in Singapore. Our ASR agent outperforms baselines, showing significant improvements in word error rate, especially in Singaporean-English. Notably, the proposed model achieves a 7.12 error rate on the Imda (2) compared to Kaldi's 21.95 and further improves to 4.56 with Hotword Boosting. This result highlights the inherent challenge of accurately identifying locations in speech recognition. The model also performs better on SCDF-Mandarin and SCDF-Malay, highlighting its effectiveness in multilingual scenarios.

In Figure 6, we compare the accuracy of different caller's attributes between TANL (Paolini et al., 2021), LIama-3.1 8B, and our NLU agent. Overall, our method outperforms TANL across all attributes, with the most significant improvement

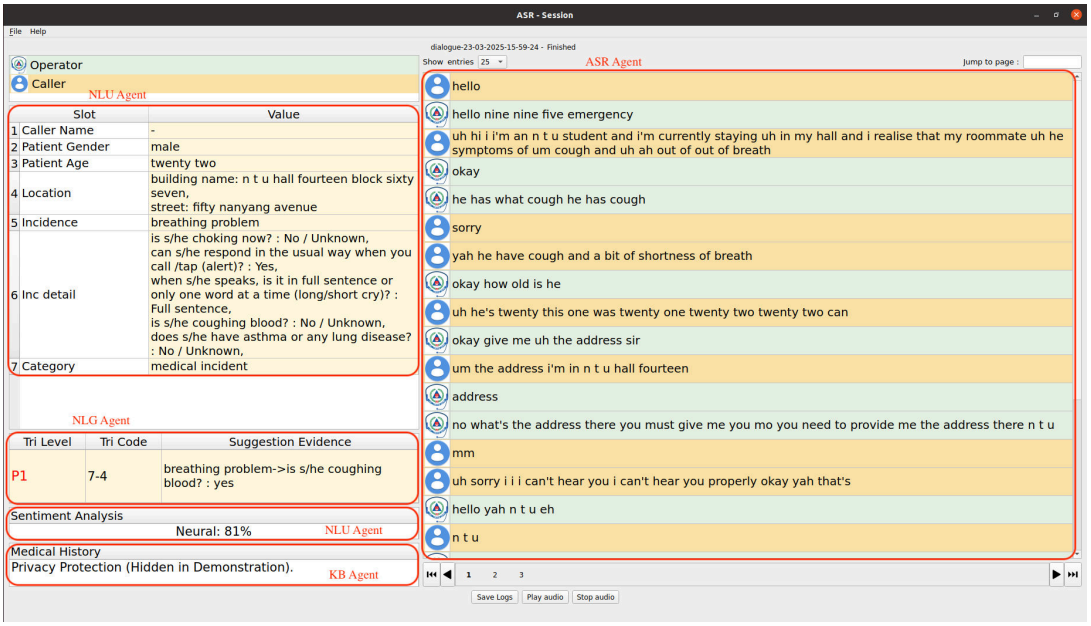


Figure 5: Screenshot of *InTriage* for an emergency case with P1 level. More show cases with different triage levels are in Appendix. All show cases in the paper use simulated conversations to ensure privacy.

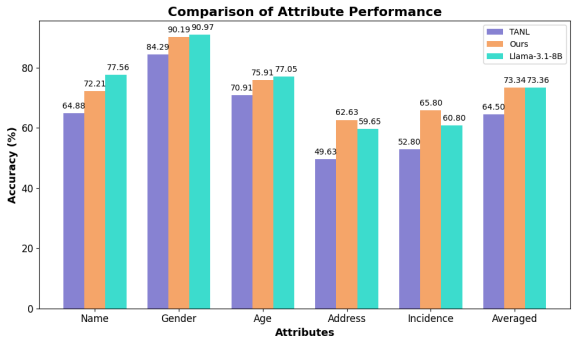


Figure 6: Breakdown analysis of key information extraction from emergency calls.

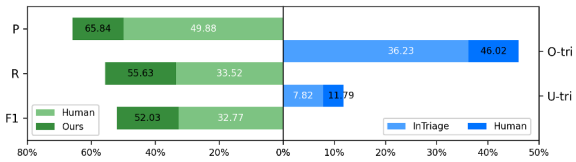


Figure 7: The triage performance comparison between InTriage with human call-taker. O-tri and U-tri indicate Over-triage and Under-triage. More detail breakdown results can be seen in appendix.

observed in the Gender attribute. The Address attribute shows the lowest performance for both methods, although our model achieves a notable improvement from 49.63% to 62.63%. Due to we take the extract matching as evaluation strategy, this performance for location recognition is satisfied. The average accuracy of our method reaches 73.34%, significantly surpassing TANL's 64.50%. This demonstrates the effectiveness of our approach across various attribute types, particularly in challenging tasks such as Incidence and Address prediction. LIama-3.1 8B performs slightly better than our NLU agent overall, but our NLU agent is only 1B in size and runs relatively faster. Considering real-time ability, our agent is more suitable when taking all factors into account. Figure 7 presents a comparison of triage performance between InTriage and human call-takers. The results show that InTriage achieves 19.25%, 22.11%, and 15.96% higher Precision, Recall, and F1-score, respectively, compared to human call-takers. Meanwhile, the rates of Over-triage and Under-triage are reduced by 9.79% and 3.97%, respectively. Figure 5 is a showcase with a P1 triage level of *InTriage*. More results, case studies, analysis and discussions can be seen in appendix.

6 Conclusion

This study presents *InTriage*, an AI-driven multilingual TT system for alleviating ineffectiveness in PEC triage. By automating call transcription, extracting information, and providing real-time triage decision support, *InTriage* improves accuracy, and enhances resource allocation in emergency settings. *InTriage* offers a scalable solution for modernizing TT, ensuring timely care for high-acuity cases and alleviating the burden on call-takers.

7 Ethics and Broader Impact Statement

This study was conducted in compliance with ethical guidelines to ensure the protection of participants' rights, privacy, and well-being. The research involved the collection, analysis, and use of anonymized data to improve triage protocols and enhance emergency response systems.

The data used in this study were fully anonymized before analysis to protect the privacy of individuals. No personally identifiable information was accessed or stored during the research process. Anonymization techniques ensured that participants could not be identified directly or indirectly from the data. Measures were taken to prevent potential biases in the data and to ensure that the AI system's recommendations adhered to the protocols established by the Singapore Civil Defence Force (SCDF). Regular monitoring and audits were implemented to assess the accuracy and fairness of the AI outputs. The study adhered to ethical standards regarding the use of AI technologies in healthcare. The AI system was designed to assist and augment human decision-making rather than replace it. Ethical safeguards were put in place to prevent harm and ensure the system's use aligns with SCDF's protocols.

Beyond ensuring privacy and data anonymization, we foresee two additional ethical risks that must be managed as InTriage scales: (1) over-reliance on automated recommendations and (2) inadvertent propagation of bias.

(1) Although our platform is designed to support, rather than replace human call-takers, the convenience brought by automation may gradually lead to over-reliance, potentially weakening call-takers' independent judgment and critical thinking. This risk is especially pronounced in emergency-care contexts when the system provide incorrect result. In our application scenarios, callers often speak incoherently under emotional distress and in noisy environments—conditions that may degrade ASR performance and further affect the system's overall reliability. Without appropriate safeguards, such over-reliance in high-stakes, unpredictable scenarios may compromise the accuracy and timeliness of medical responses, the very outcomes the system is intended to improve.

To curb over-reliance, we plan to clearly label all InTriage outputs as advisory only, reinforcing that the system is meant to support, not replace—human decision-making. Final clinical and dispatch decisions will remain the responsibility of trained call-takers. Furthermore, we aim to implement regular refresher training programs that help call-takers interpret model confidence scores, recognize edge cases where automation may be unreliable, and apply override rules when necessary. These training sessions will include scenario-based simulations that expose call-takers to challenging cases where the system might fail or produce uncertain outputs, thereby cultivating vigilance and situational awareness. Additionally, we intend to integrate real-time uncertainty indicators into the interface, such as low-confidence alerts, flagging ambiguous inputs, or highlighting cases with degraded ASR quality. These indicators will help users critically assess system recommendations rather than accept them at face value. Over time, we will also explore adaptive user interfaces that modulate the level of automation based on contextual reliability, for instance, reducing system assertiveness when acoustic conditions are poor or when user hesitation is detected. Ultimately, promoting a culture of active human oversight, where AI is seen as a collaborative tool rather than an infallible authority.

To counter potential biases from imbalanced training data, such as under-representation of non-standard accents or minority medical histories, we intend to conduct quarterly fairness audits across demographic subgroups and incident types. When disparities exceed predefined thresholds, we will apply targeted data augmentation or model fine-tuning. An internal ethics board comprising clinicians, technologists, and community representatives will review each audit and approve any algorithmic updates before deployment, ensuring continuous human oversight and accountability throughout the system's life cycle.

8 Acknowledgment

This project was funded by the National Research Foundation Singapore, Singapore under under AI Singapore Programme (Award Number: AISG2-TC-2022-004); by National University of Singapore, Saw Swee Hock School of Public Health, AI for Public Health (AI4PH) program, and The National Social Science Fund of China under Grant (23CWW006).

References

Christine Adler, Marion Krüsmann, Thomas Greiner-Mai, Anton Donner, Javier Mulero Chaves, and Àngels Via Estrem. 2011. It-supported management of mass casualty incidents: The e-triage project. In *Proceedings of the 8th International ISCRAM Conference*.

Mike Brady. 2020. Patient experiences of uk ambulance service telephone triage: a review of the literature. *International Journal of Emergency Services*, 9(2):89–108.

Min-Chen Chen, Ting-Yun Huang, Tzu-Ying Chen, Panchanit Boonyarat, and Yung-Chun Chang. 2023a. Clinical narrative-aware deep neural network for emergency department critical outcome prediction. *Journal of Biomedical Informatics*, 138:104284.

Yu-Hsuan Jamie Chen, Chin-Sheng Lin, Chin Lin, Dung-Jang Tsai, Wen-Hui Fang, Chia-Cheng Lee, Chih-Hung Wang, and Sy-Jou Chen. 2023b. An ai-enabled dynamic risk stratification for emergency department patients with ecg and cxr integration. *Journal of Medical Systems*, 47(1):81.

Hamza Elhaj, Nebil Achour, Marzia Hoque Tania, and Kurtulus Aciksari. 2023. A comparative study of supervised machine learning approaches to predict patient triage outcomes in hospital emergency departments. *Array*, 17:100281.

Ru Ying Fong, Wee Sern Sim Glen, Ahmad Khairil Mohamed Jamil, Wilson Wai San Tam, and Yanika Kowitlawakul. 2018. Comparison of the emergency severity index versus the patient acuity category scale in an emergency setting. *International Emergency Nursing*, 41:13–18.

Thomas Greiner-Mai and Anton Donner. 2010. Data management in mass casualty incidents: The e-triage project. In *INFORMATIK 2010 - 40. Jahrestagung der Gesellschaft für Informatik e.V.*, pages 200–206, Berlin, Germany.

Kai He, Rui Mao, Qika Lin, Yucheng Ruan, Xiang Lan, Mengling Feng, and Erik Cambria. 2025. A survey of large language models for healthcare: from data, technology, and applications to accountability and ethics. *Information Fusion*, 118:102963.

Kai He, Jialun Wu, Xiaoyong Ma, Chong Zhang, Ming Huang, Chen Li, and Lixia Yao. 2019. Extracting kinship from obituary to enhance electronic health records for genetic research. In *Proceedings of the Fourth social media mining for health applications (# SMM4H) workshop & shared task*, pages 1–10.

Thongpitak Huabbangyang, Rapeeporn Rojsaengroeng, Gawin Tiyawat, Agasak Silakoon, Alissara Vanichkulbodee, Jiraporn Sri-On, and Siriwimol Buathong. 2023. Associated factors of under and over-triage based on the emergency severity index; a retrospective cross-sectional study. *Archives of academic emergency medicine*, 11(1).

Ryota Inokuchi, Masao Iwagami, Yu Sun, Ayaka Sakamoto, and Nanako Tamiya. 2022. Machine learning models predicting undertriage in telephone triage. *Annals of Medicine*, 54(1):2989–2996.

Varada Vivek Khanna, Krishnaraj Chadaga, Niranjana Sampathila, Srikanth Prabhu, and Rajagopala Chadaga. 2023. A machine learning and explainable artificial intelligence triage-prediction system for covid-19. *Decision Analytics Journal*, 7:100246.

Qika Lin, Yifan Zhu, Xin Mei, Ling Huang, Jingying Ma, Kai He, Zhen Peng, Erik Cambria, and Mengling Feng. 2025. Has multimodal learning delivered universal intelligence in healthcare? a comprehensive survey. *Information Fusion*, 116:102795.

Jiafa Lu, Xin Wang, Linghao Chen, Xuedong Sun, Rui Li, Wanjing Zhong, Yajing Fu, Le Yang, Weixiang Liu, and Wei Han. 2023. Unmanned aerial vehicle based intelligent triage system in mass-casualty incidents using 5g and artificial intelligence. *World journal of emergency medicine*, 14(4):273.

Joshua R Lupton, Cynthia Davis-O'Reilly, Rebecca M Jungbauer, Craig D Newgard, Mary E Fallat, Joshua B Brown, N Clay Mann, Gregory J Jurkovich, Eileen Bulger, Mark L Gestring, et al. 2023. Undertriage and over-triage using the field triage guidelines for injured patients: a systematic review. *Prehospital Emergency Care*, 27(1):38–45.

Fateme Mohammadi, Ali Khani Jeihooni, Parisa Sabetsarvestani, Fozieh Abadi, and Mostafa Bijani. 2022. Exploring the challenges to telephone triage in prehospital emergency care: a qualitative content analysis. *BMC Health Services Research*, 22(1):1195.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Ma Jie, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, Stefano Soatto, et al. 2021. Structured prediction as translation between augmented natural languages. In *ICLR 2021-9th International Conference on Learning Representations*, pages 1–26. International Conference on Learning Representations, ICLR.

Ju Young Park. 2021. Real-time monitoring electronic triage tag system for improving survival rate in disaster-induced mass casualty incidents. In *Healthcare*, 7, page 877. MDPI.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *arXiv preprint*.

Singapore Civil Defence Force. 2023. Scdf annual statistics 2023. Accessed: 2024-12-19.

Andrew AS Soltan, Samaneh Kouchaki, Tingting Zhu, Dani Kiyasseh, Thomas Taylor, Zaamin B Hussain, Tim Peto, Andrew J Brent, David W Eyre, and David A Clifton. 2021. Rapid triage for covid-19 using routine clinical data for patients attending hospital: development and prospective validation of an artificial intelligence screening test. *The Lancet Digital Health*, 3(2):e78–e87.

Veronica Vicente, Maaret Castren, Fredrik Sjöstrand, and BIRGITTA WIREKLINT SUNDSTRÖM. 2013. Elderly patients' participation in emergency medical services when offered an alternative care pathway. *International Journal of Qualitative Studies on Health and well-being*, 8(1):20014.

Naythan Villafuerte, Santiago Manzano, Paulina Ayala, and Marcelo V García. 2023. Artificial intelligence in virtual telemedicine triage: A respiratory infection diagnosis tool with electronic measuring device. *Future Internet*, 15(7):227.

Jialun Wu, Xinyao Yu, Kai He, Zeyu Gao, and Tieliang Gong. 2024. Promise: A pre-trained knowledge-infused multimodal representation learning framework for medication recommendation. *Information Processing & Management*, 61(4):103758.

Yi Xiao, Jun Zhang, Cheng Chi, Yuqing Ma, and Aiguo Song. 2023. Criticality and clinical department prediction of ed patients using machine learning based on heterogeneous medical data. *Computers in Biology and Medicine*, 165:107390.

Guanrou Yang, Ziyang Ma, Zhifu Gao, Shiliang Zhang, and Xie Chen. 2024. Ctc-assisted llm-based contextual asr. *arXiv preprint arXiv:2411.06437*.

Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, Fan Yu, Chao Yang, Zhendong Peng, Xiaoyu Chen, Lei Xie, and Xin Lei. 2021. Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit. In *Proc. Interspeech*, Brno, Czech Republic. IEEE.

A Appendix

A.1 Integration Challenges

Although InTriage convincingly demonstrates strong technical merit, navigating the complexities of deployment within a tightly regulated emergency care ecosystem remains a significant challenge. The most significant friction point is likely to come from government policy governing call monitoring and data sovereignty: transcribed audio and extracted medical data must comply with government policies. These rules impose strict limits on where audio can be stored, how long recordings may be retained, and which agencies may access them. These constraints could delay real-time hand-offs between InTriage, national EHR repositories, and ambulance-dispatch consoles. Also, call-takers may resist automation that appears to "shadow" or audit their work, especially if they fear increased scrutiny or deskilling. Overcoming these hurdles will require early engagement with regulators to codify permissible data flows, the introduction of opt-in call-monitoring policies that clarify accountability, and change-management programmes that position InTriage as a decision-support ally rather than a replacement. Without such policy alignment and workforce buy-in, even a well-validated AI pipeline risks limited uptake or protracted pilot phases within emergency services.

A.2 Training data

Table 3 provides a comprehensive summary of the training data used to develop the ASR agent of InTriage. The dataset is sourced from four primary audio types: Audiobooks, Podcasts, YouTube, and SCDF. Each source is characterized by varying acoustic conditions to ensure robustness across different environments and speaker profiles.

The Audiobook dataset comprises 2,655 transcribed hours and a total of 11,982 hours, capturing a wide range of reading voices across various ages and accents. Podcasts contribute 3,498 transcribed hours and a total of 9,254 hours, reflecting diverse conditions, including clean audio, background music, indoor and near-field recordings, and spontaneous speech. The YouTube dataset provides 3,845 transcribed hours and a total of 11,768 hours, covering both clean and noisy environments, with indoor, outdoor, near-field, and far-field recordings, as well as reading and spontaneous speech. Lastly, SCDF data includes 500 transcribed hours, comprising three years of real emergency call data from over 360,000 cases, providing real-world emergency scenarios. This diverse collection of audio data, with various acoustic conditions and speaker demographics, was essential in creating a robust ASR system capable of handling a wide range of real-world emergency call situations. At present, only transcribed data is used as train data. We will further enhance the ASR agent by employing more data in the future.

Audio Source	Transcribed Hours	Total Hours	Acoustic Condition
Audiobook	2,655	11,982	Reading, Various ages and accents
Podcast	3,498	9,254	Clean or background music, Indoor, Near-field, Spontaneous, Various ages and accents
YouTube	3,845	11,768	Clean and noisy, Indoor and outdoor, Near- and far-field, Reading and spontaneous, Various ages and accents
SCDF	500	-	3-years of real data from SCDF, more than 360,000 cases.

Table 3: Summary of used training data for ASR agent of *InTriage*.

Label	Precision (%)	Recall (%)	F1 (%)	Over-triage Rate (%)	Under-triage Rate (%)
P1	18.82 / 28.27	65.31 / 77.14	29.22 / 41.38	0.00 / 0.00	34.69 / 22.86
P2	80.81 / 85.90	54.79 / 59.45	65.31 / 70.27	44.52 / 39.94	0.68 / 0.61
P3	50.00 / 83.33	6.45 / 31.25	11.43 / 45.45	93.55 / 68.75	0.00 / 0.00
Macro-Average	49.88 / 65.84	33.52 / 55.63	32.77 / 52.03	–	–
Micro-Average	52.15 / 60.23	45.01 / 60.09	45.32 / 60.16	–	–
Avg.				46.02 / 36.23	11.79 / 7.82

Table 4: Triage performance breakdown. Black text indicates human performance, while red text represents the performance of our InTriage system.

A.3 Evaluation data

Table 6 presents the various s used to evaluate the performance of the ASR system across different language varieties, primarily focusing on Singaporean-English, Mandarin-English, and Malay-English. The s were carefully selected to cover a range of real-world scenarios to ensure robustness in different emergency communication contexts.

For Singaporean-English, multiple s were employed, including data from the Ministry of Social and Family Development, Singapore (MSF), MSF-DHL Express, Singapore (DHL), and NTU Inhouse recordings. Additionally, s from the Singapore Civil Defence Force (SCDF) were used, comprising two distinct s (SCDF testset (1) and SCDF testset (2)), reflecting emergency call scenarios. The Infocomm Media Development Authority, Singapore (IMDA) testsets (1 and 2) were also included to assess performance in broader communication contexts. Notably, IMDA testset (2) Hotword Boosting was used to evaluate the ASR system's ability to recognize critical keywords and phrases in emergency situations.

For Mandarin-English and Malay-English, the s were sourced from SCDF, labeled as SCDF-Mandarin and SCDF-Malay respectively. These two datasets support language-mixing between Mandarin, Malay, and English languages. Under such conditions, it ensures that the ASR system is evaluated across a multilingual context with different accent, reflecting the linguistic diversity of Singapore's population and the need for multilingual support in emergency response systems. The comprehensive inclusion of these s highlights the ASR system's capacity to handle various accented English varieties and language-switched speech, ensuring reliable performance in multilingual, spontaneous, and formal communication settings.

Table 4 provides a detailed performance comparison between human call-takers and our proposed InTriage system across various triage categories. Overall, InTriage demonstrates substantial improvements in critical performance metrics. For instance, in category P1, our system notably increases precision from 18.82% to 28.27% and recall from 65.31% to 77.14%, resulting in an improvement in the F1 score from 29.22% to 41.38%. Even more pronounced improvements are observed in category P3, with precision significantly rising from 50.00% to 83.33%, recall increasing from 6.45% to 31.25%, and consequently, the F1 score markedly improving from 11.43% to 45.45%.

The results further indicate that while both human call-takers and InTriage exhibit relatively low under-triage rates, over-triage remains a notable challenge. This primarily stems from the conservative, safety-first approach generally adopted by human call-takers, as evidenced by the particularly high over-triage rates observed in categories P2 and P3. Even in cases not clearly urgent, human call-

<table>
<tr><th colspan="6">Case 1</th></tr>
<tr><td>ASR
(part results)</td><td colspan="5">...
Operator: oh did she happen to choke on anything
Caller: oh, no i am not sure
Operator: uh if she speaks to you is it in full sentence or word by word
Caller: half half yup correct
Operator: can I say word by word
Caller: ya just word by word
...</td></tr>
<tr><td rowspan="2">NLU</td><td>Name</td><td>Gender</td><td>Age</td><td>Location</td><td>Incidence</td></tr>
<tr><td>[mask]</td><td>female</td><td>66</td><td>[mask]</td><td>breathing
problem</td></tr>
<tr><td>Triage</td><td>P1+</td><td>SA</td><td>neural</td><td colspan="2"></td></tr>
<tr><td>Evidence</td><td colspan="5">breathing problem → when s/he speaks, is it in full sentence or only one word at a time ? : one/few words</td></tr>
<tr><td rowspan="5">Details</td><td colspan="5">is s/he choking now? : no (unknown)</td></tr>
<tr><td colspan="5">can s/he respond in the usual way when you call /tap (alert)? : not given</td></tr>
<tr><td colspan="5">when s/he speaks, is it in full sentence or only one word at a time (long/short cry)? : One/few words</td></tr>
<tr><td colspan="5">is s/he coughing blood? : not given</td></tr>
<tr><td colspan="5">does s/he have asthma or any lung disease? : not given</td></tr>
<tr><th colspan="6">Case 2</th></tr>
<tr><td>ASR
(part results)</td><td colspan="5">...
Operator: is there any blood
Caller: no (oh), on my head, i feel very painful
Operator: if anyone trapped in the car
Caller: no no one trapped in the car
...</td></tr>
<tr><td rowspan="2">NLU</td><td>Name</td><td>Gender</td><td>Age</td><td>Location</td><td>Incidence</td></tr>
<tr><td>[mask]</td><td>-</td><td>-</td><td>[mask]</td><td>motor
vehicle
accident</td></tr>
<tr><td>Triage</td><td>P3 (P2)</td><td>SA</td><td>anxious</td><td colspan="2"></td></tr>
<tr><td>Evidence</td><td colspan="5">motor vehicle accident-> is there any blood? : no</td></tr>
<tr><td rowspan="6">Details</td><td colspan="5">what vehicle(s)is/are involved? : vehicle</td></tr>
<tr><td colspan="5">is anyone trapped (pinned) in/under the vehicle ? : no</td></tr>
<tr><td colspan="5">was anyone thrown (flung) from the vehicle?: not given</td></tr>
<tr><td colspan="5">is everyone involved in the accident able to walk (alert)?: not given</td></tr>
<tr><td colspan="5">are there any obvious injuries? : not given</td></tr>
<tr><td colspan="5">is there any blood? : no (yes)</td></tr>
</table>

Table 5: Case study of two flawed examples. Erroneous elements are highlighted in red, and corrections are provided within brown parentheses.

takers tend to dispatch ambulances as a precaution, leading to an extremely high over-triage rate of up to 93.55% in category P3. In contrast, InTriage, trained using hospital-derived goal triage labels, is less influenced by subjective biases. We anticipate that InTriage will provide additional objective guidance to assist human call-takers in reducing over-triage rates, thereby optimizing ambulance resource utilization.

A.4 Cases studies

Table 5 presents two representative error cases that illustrate the different ways in which component-level failures occur.

In Case 1, the ASR module accurately transcribes all salient utterances, allowing the evidence extractor to trigger on the critical "breathing prob-

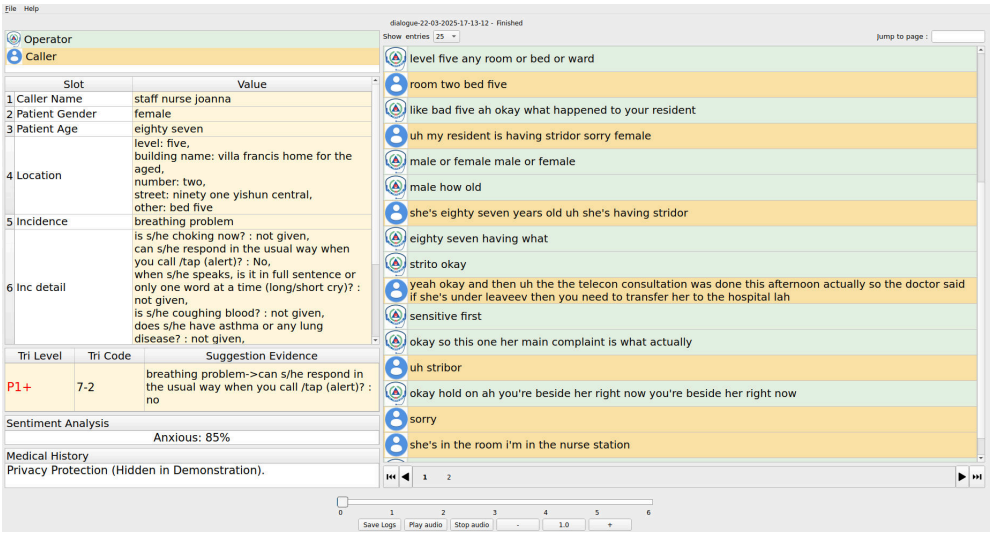


Figure 8: Screenshot of *InTriage* for an emergency case with P1+ level.

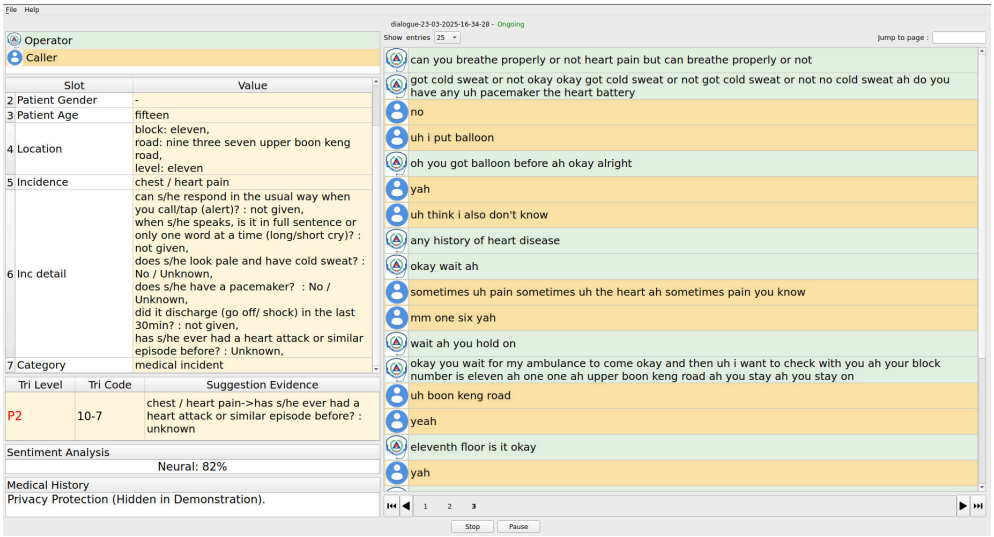


Figure 9: Screenshot of *InTriage* for an emergency case with P2 level.

lem" pattern. However, the NLU agent misidentifies "no i am not sure" as "no" (highlighted in red; correct value shown in brown parentheses). Because the caller's description is somewhat ambiguous and includes the literal expression "no," the error is understandable. Nevertheless, it did not affect the accuracy of the final triage outcome. This example highlights the system's robustness to NLU errors in non-critical fields when key clinical triggers are correctly identified.

In Case 2, the caller confirms the presence of bleeding after a motor-vehicle collision, but ASR mistranscribes the response as "no" instead of "oh". This may be due to the similar pronunciation of the two words in a noisy background. Consequently, NLU extracts the feature no bleeding, and the inference module downgrades the incident from the correct P2 (potentially life-threatening) to P3 (less urgent). This error highlights how a single misrecognized word can sometimes lead to an incorrect final triage outcome. As such, there is still significant room for further research in this area. This also underscores why we position our system as decision support rather than decision making.

A.5 More show cases

Figure 8, 9, 10 provide three examples with correct results for P2, P3, and P4 triage levels. Based on the different triage levels, the call-taker will make the final decision on whether to dispatch an ambulance and determine the appropriate type of ambulance to be dispatched.

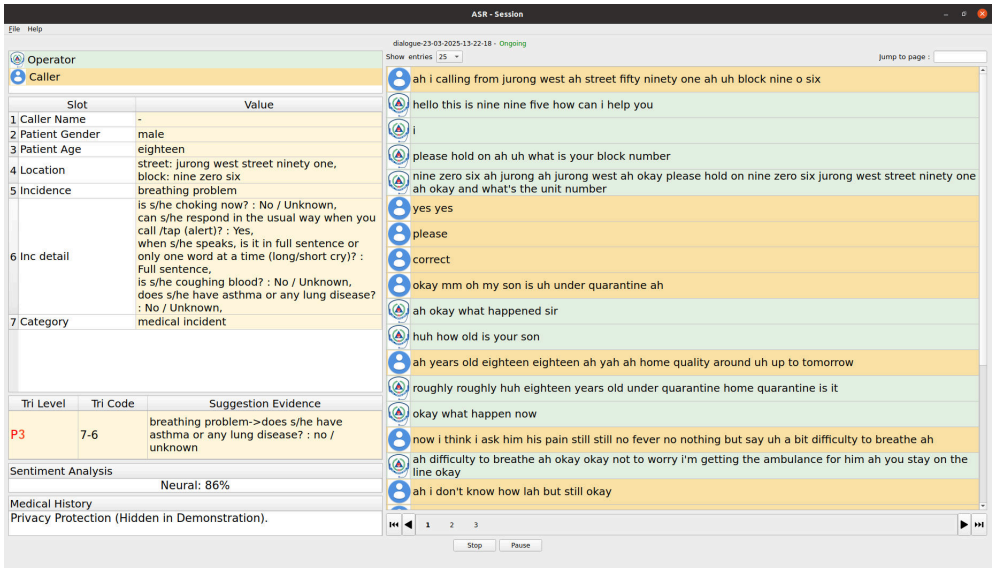


Figure 10: Screenshot of *InTriage* for an emergency case with P3 level.

From Behavioral Performance to Internal Competence: Interpreting Vision-Language Models with VLM-Lens

Hala Sheta[1,2,*] **Eric Huang**[1,5,*] **Shuyu Wu**[3,*]
Ilia Alenabi[1] **Jiajun Hong**[4] **Ryker Lin**[1] **Ruoxi Ning**[1,2] **Daniel Wei**[1] **Jialin Yang**[1,6]
Jiawei Zhou[4] **Ziqiao Ma**[3] **Freda Shi**[1,2]
[1]University of Waterloo [2]Vector Institute [3]University of Michigan
[4]Stony Brook University [5]McGill University [6]Dartmouth College
https://github.com/compling-wat/vlm-lens

Abstract

We introduce VLM-Lens, a toolkit designed to enable systematic benchmarking, analysis, and interpretation of vision-language models (VLMs) by supporting the extraction of intermediate outputs from any layer during the forward pass of open-source VLMs. VLM-Lens provides a unified, YAML-configurable interface that abstracts away model-specific complexities and supports user-friendly operation across diverse VLMs. It currently supports 16 state-of-the-art base VLMs and their over 30 variants, and is extensible to accommodate new models without changing the core logic.

The toolkit integrates easily with various interpretability and analysis methods. We demonstrate its usage with two simple analytical experiments, revealing systematic differences in the hidden representations of VLMs across layers and target concepts. VLM-Lens is released as an open-sourced project to accelerate community efforts in understanding and improving VLMs.

1 Introduction

Vision-language models (VLMs; Kirillov et al., 2023; Radford et al., 2021; Li et al., 2022; Liu et al., 2023; Wang et al., 2024a, *inter alia*) have become essential across a wide range of applications, including multimodal understanding (Yue et al., 2024), robotics (Li et al., 2024), and world modeling (Gao et al., 2025). However, existing VLM benchmarks predominantly adopt exact-match based accuracy and its derivations to evaluate model performance (Lin et al., 2014; Johnson et al., 2017; Yue et al., 2024; Fu et al., 2024, *inter alia*), which may either overlook the information embedded in their hidden representations (Zhang et al., 2025) or yield misleading assessments due to shortcut exploitation (Xu et al., 2025). Currently, there lacks a unified framework for extracting the internal representations of VLMs, making it challenging to assess model capabilities that go beyond simple performance evaluations.

Meanwhile, interpretability research and toolkits for VLMs remain underdeveloped compared to their text-only counterparts (Nanda and Bloom, 2022; Belrose et al., 2023; Ali et al., 2025, *inter alia*), posing significant challenges to systematically understanding their internal knowledge and decision-making processes. To the best of our knowledge, extending existing interpretability toolkits, such as TransformerLens (Nanda and Bloom, 2022), to support VLMs requires substantial engineering effort, as these tools are primarily designed for text-based Transformers.

To address these challenges in both benchmarking and interpretability, we present **VLM-Lens** (Figure 1), a toolkit that enables easy extraction of VLM intermediate output from any layer in a forward pass. The key features include:

- **Unified interface.** It abstracts out the model-specific setup and preprocessing complexities, allowing operations across models through a unified interface. Users can specify custom configurations via a YAML file with minimal boilerplate code provided, and the toolkit automatically handles model loading, preprocessing, and inference.
- **Model-specific environmental support.** Different VLMs often require different, and sometimes mutually conflicting, libraries. To address this issue, VLM-Lens provides model-specific environment setups, each of which can be easily installed with a single-line pip install command. A rigorous code review process ensures the consistency and reproducibility of the environment setups across different platforms.
- **Extensive model coverage and flexible nature.** The toolkit supports a diverse set of state-of-the-art VLMs, spanning widely used open-source models to recently developed, less-documented ones. Currently, VLM-Lens supports 16 base

[*] Equal contribution.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 886–895
November 4-9, 2025

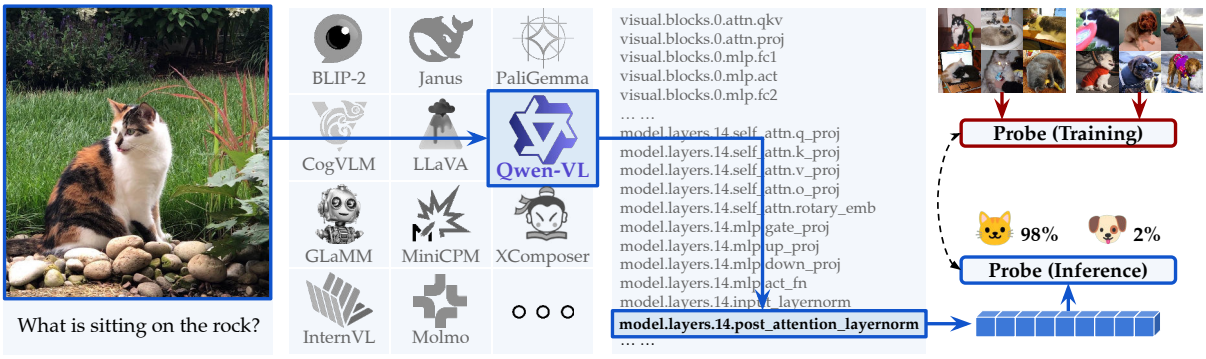


Figure 1: An example use case of VLM-LENS, where intermediate output from Qwen2-VL (Wang et al., 2024a) is extracted for probing.

VLMs and over 30 variants, with a highly extensible design that allows users or contributors to add new models with minimal effort.

VLM-LENS streamlines analytical tasks for VLMs, such as probing (Ettinger et al., 2016), neural circuit inspection (Chughtai et al., 2023), and knowledge tracing (Basu et al., 2024), as well as diagnosing model capabilities and limitations (Zhang et al., 2025; Stevens et al., 2025). As such, we anticipate that the toolkit will enable researchers and practitioners to conduct more fine-grained and rigorously controlled analyses of VLMs. The toolkit is released under the Apache-2.0 license.

2 Related Work

Vision-language models. Since vision and text naturally convey information about the world in two complementary modalities, there has been sustained interest in integrating them within unified frameworks (Kiros et al., 2014; Radford et al., 2021; Liu et al., 2023, *inter alia*). Earlier work primarily encoded images and text in a shared embedding space to facilitate efficient retrieval and matching (Kiros et al., 2014; Faghri et al., 2018; Radford et al., 2021). With recent advances in generative text models, exemplified by Brown et al. (2020), the focus has shifted toward building large-scale VLMs that generate text conditioned on both images and textual prompts (Liu et al., 2023; Wang et al., 2024a; Zhang et al., 2023, *inter alia*). While some models offer straightforward hidden-state extraction through open-sourced libraries such as HuggingFace Transformers (Wolf et al., 2020),[1] many require insufficiently documented customizations. Additionally, there is no systematic support for extracting intermediate representations beyond the layer-wise output, such as attention maps and intermediate features before layer normalization (Ba et al., 2016). VLM-LENS addresses this gap by providing a structured and unified interface for extracting intermediate output across many VLMs, thereby enabling detailed analyses of models.

Performance and competence analysis of VLMs. Efforts have been made to benchmark the performance of VLMs on various tasks, such as visual question answering (Johnson et al., 2017), image captioning (Lin et al., 2014) and general cross-modal understanding (Yue et al., 2024; Fu et al., 2024). These benchmarks largely assess models performance through exact-match based accuracy. However, accuracies fall short of capturing the full spectrum of model *competence*, which encompasses the internal mechanisms and generalizable knowledge that a model possesses, possibly beyond its observable textual output. To address this issue, recent work has started to explore the competence of VLMs through more fine-grained analyses via hidden states (Stevens et al., 2025) or output probability (Zhang et al., 2025). As such, VLM-LENS offers a toolkit for localizing hidden causal mechanisms in VLM, enabling convenient and flexible model competence analysis and assessing beyond simple accuracy-based evaluations.

Transformer interpretability toolkits. With increasing interest in interpreting Transformers, there have been various toolkits supporting their interpretation and analysis (Clark et al., 2019; Nanda and Bloom, 2022; Belrose et al., 2023; Ali et al., 2025, *inter alia*). However, most existing generic toolkits focus on text-only (e.g., Nanda and Bloom, 2022) or vision-only (e.g., Joseph et al., 2025) Transform-

[1] For example, by setting `output_hidden_state=True` in the forward function of LLaVA-1.5-7B (Liu et al., 2023); see the documentation at https://huggingface.co/docs/transformers/model_doc/llava.

ers, while the VLM counterparts significantly fall behind.[2] To the best of our knowledge, all existing VLM interpretability toolkits (Palit et al., 2023; Ben Melech Stan et al., 2024; Neo et al., 2024) support no more than a single model, and the designs are not easily extensible to support other models. To bridge this gap, VLM-LENS provides a unified framework to extract internal representations of VLMs, which can be coupled with existing interpretability methods (Thrush et al., 2022; Zhang et al., 2024; Basu et al., 2024; Stevens et al., 2025, *inter alia*) to analyze and assess a wide range of state-of-the-art VLMs.

3 VLM-Lens

At a high level, the VLM-LENS toolkit incorporates the hook mechanism of PyTorch (Paszke et al., 2019) to extract the internal representations of VLMs, which are then stored in a database for further analysis. We detail the design philosophy and key implementation aspects as follows.

Design philosophy. The design of VLM-LENS is driven by the need for a simple yet flexible toolkit to extract internal representations while accommodating the diverse dependency requirements of different VLMs. This is implemented through a central interface (`src/main.py`) that accepts model-specific configuration files in YAML format, along with dedicated environment setups for each model. Each model-specific implementation inherits from the base class, which standardizes model loading, preprocessing, and inference, while still allowing for model-specific customizations. The extracted intermediate representations are stored in a database with a standardized schema, enabling efficient retrieval and analysis. This design, executed by rigorous peer code review, ensures high extensibility: to support a new model, developers only need to implement a new model-specific class without modifying the core logic of the toolkit.

Supported VLMs. We currently support: Aya-Vision (Dash et al., 2025), Blip-2 (Li et al., 2023), CLIP (Radford et al., 2021), CogVLM (Wang et al., 2024b), GLaMM (Rasheed et al., 2024), InternLM-XComposer (Zhang et al., 2023), InternVL (Chen et al., 2024), Janus (Wu et al., 2025), LLaVA (Liu et al., 2023), MiniCPM-o (Team, 2025) and MiniCPM-V-2 (Yao et al., 2024), Molmo (Deitke et al., 2025), Paligemma (Beyer et al., 2024), Pixtral (Agrawal et al., 2024), PerceptionLM (Cho et al., 2025) and Qwen2-VL (Wang et al., 2024a). The toolkit supports all variants of these models across sizes, with the only requirement being sufficient hardware resources to load the model.

Configuration files. VLM-LENS allows users to specify model configurations, input and output data paths, model layers of interest, and other model-specific parameters through YAML configuration files. Users may extend the configuration files to include additional parameters for their experiments. No hard-coded parameters are used throughout the codebase. As an example, the following file (`configs/models/blip2/blip2.yaml`) specifies the required parameters for extracting the output of layers `vision_model.post_layernorm` and `language_model.lm_head` in `Blip2-OPT-2.7B` (Li et al., 2023):

```yaml
architecture: blip2
model_path: Salesforce/blip2-opt-2.7b
model:
  - torch_dtype: auto
output_db: output/blip2.db
input_dir: ./data/test-images/
prompt: "Describe the color in this image in one word."
modules:
  - language_model.lm_head
  - vision_model.post_layernorm
```

The command `python src/main.py --config configs/models/blip2/blip2.yaml` will provide the user with the corresponding intermediate output tensors in `output/blip2.db`, a SQLite3 database that can be further queried for analysis.

Pipeline implementation and output database organization. To initialize a model, an accompanying preprocessor is required to process the input images and format the prompt: for example, the chat template for LLaVA-1.5-7B (Liu et al., 2023) and Qwen2-VL (Wang et al., 2024a) is implemented in the preprocessor. The hidden representation extraction process involves registering a forward hook, a callable function that provides access to the input and output tensors of a layer specified by the user. It takes the preprocessed image and prompt as input, and saves the output tensors in a database following the forward pass. All forward hooks are unregistered to prevent contamination in later iterations. A list of possible layers (i.e., modules in PyTorch) of a particular model can be returned easily using the `--log-named-modules` parser argument, if the user requires more information on a model's internal structure. Lastly, the extracted tensors are stored in a database with the following attributes:

[2] For example, TransformerLens (Nanda and Bloom, 2022) can be patched to support vision analysis, but requires a non-trivial setup and is rigid in extending functionality.

`name`, `architecture`, `image_path`, `prompt`, `label`, `layer`, `tensor_dim`, `tensor`. These attributes correspond to the model HuggingFace identifier (e.g., `Salesforce/blip2-opt-2.7b`), model architecture specified in this toolkit (e.g., `blip2`), image path, textual prompt content, label of the example (if applicable), layer name (e.g., `language_model.lm_head`), the dimensionality of the extracted tensor, and the extracted intermediate result tensor itself, respectively.

While we provide a default preprocessor and a hooked forward pass implementation in `src/models/base.py` that can handle simple cases (e.g., LLaVA), these functions can be overridden to accommodate model-specific requirements (see more examples in `src/models`).

4 Usage Example I: Probing

4.1 Experimental Setups

We first demonstrate the usage of VLM-LENS by probing (Ettinger et al., 2016), where we train a set of probes on the extracted representations, evaluating the internal competence of VLMs on recognizing a set of primitive concepts.

Dataset. We create our dataset using CLEVR (Johnson et al., 2017). As depicted in Table 1, we define five categorical splits: `color`, `material`, `number`, `shape`, and `size`, each corresponding to a primitive object attribute in the images, as well as a `boolean` split that may cover any attribute with binary questions. Each split can be considered as a c-way classification task, where c represents the number of possible choices in the split. For each split, 80% data is used for training the probe, whereas the remaining 20% is used for testing.[3]

Probing approach. Each combination of split D, model $\mathcal{M}$, and layer ℓ is considered independently in the probing process. For an example $(e_i, y_i) \in D$, where e_i represents the input (i.e., image and text) of the task and y_i stands for its ground-truth prediction category, we extract the intermediate output of model $\mathcal{M}$ at the layer ℓ with mean pooling across the input tokens, denoted as $\mathcal{M}_\ell(e_i) \in \mathbb{R}^d$, where d is the dimensionality of layer ℓ. $\mathcal{M}_\ell(e_i)$ is used as the feature of e_i to train a probe that best predicts the y_i label.

We employ a two-layer perceptron as the probe, with ReLU activations and 512 hidden units.[4] A k-fold cross-validation is conducted to search for the best optimization hyperparameters for each split, using matching-based accuracy as the metric. We then train the probe model on the full training set using the best optimization hyperparameters, and report the accuracy on the test set. Following Hewitt and Liang (2019), we complement our main probe (trained to predict the true ground-truth labels) with a control probe (trained on randomly shuffled labels). A statistically significant advantage of the main probe over the control probe suggests that the VLM encodes task-relevant information, whereas no advantage implies that the probe relies on its own capacity to memorize spurious patterns.

Split	Question	Answer
`boolean`	Are there any other things that are the same size as the brown object?	yes
`color`	There is a small cylinder that is made of the same material as the big brown thing; what color is it?	green
`material`	What is the material of the big object that is the same color as the small metal cylinder?	metal
`number`	What number of things are brown blocks or green metallic cylinders that are to the right of the tiny cylinder?	2
`shape`	The big brown shiny thing is in what shape?	cube
`size`	There is another thing that is the same shape as the brown metallic object; what is its size?	small

Table 1: Examples from our probing dataset. We input the image and the question to the VLM, extract hidden states, and train a lightweight probe to decode the answer from these representations.

4.2 Results and Discussion

We evaluate the middle (i.e., $\lceil \frac{L}{2} \rceil$) and last layers of eight supported models, where L is the total number of layers in the model of interest (Figure 2). Notably, the probes trained on the Qwen-7b and MiniCPM-o representations achieve an almost perfect accuracy within many dataset splits, with a statistically significant difference from the control probe performance. This effect is especially preva-

[3] Data available at `https://huggingface.co/compling`.

[4] We searched for the probe hidden size with small-scale experiments, and found 512 to be the hidden size with the best performance consistency.

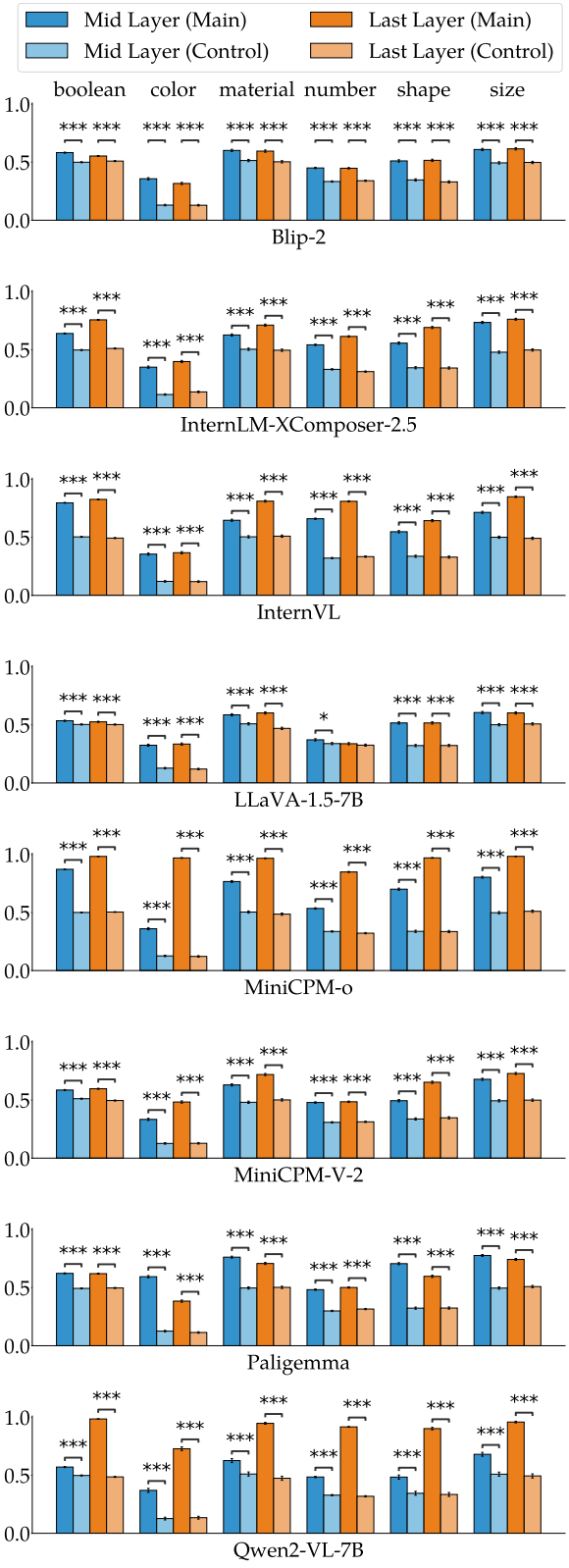


Figure 2: Evaluation Accuracy on our probing dataset by model, layer, and split. *Main* refers to probing on the regular data, while *control* stands for probing using data with random labels. The number of asterisks represents the significance level of the Z-test for Bernoulli variables (***: $p = .001$, **: $p = .01$, *: $p = .05$).

lent in the results of the last layer representations with a few exceptions (e.g., Blip-2 and Paligemma). In contrast, models like LLaVA-1.5-7B demonstrate a minor (albeit statistically significant in most cases) difference between the evaluation results in the original and control conditions, indicating weaker competence.

Across all models, the `color` attribute appears to be the most salient feature, with the main probe performance significantly better than the control results across both layers. Models with instruction-following and multimodal understanding capabilities, such as Qwen2-VL, MiniCPM-o, and InternVL, performed well on the more difficult splits such as `material`, `number`, and `shape`, especially when using the last-layer representations. This probing-based competence evaluation complements existing accuracy-driven benchmarks by providing a more detailed understanding of what is represented in the internal states, grounding the model performance in interpretable primitive knowledge.

5 Usage Example II: Concept Similarity

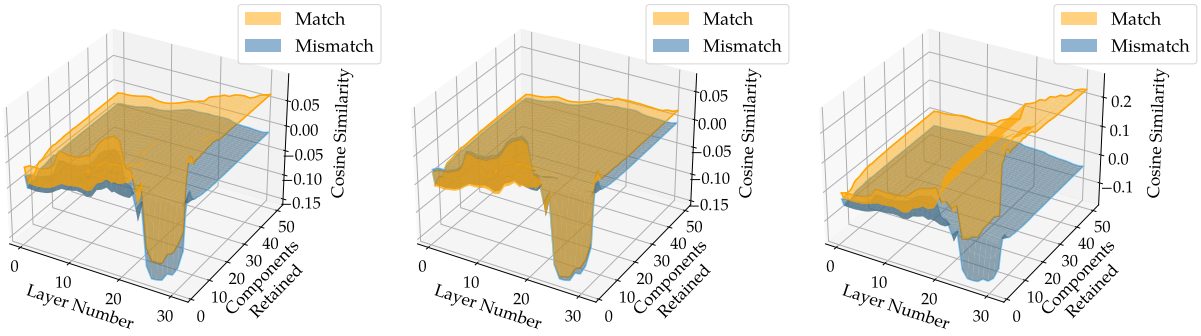
The second experiment is inspired by the classic *Stroop Effect* (Stroop, 1935), which demonstrates that humans exhibit slower and more error-prone responses when asked to name the font color of a word that is itself an incongruent color term (e.g., the word red printed in blue ink). We adapt this paradigm to VLMs by constructing images with deliberate incongruities between three color cues (Figure 4): the lexical word (e.g., white), the font color (e.g., yellow), and the background color (e.g., blue). This design probes how VLMs ground the notion of color under ambiguous instructions.

5.1 Experimental Setups

Prompt setup. Unlike humans in the Stroop task, who are explicitly instructed to name the ink color, we query the model with an intentionally ambiguous prompt: *Describe the color in this image in one word.*, coupled with a single image with incongruent cues. This allows us to study which representation (lexical, font, or background) the model privileges in its internal embeddings.

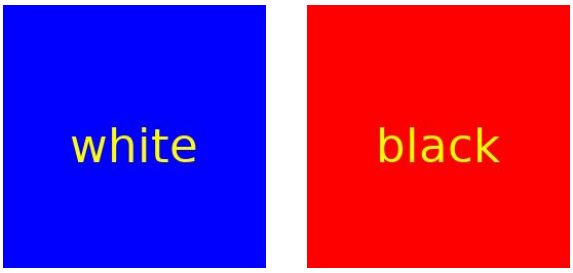
Prototype construction. To establish references for primitive color concepts (e.g., red, blue, green), we retrieve the top 10 de-duplicated Creative Commons–licensed images using the Google Images API.[5] We extract the intermediate layer rep-

[5] https://developers.google.com/photos



(a) Lexical content vs. reference. (b) Foreground font color vs. reference. (c) Background color vs. reference.

Figure 3: Cosine similarity between Stroop task images and primitive color concepts. Results are shown as a function of model layer (x-axis) and number of PCA components retained (y-axis), with orange surfaces indicating matching conditions and blue surfaces indicating mismatching conditions when considering different aspects.



(a) Word *white* written in yellow on blue canvas. (b) Word *black* written in yellow on red canvas.

Figure 4: Example images used in the Stroop Task.

resentations of these images coupled with the ambiguous textual instruction using VLM-LENS.

Similarity-based analysis. We apply Principal Component Analysis (PCA) to identify the directions that capture the most color variation. Let the zero-meaned embeddings be denoted as $E \in \mathbb{R}^{n \times d}$, where n is the number of reference images corresponding to primitive color concepts, and d is the dimensionality of the intermediate layer output. PCA learns a linear projection $W \in \mathbb{R}^{d \times d'}$ that maps E to a lower-dimensional representation $E' = EW \in \mathbb{R}^{n \times d'} (d' \ll d)$.

We evaluate the Stroop task images under different numbers of retained principal components: for one image at a specific model layer, we extract the corresponding hidden representation, projecting it into the transformed space with W, and compute the average cosine similarity with the reference color concepts. For lexical, font, and background colors, we report the average matched and mismatched cosine similarities across layers.

5.2 Results and Discussion

We visualize the results for LLaVA-1.5-7B (Figure 3), where a larger gap between matched and mismatched data indicates a more prominent feature in the Stroop task. We find a clear separation between the match and mismatch conditions across all three settings, indicating all three types of information (i.e., lexical content, foreground font color, and background color) are reliably encoded in the model. As expected, background color (Figure 3c) produces the strongest contrast between matched and mismatched examples. However, somewhat surprisingly, color presented as lexical content is more prominent than font color, evidenced by the gap in Figure 3a than that in Figure 3b. In addition, all concepts require a sufficient number of PCA components to achieve a clear separation, suggesting that color information is not captured in a single linear direction in the representation space.

6 System Evaluation

We evaluate VLM-LENS, in terms of time and memory efficiency, on the inference procedure of our supported models, using a subset of the MSCOCO validation set (Lin et al., 2014)[6] with 2,690 examples (Table 2). All experiments are done on a single NVIDIA-A40 GPU with sufficient CPU memory, using an inference batch size of 1. Users may use the reported performance statistics for informed choices on GPU selection and advanced inference techniques.

6.1 Inference Time

The model inference times are calculated for the duration of the forward inference on the dataset, including the database saving execution, and disregarding the model and processor loading time. As expected, CLIP is the fastest despite its high precision point, which is a result of its small number

[6] https://huggingface.co/datasets/compling/coco-val2017-obj-qa-categories

Model	# Params	Precision	Peak Mem (MB)	Inference Time (seconds)	Per-Instance Time (seconds)
CLIP (2021)	150M	`float32`	616.69	135	0.025
Blip-2 (2023)	2.7B	`float32`	15,261.28	295	0.055
InternLM-XComposer-2.5 (2023)	7B	`bfloat16`	24,037.64	3,056	0.569
InternVL (2024)	8B	`bfloat16`	21,136.92	3,347	0.623
LLaVA-1.5 (2023)	7B	`float16`	29,031.55	1,566	0.291
MiniCPM-V-2 (2024)	2.8B	`bfloat16`	7,154.23	495	0.092
MiniCPM-o (2025)	8B	`bfloat16`	18,058.20	671	0.125
Molmo (2025)	7B	`float32`	34,403.26	2,841	0.529
Paligemma (2024)	3B	`float32`	12,018.30	554	0.103
Qwen2-VL (2024a)	7B	`bfloat16`	33,840.66	1,497	0.279

Table 2: Inference properties of different models in VLM-LENS after execution on the MSCOCO (Lin et al., 2014) dataset, including GPU memory usage and inference time metrics.

of parameters. It is worth noting that CLIP is also the only model trained using contrastive image-text matching, rather than incorporating visual tokens alongside text, and, therefore, is not directly comparable to other models. InternVL, InternLM-XComposer-2.5, and MolMo are the slowest models, with high per-inference times compared to other models with similar parameter counts (7B). Furthermore, within the same parameter count and precision point, InternVL and MiniCPM-o demonstrate a disparate difference in inference time (0.623s vs 0.125s per-inference, respectively), which is likely due to differing architecture optimizations and input processing methods.

6.2 Memory Usage

To approximate the memory usage of each model using our toolkit, we record the precision and peak GPU memory usage (in MB; Table 2). Similar to inference time, CLIP demonstrates the lowest memory footprint (617MB), due to its compact architecture. In contrast, LLaVA-1.5 and Qwen2-VL use the most memory, despite having a low precision point and the same parameter count as many other models. In general, `bfloat16` and `float16` precision points reduce memory usage, but their effectiveness varies depending on the architecture: InternLM-XComposer and Qwen2-VL still demonstrate a large memory footprint (>24GB).

7 Conclusion and Discussion

In this paper, we introduce VLM-LENS, a toolkit that aims to benchmark, analyze, and interpret VLMs systematically. We demonstrate that the toolkit enables the assessment of the internal competence of a wide range of VLMs (§ 4), going beyond the simple accuracy-based evaluations provided by most existing benchmarks (Yue et al., 2024, *inter alia*). We offer performance statistics (Table 2) for users' informed choice on both models and hardware. Users who develop new VLMs through training and fine-tuning can also easily use the provided probing framework to diagnose their model capabilities. We anticipate that this toolkit will lower the barrier for evaluations of VLMs in a scientifically rigorous way.

Results from our demonstrative experiment (§ 4) align with the caveat that performance alone may be insufficient for evaluating models (Hu and Levy, 2023; Zhang et al., 2025; Wang and Shi, 2025). Although all evaluated models are considered highly capable on existing benchmarks, many still fail on simple synthetic data. These findings reaffirm concerns about the reliability of VLMs, and VLM-LENS will actively support their analysis and improvement along these lines.

Limitations and planned community support. The current toolkit does not directly support more downstream tasks than probing, such as attention interpretation and neural circuit discovery. Additionally, our current inference and database storage approach prevents the use of gradient-based saliency analyses such as Grad-CAM (Selvaraju et al., 2017). Current users still need to implement their customized functions; however, we anticipate rapid community contributions to the repository for a diverse range of tasks—efforts we are committed to supporting in the long term.

Acknowledgments

We thank Yifan Jiang and Michael Ogezi for their constructive feedback. This work is supported in part by NSERC RGPIN-2024-04395, a Vector Scholarship to HS, and a Canada CIFAR AI Chair award to FS.

References

Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, et al. 2024. Pixtral 12B. *arXiv preprint arXiv:2410.07073*.

Riccardo Ali, Francesco Caso, Christopher Irwin, and Pietro Liò. 2025. Entropy-Lens: The information signature of transformer computations. *arXiv preprint arXiv:2502.16570*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Samyadeep Basu, Martin Grayson, Cecily Morrison, Besmira Nushi, Soheil Feizi, and Daniela Massiceti. 2024. Understanding information storage and transfer in multi-modal large language models. In *NeurIPS*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Gabriela Ben Melech Stan, Estelle Aflalo, Raanan Yehezkel Rohekar, Anahita Bhiwandiwalla, Shao-Yen Tseng, Matthew Lyle Olson, Yaniv Gurwicz, Chenfei Wu, Nan Duan, and Vasudev Lal. 2024. LVLM-Intrepret: An interpretability tool for large vision-language models. In *CVPR*.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. 2024. Paligemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*.

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. In *NeurIPS*.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. 2024. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*.

Jang Hyun Cho, Andrea Madotto, Effrosyni Mavroudi, Triantafyllos Afouras, Tushar Nagarajan, Muhammad Maaz, Yale Song, Tengyu Ma, Shuming Hu, Suyog Jain, et al. 2025. PerceptionLM: Open-access data and models for detailed visual understanding. *arXiv preprint arXiv:2504.13180*.

Bilal Chughtai, Lawrence Chan, and Neel Nanda. 2023. A toy model of universality: Reverse engineering how networks learn group operations. In *ICML*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In *ACL*.

Saurabh Dash, Yiyang Nan, John Dang, Arash Ahmadian, Shivalika Singh, Madeline Smith, Bharat Venkitesh, Vlad Shmyhlo, Viraat Aryabumi, Walter Beller-Morales, et al. 2025. Aya vision: Advancing the frontier of multilingual multimodality. *arXiv preprint arXiv:2505.08751*.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. 2025. Molmo and pixMo: Open weights and open data for state-of-the-art multimodal models. In *CVPR*.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. VSE++: Improving visual-semantic embeddings with hard negatives. In *BMVC*.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. 2024. MME: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*.

Qiyue Gao, Xinyu Pi, Kevin Liu, Junrong Chen, Ruolan Yang, Xinqi Huang, Xinyu Fang, Lu Sun, Gautham Kishore, Bo Ai, et al. 2025. Do vision-language models have internal world models? towards an atomic evaluation. In *Findings of ACL*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *EMNLP*.

Jennifer Hu and Roger Levy. 2023. Prompting is not a substitute for probability measurements in large language models. In *EMNLP*.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.

Sonia Joseph, Praneet Suresh, Lorenz Hufe, Edward Stevinson, Robert Graham, Yash Vadi, Danilo Bzdok, Sebastian Lapuschkin, Lee Sharkey, and Blake Aaron Richards. 2025. Prisma: An open source toolkit for mechanistic interpretability in vision and video. *arXiv preprint arXiv:2504.19475*.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *ICCV*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, et al. 2024. Embodied agent interface: Benchmarking llms for embodied decision making. In *NeurIPS*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *NeurIPS*.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens.

Clement Neo, Luke Ong, Philip Torr, Mor Geva, David Krueger, and Fazl Barez. 2024. Towards interpreting visual information processing in vision-language models. *arXiv preprint arXiv:2410.07149*.

Vedant Palit, Rohan Pandey, Aryaman Arora, and Paul Pu Liang. 2023. Towards vision-language mechanistic interpretability: A causal tracing tool for Blip. In *ICCV*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*.

Hanoona Rasheed, Muhammad Maaz, Sahal Shaji, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao M Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad S Khan. 2024. Glamm: Pixel grounding large multimodal model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13009–13018.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *ICCV*.

Samuel Stevens, Wei-Lun Chao, Tanya Berger-Wolf, and Yu Su. 2025. Sparse autoencoders for scientifically rigorous interpretation of vision models. *arXiv preprint arXiv:2502.06755*.

J Ridley Stroop. 1935. Studies of interference in serial verbal reactions. *Journal of experimental psychology*, 18(6):643.

OpenBMB MiniCPM-o Team. 2025. MiniCPM-o 2.6: A GPT-4o level MLLM for vision, speech, and multimodal live streaming on your phone.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *CVPR*.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024a. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Song XiXuan, et al. 2024b. CogVLM: Visual expert for pretrained language models. In *NeurIPS*.

Yixuan Wang and Freda Shi. 2025. Logical forms complement probability in understanding language model (and human) performance. In *ACL*.

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 *EMNLP*.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. 2025. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12966–12977.

Zhuo Xu, Xiang Xiang, and Yifan Liang. 2025. Overcoming shortcut problem in vlm for robust out-of-distribution detection. In *CVPR*.

Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. 2024. MiniCPM-V: A GPT-4V level MLLM on your phone. *arXiv preprint arXiv:2408.01800*.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*.

Kaichen Zhang, Yifei Shen, Bo Li, and Ziwei Liu. 2024. Large multi-modal models can interpret features in large multi-modal models. *arXiv preprint arXiv:2411.14982*.

Pan Zhang, Xiaoyi Dong, Bin Wang, Yuhang Cao, Chao Xu, Linke Ouyang, Zhiyuan Zhao, Haodong Duan, Songyang Zhang, Shuangrui Ding, et al. 2023. InternLM-XComposer: A vision-language large model for advanced text-image comprehension and composition. *arXiv preprint arXiv:2309.15112*.

Zheyuan Zhang, Fengyuan Hu, Jayjun Lee, Freda Shi, Parisa Kordjamshidi, Joyce Chai, and Ziqiao Ma. 2025. Do vision-language models represent space and how? Evaluating spatial frame of reference under ambiguities. In *ICLR*.

A Contribution Statement

We list the contributions of each author below:

- Project idea: Freda Shi, Ziqiao Ma.
- Substantial prototype design: Eric Huang, Hala Sheta, Freda Shi.
- Prototype implementation: Eric Huang.
- Model implementation: Shuyu Wu, Jiajun Hong, Hala Sheta, Eric Huang, Ilia Alenabi, Ryker Lin, Ruoxi Ning, Daniel Wei, Jialin Yang.
- Documentation, user guide, and tutorials: Ziqiao Ma, Freda Shi, Ilia Alenabi, Ryker Lin, Daniel Wei.
- Substantial code review: Hala Sheta, Shuyu Wu, Eric Huang, Freda Shi.
- Probing experiments: Hala Sheta, Shuyu Wu, Ilia Alenabi, Jiajun Hong, Ryker Lin, Daniel Wei.
- Concept similarity experiments: Ziqiao Ma, Shuyu Wu, Freda Shi.
- Substantial project discussion: Hala Sheta, Eric Huang, Shuyu Wu, Jiajun Hong, Jiawei Zhou, Ziqiao Ma, Freda Shi.
- Paper writing: Hala Sheta, Freda Shi, Ziqiao Ma.
- Project supervision: Freda Shi, Jiawei Zhou, Ziqiao Ma.

B Probing Configuration

For each of our supported models, we extract the hidden representations of the middle (e.g., 16) and last (e.g., 32) of the `post-attention layer norm` module (or equivalent) using our library. This is because we require a 4096-dimensional (or equivalent) representation that can be efficiently stored and contains some relevant information about the input.

Each probe is instantiated with `ReLU` activation and a hidden size of 512, selected empirically to balance performance and efficiency. The input and output sizes are determined dynamically based on the tensors and labels in the input database. We tune the hyperparameters, `learning_rate`, `num_epochs` and `batch_size`, to find the best combinations that incurs the lowest mean validation loss after k-fold cross-validation with $k = 5$. With 3 options for each former parameter, we search through $3^3 = 27$ configurations. Lastly, using the best training configuration, we train two new models from scratch on the original and shuffled datasets, respectively.

ResearStudio: A Human-Intervenable Framework for Building Controllable Deep-Research Agents

Linyi Yang
Southern University of Science and Technology
Resear AI
yanglinyiucd@gmail.com

Yixuan Weng✉
Resear AI
✉ wengsyx@gmail.com

Abstract

Current deep-research agents run in a "fire-and-forget" mode: once started, they give users no way to fix errors or add expert knowledge during execution. We present RESEARSTUDIO, the first open-source framework that places real-time human control at its core. The system follows a *Collaborative Workshop* design. A hierarchical Planner–Executor writes every step to a live "plan-as-document," and a fast communication layer streams each action, file change, and tool call to a web interface. At any moment, the user can pause the run, edit the plan or code, run custom commands, and resume – switching smoothly between *AI-led, human-assisted* and *human-led, AI-assisted* modes. In fully autonomous mode, ResearStudio achieves state-of-the-art results on the GAIA benchmark, surpassing systems like OpenAI's DeepResearch and Manus. These results show that strong automated performance and fine-grained human control can coexist. The full code, protocol, and evaluation scripts are available at https://github.com/ResearAI/ResearStudio. We will continue to update the repository to encourage further work on safe and controllable research agents.[1]

1 Introduction

The advent of Large Language Models (LLMs) (Ouyang et al., 2022; Achiam et al., 2023; Yang et al., 2025) has catalyzed a new era in artificial intelligence, providing powerful engines for reasoning (Weng et al., 2022; Besta et al., 2024), comprehension (Rein et al., 2024; Weng et al., 2024), and generation (Zhu et al., 2024; Li et al., 2024). This has naturally led to the development of LLM-based autonomous agents (Roy et al., 2024; Huang et al., 2025), which leverage these models as a cognitive core to tackle complex (Yao et al., 2023). These long-horizon tasks were previously intractable (Zhang et al., 2025a). Most recently, a new class of advanced autonomous systems, termed Deep Research (DR) agents, have emerged, exemplified by industry-leading solutions such as OpenAI DR (OpenAI, 2025), Gemini DR (Team et al., 2023), and Memento (Zhou et al., 2025).

Yet prevailing agent frameworks (Chen et al., 2024; Zhu et al., 2025; Zheng et al., 2025; OpenAI, 2025) offer only a rigid, one-directional pipeline: once a task is issued, the user is reduced to a passive observer. When the agent misinterprets goals or pursues flawed strategies, there is no channel for timely human intervention, leading to errors, wasted compute, and diminished trust. Current Deep Research agents, therefore, fall short of the collaborative interface envisioned for AI.

We address this gap with the *Collaborative Workshop*, a shared, persistent, and interactive digital interface characterized by three key properties: (1) Transparency – all plans, intermediate artifacts, and actions are visible; (2) Symmetrical Control – humans and AI possess equivalent authority to modify any element; and (3) Dynamic Role Fluidity – control can seamlessly shift between AI-led and human-led workflows.

To this end, we present RESEARSTUDIO, which is the first open-source realisation of this paradigm. Its layered architecture and custom protocol let users pause or resume execution, edit any plan or file, execute their terminal commands, and export the full workspace at any time. Through the human-intervenable interface, users are able to pause and resume the agent's execution, directly edit not only the plan (TODO.md) but also any code or data file, take control of the terminal to run commands, and download the complete state of the workspace. This capability enables two complementary collaboration modes: **AI-led, Human-assisted**: the agent drives the workflow while the user audits, refines, and injects domain knowledge. **Human-led, AI-**

[1]Our live demo is publicly accessible at http://ai-researcher.net:3000/.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 896–905
November 4-9, 2025

Deep Research Agent	Online Search	OpenSource Framework	Pre-research Intervention	Real-time Content Adjustment
OpenAI DeepResearch	✓	×	×	×
OpenAI/Google Canvas	×	×	×	Text editing Only
Google DeepResearch	✓	×	✓	×
Kimi-Researcher	✓	×	×	×
Grok DeepSearch	✓	×	✓	×
Skywork Agent	✓	×	✓	Slide/Doc Only
ResearStudio (Ours)	✓	✓	✓	✓

Table 1: Comparative analysis of Deep Research Agent features. Symbol guide: ✓ indicates explicit support for the feature; × indicates no native support found in the provided materials.

assisted: the user orchestrates high-level strategy and delegates well-defined subtasks to the agent. Our contributions are three-fold:

1. We formalise the *Collaborative Workshop*, unifying transparency, symmetric control, and role fluidity for human-intervenable deep research interface.

2. We release RESEARSTUDIO, a fully open-source deep research agent that enables real-time bidirectional collaboration and live plan editing with the help of a search agent.

3. We empirically show that our design achieves state-of-the-art performance on the GAIA benchmark among existing Deep Research agents from both industry and academia, demonstrating that collaboration enhances, rather than sacrifices, capability.

2 Related Work

The development of autonomous agents has been accelerated by foundational LLM advancements in reasoning, such as Chain-of-Thought (Wei et al., 2022) and self-reflection (Shinn et al., 2023), and in action, through tool-use integration (Schick et al., 2023). Building on this, agent architectures have explored multi-agent coordination, as in AutoGen (Wu et al.), LangGraph and MetaGPT (Hong et al.), or hierarchical task decomposition, such as in AgentOrchestra (Zhang et al., 2025b) and OWL (Hu et al., 2025). While these frameworks significantly advance agent-to-agent and agent-to-environment interactions, they largely overlook the paradigm of direct, real-time human-agent collaboration. ResearStudio, in contrast, applies these foundational reasoning and tool-use capabilities within an architecture designed specifically to place the human user at the center of the workflow.

This collaborative gap is highlighted by two recent, divergent trends in AI systems, as summarized in our comparative analysis in Table 1. On one hand, highly capable autonomous agents like OpenAI's DeepResearch and Grok's DeepSearch demonstrate impressive performance on complex tasks. However, they operate with limited interactivity, offering minimal support for the kind of **Real-time Content Adjustment** necessary for course correction, thus relegating the user to a passive role. On the other hand, collaborative interfaces like OpenAI's and Google's Canvas offer rich editing environments but are typically confined to single-file manipulation ("Text editing Only") and lack the ability to execute complex, multi-tool tasks across an entire project. ResearStudio bridges this divide. It embeds a powerful autonomous agent capable of complex task execution on par with leading systems, but within a fully interactive and multi-file workshop environment. As shown in Table 1, ResearStudio is unique in providing an open-source framework that supports intervention at all stages, uniting state-of-the-art autonomy with genuine human control.

3 The ResearStudio Framework

The "Collaborative Workshop" paradigm is realized through a three-layer architecture, as depicted in Figure 1. Together, these tools form an intervenable substrate: every transformation is transparent, every action reversible, and every intermediate product editable, **turning the agent from an opaque executor into a controllable and reliable research partner.**

3.1 Tool Usage

To make deep-research agents *human-intervenable*, each tool is exposed through the same live interface that streams its inputs and outputs, allowing users to override or refine any step in real time.

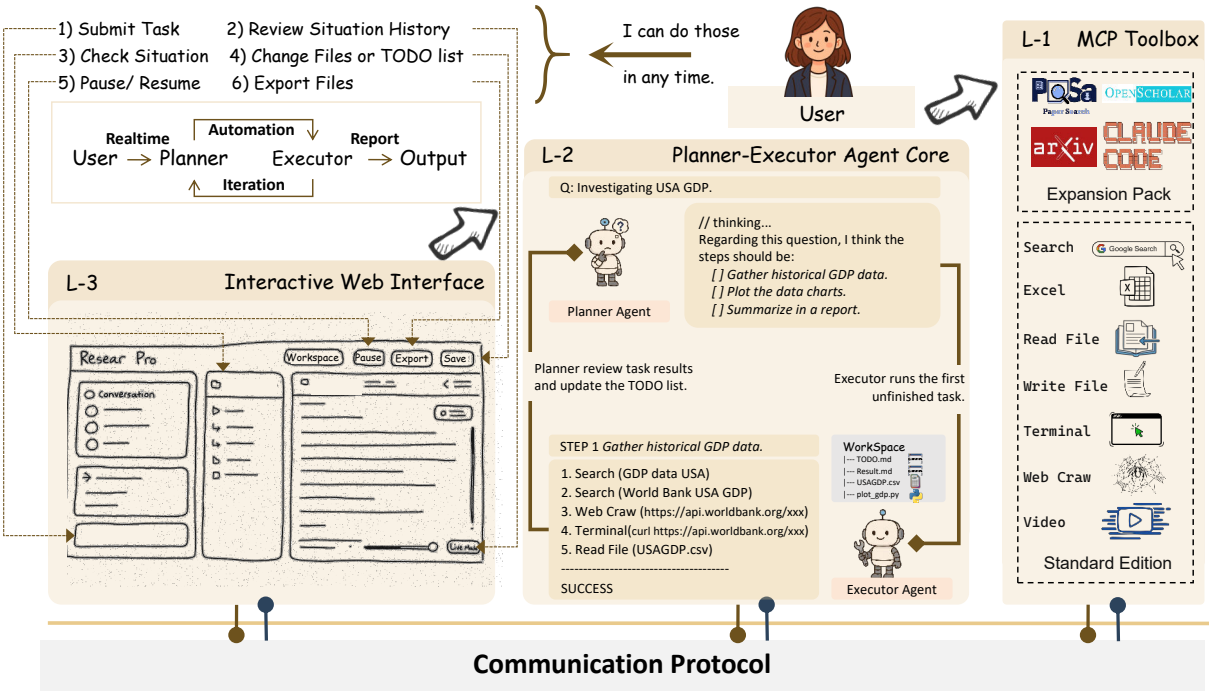


Figure 1: The overall architecture of the ResearStudio framework. This diagram illustrates the three core layers (L-1, L-2, L-3) and the primary workflow. The L-2 Agent Core, composed of a Planner and an Executor, processes a user's request. The Executor carries out steps by using tools from the L-1 MCP Toolbox. The entire process is accessible to the user via the L-3 WebPage, linked by a central Communication Protocol.

Document-Processing Toolkit. Given a file $f \in \mathcal{F}$ with

$$\mathcal{F} = \{\text{jpg}, \text{png}, \text{mp3}, \text{pptx}, \text{xlsx}, \text{csv}, \text{zip}, \text{txt}, \text{json}, \text{xml}, \text{docx}, \text{mov}, \text{pdf}, \ldots\},$$

the system invokes a modality-specific extractor $D(f)$ that yields text, captions, or structured objects (e.g., VLM captions for images, ASR transcripts for audio, slide-wise markdown for `.pptx`, row-wise CSV for spreadsheets). Extracted artefacts immediately appear in the UI, where the user can edit, annotate, or discard them before the agent continues – ensuring that downstream reasoning is always grounded in vetted content.

Search Toolkit. In terms of the search agent, we combine a self-hosted SEARXNG metasearch with CRAWL4AI page fetches. Results are re-ranked by contextual similarity, then streamed to the interface. The researcher can (i) accept, (ii) reject, or (iii) request deeper crawling of any hit. This tight human-in-the-loop filter cuts noise and steers the agent toward authoritative sources.

Deliberate Browser Omission. Although full browser automation offers rich interactions, we observed that LLM planners overuse it, incurring latency and extracting little structured data. By default we exclude browser control; users can re-enable it with a single toggle if a page truly requires dynamic rendering. This design keeps the interface fast and the provenance chain clean.

Code Toolkit. A sandboxed workspace lets the agent (or the user) create files, run shell or Python commands, and inspect outputs with state preserved across iterations. Every script is displayed pre-execution; the researcher can modify code, inject assertions, or roll back to a prior snapshot. Safe-exec guards whitelist common packages (*numpy, pandas, torch, etc.*) to balance flexibility and security. Every script or shell command is first rendered in the UI; the user may edit, comment, or disable the snippet before execution. Standard output, error streams, and rich artefacts (tables, figures) stream back in real time and are logged as immutable cells. A built-in diff viewer records successive file changes, enabling one-click rollback or branch creation for "what-if" explorations.

Interactive Web Interface. We provide a comprehensive view of the agent's conversation, the workspace files, and global controls, enabling the user to monitor progress and intervene at any time.

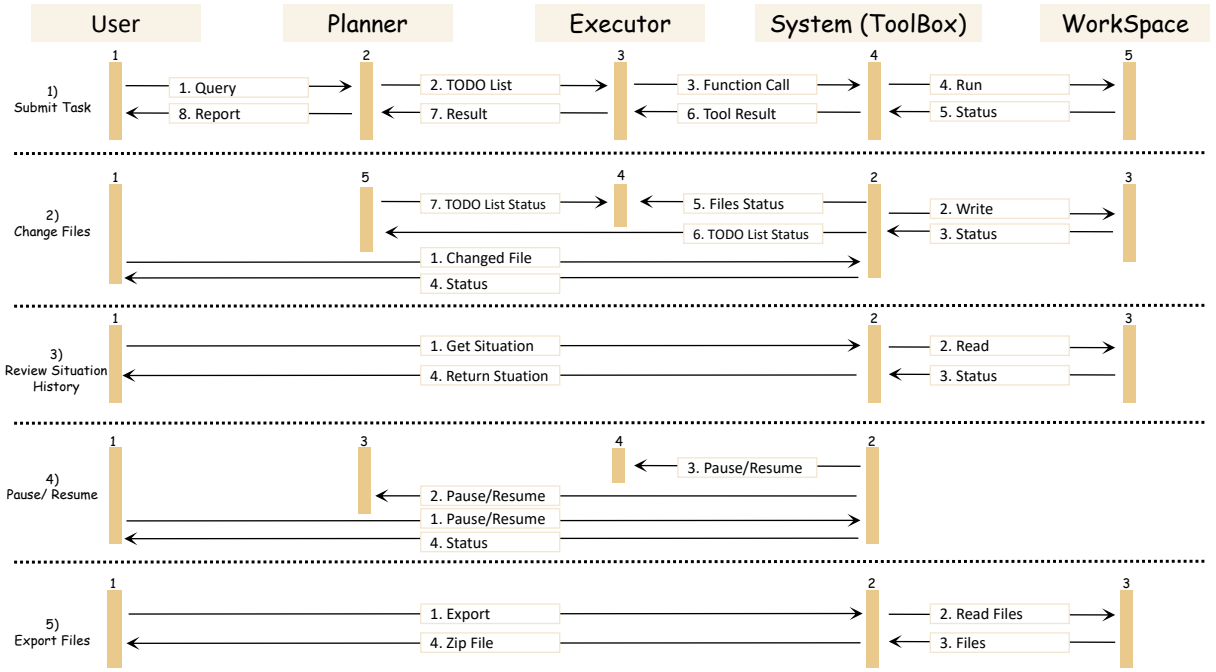


Figure 2: Core interaction workflows of the ResearStudio communication protocol. This diagram details the sequence of messages between the User, Planner, Executor, System (Toolbox), and Workspace for key operations, illustrating the bidirectional flow of information that enables real-time collaboration.

3.2 Bidirectional Protocols

The framework's collaborative capabilities are powered by a dual-layered communication system, whose core workflows are detailed in the sequence diagrams of Figure 2. At the machine level, the Model-Context Protocol (MCP), implemented with 'fastmcp', standardizes the Executor's tool calls into reliable, JSON-based functions. This corresponds to the agent's autonomous execution loop shown in Figure 2 ("Submit Task" workflow). The entire system is orchestrated by a central communication protocol that enables seamless interaction between all components.

More central to our contribution is the event-driven protocol governing the human-agent partnership. This protocol provides the technical foundation for the direct manipulation and control workflows. Upon starting a task, a long-lived connection is established between the frontend and backend. User actions are then translated into specific API calls tagged with a unique task ID. For instance, the "Change Files" workflow is enabled by a 'POST' request containing the new file content, which updates the Workspace and notifies the Executor. Similarly, the "Pause/Resume" workflow is implemented by a request that stalls all backend LLM calls, effectively freezing the agent's cognitive state until resumed. All real-time updates from the agent are streamed back to the user through this persistent connection, with large file contents being lazy-loaded on click to maintain UI responsiveness. This protocol transforms the system into a truly interactive and auditable workshop, where the user is a continuous collaborator rather than a passive observer.

3.3 Backend Models

The Planner is powered by `gpt-4.1`, the Executor by `o4-mini` for datasets excluding GAIA, the image processing by `gpt-4o`, the video agent by `gemini-2.5-pro` and the audio agent by `Assembly AI`. We select the `o3` as the executor in the GAIA benchmark. Each module is chosen to balance efficiency and task-specific capability, ensuring reliable multimodal perception, robust planning, and precise execution.

4 Experiments

To validate the effectiveness and capabilities of the ResearStudio framework, we conducted a rigorous evaluation on the GAIA benchmark (Mialon et al., 2023), a standard for testing general-purpose agentic systems on complex reasoning and multi-step tool use. **All experiments were performed in a fully autonomous mode, with no human intervention from task initiation to completion**,

Agent	Level-1	Level-2	Level-3	Average
ODR-smolagents	67.92	53.49	34.62	55.15
AutoAgent	71.70	53.49	26.92	55.15
OWL	84.91	67.44	42.31	69.09
A-World	**86.79**	69.77	34.62	69.70
OpenAI-DeepResearch	74.29	69.06	47.60	67.36
ResearStudio (Pass@1)	**77.36**	**69.77**	**61.54**	**70.91**

Table 2: Performance comparison of our agent against baseline methods ODR-smolagents (Roucher et al., 2025), AutoAgent (Chen et al., 2024), OWL (Hu et al., 2025), A-World (at Ant Group, 2025), and OpenAI-DeepResearch (OpenAI, 2025) on GAIA benchmark (Mialon et al., 2023). Average Task Runtime across all GAIA levels is approximately 20 minutes, while simpler tasks or specific successful cases (as discussed in Section 5) can be completed in under 10 minutes.

Agent	Level-1	Level-2	Level-3	Average
OWL (Hu et al., 2025)	75.27	61.01	32.65	60.80
A-World (at Ant Group, 2025)	80.65	64.78	24.49	63.12
ResearStudio (Ours)	**84.95**	**72.33**	**59.18**	**74.09**

Table 3: Performance comparison of our agent against open-source frameworks on the test set of the GAIA benchmark.

thereby assessing the core performance of our architecture. For our agent's configuration, the **Planner** is powered by `gpt-4.1`, while the **Executor** utilizes `o4-mini` for its tactical operations. To handle multimodal tasks, the framework is equipped with specialized models: `gpt-4o` for image-related tasks, `gemini-2.5-pro` for video processing, and Assembly AI for audio tasks. For evaluation, we report the **Exact Match (EM)** metric, where a prediction is considered correct only if it exactly matches the reference answer after normalizing for case, punctuation, and articles. The EM score is defined as the percentage of answers that achieve a perfect match.

Overall Results. Our experimental results demonstrate that ResearStudio achieves state-of-the-art performance on the GAIA benchmark across both validation and test sets. As shown in Table 2, on the GAIA validation set, our framework achieves a leading average score of **70.91%**, outperforming other established agents such as A-World (69.70%) and OpenAI-DeepResearch (67.36%). Notably, ResearStudio shows exceptional capability on the more complex tasks, achieving the highest scores on both Level-2 (**69.77%**) and the highly challenging Level-3 (**61.54%**) tasks. To further validate these findings on unseen data, we evaluated our agent on the GAIA test set. The results, presented in Table 3, solidify ResearStudio's superior performance. Our agent achieves an overall average score of **74.09%**, surpassing all listed baselines across every difficulty level, with scores of **84.95%** on Level-1, **72.33%** on Level-2, and **59.18%** on Level-3. These robust results, obtained in a fully autonomous setting, validate the efficacy of our Planner-Executor architecture and modular tool integration, proving that a framework designed for human-collaboration can also deliver superior performance on its own.

5 Discussion

The architecture of ResearStudio materializes into a practical and effective collaborative workflow, as illustrated by the user interface in Figure 3 and Table 4. The multi-panel design provides a user with complete situational awareness. For instance, a user can monitor the agent's detailed execution steps in the "Conversation & Activities" log while simultaneously inspecting the files it generates, such as "analysis_penguin.py" and "todo.md", in the "File Explorer". This transparent process allows for timely and precise intervention. If the user observes the agent writing flawed code into the "File Editor", they are not forced to wait for an error. Instead, using the "Global Controls", they can pause the agent, directly correct the script, and then resume the task. This action of switching from passive observation to active contribution demonstrates the fluid transition between an AI-led and a human-assisted workflow.

Operational Parameter / Metric	Typical Value
Average Task Runtime (GAIA)	~20 minutes
Max Concurrent Workers	50
Max Interaction Rounds	30
Average Steps per Task	~25
Typical Final Workspace Size	~100 MB

Table 4: Key operational parameters and metrics for a typical ResearStudio run on a complex task. The system is designed for both efficiency and the capacity to handle long-horizon problems.

Beyond enabling a more collaborative workflow, the ResearStudio architecture is also engineered for practical efficiency and scalability, with typical operational parameters summarized in Table 4. The

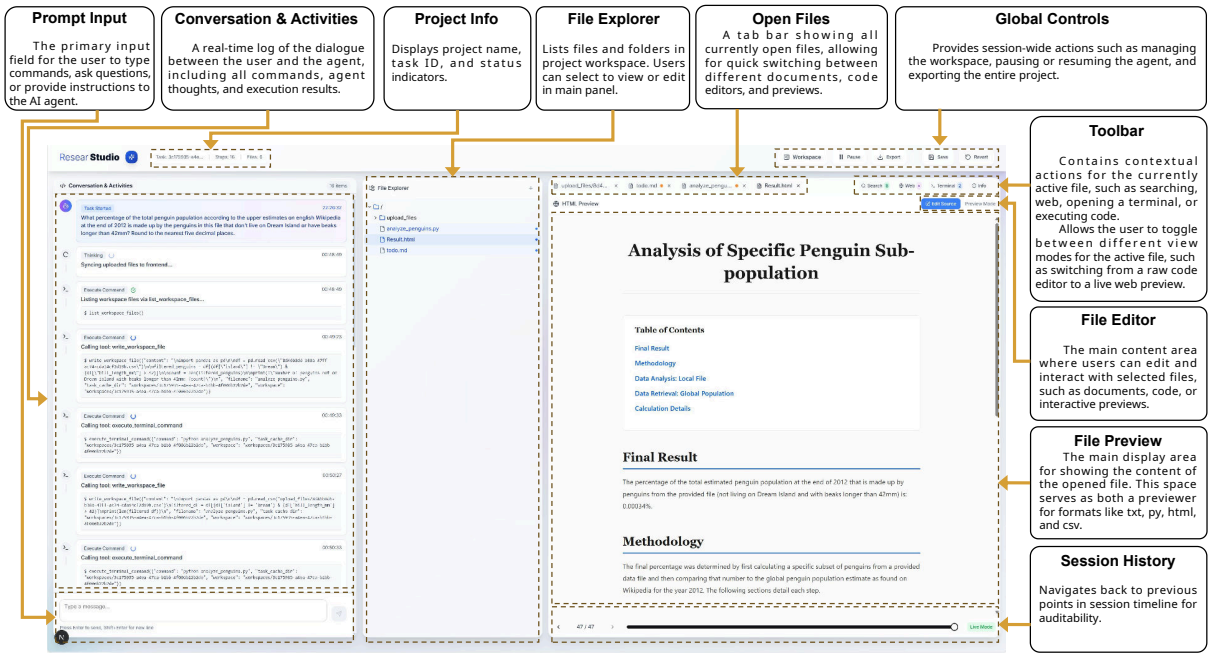


Figure 3: An overview of the ResearStudio user interface, designed as an integrated workspace for human-agent partnership. The layout provides a comprehensive view by juxtaposing the agent's real-time execution trace in the activity log with the tangible project artifacts in the file explorer and editor. This design allows a user to seamlessly monitor autonomous operations while retaining the ability to directly interact with and manage all project files and system states through dedicated controls.

framework is capable of managing long-horizon tasks, supporting up to 50 interaction rounds between the Planner and Executor, which allows for deep, iterative problem-solving. This process is highly efficient; a complex GAIA Level-3 task, for example, completes in approximately 10 minutes, a result of offloading heavy computation to dedicated tools rather than relying on pure model reasoning. Our system architecture supports a high degree of concurrency, capable of handling up to 50 concurrent workers, which facilitates scalable deployment for complex, multi-faceted projects. This efficiency makes the human-in-the-loop paradigm practical and appealing. A user is far more likely to engage and collaborate with a system that produces tangible results in minutes, not hours. The fully auditable record of all actions in the "Conversation & Activities" panel, combined with the "Session History" feature, not only enhances trustworthiness but also allows the completed workspace to serve as a detailed, reusable template for future tasks, further amplifying the long-term value of each collaborative session.

6 Conclusion

This work presented RESEARSTUDIO, an open-source framework that makes human–AI collaboration a foundational element rather than an afterthought. We contribute (i) a practical architecture that combines a hierarchical Planner–Executor with a live, bidirectional protocol to expose the agent's reasoning as an editable document, and (ii) an extensive empirical study showing that this transparency and control do not diminish autonomous performance. On the GAIA benchmark, RESEARSTUDIO attains state-of-the-art results while allowing users to intervene, correct, and guide the system at any stage. These findings demonstrate that accountability and efficiency can coexist in deep research agents. Ultimately, ResearStudio provides a concrete path forward for building the next generation of human-intervenable deep research agent. By releasing the full codebase, we aim to spur further work on AI systems that remain powerful yet verifiable, fostering safer and more trustworthy deployment in high-stakes domains.

Limitations

We acknowledge several limitations in the current version of ResearStudio. First, the framework's effectiveness in its collaborative mode is highly dependent on the user's domain expertise to identify subtle errors, and the requisite continuous monitoring can be cognitively demanding. This positions

the system more as a powerful tool for experts than a universally accessible partner for novices. Furthermore, while our experiments validate the architecture's strong autonomous performance, they do not yet formally quantify the practical benefits of the human-in-the-loop features central to our design. Finally, our current safety measures are primarily architectural, focusing on operational containment through sandboxing, but they have not yet been rigorously stress-tested against active adversarial attacks.

These limitations define clear avenues for our future work. To mitigate cognitive load and broaden the framework's accessibility, we plan to develop semi-autonomous intervention mechanisms, such as AI-powered alerts that flag potential errors or logical inconsistencies for human review. To empirically validate the "Collaborative Workshop" paradigm, a critical next step is to conduct formal Human-Computer Interaction (HCI) studies. These studies will measure key metrics—such as task completion times with and without intervention, error correction rates, and user satisfaction scores—to provide direct evidence of collaborative utility. Lastly, to bolster the system's robustness, we will undertake dedicated red-teaming efforts to assess its resilience against threats like prompt injection, data exfiltration, and the generation of harmful content, using the findings to engineer more sophisticated, dynamic safety protocols.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Agent Team at Ant Group. 2025. Aworld: A unified agent playground for computer and phone use tasks.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje Karlsson, Jie Fu, and Yemin Shi. 2024. Autoagents: a framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 22–30.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, and 1 others. 2025. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, Jianye Hao, and 1 others. 2025. Deep research agents: A systematic examination and roadmap. *arXiv preprint arXiv:2506.18096*.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, and 1 others. 2024. From generation to judgment: Opportunities and challenges of llm-as-a-judge. *arXiv preprint arXiv:2411.16594*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

OpenAI. 2025. Deep research system card.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Aymeric Roucher, A Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. smolagents: A smol library to build great agentic systems.

Devjeet Roy, Xuchao Zhang, Rashi Bhave, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. 2024. Exploring llm-based agents for root cause analysis. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, pages 208–219.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Kang Liu, and Jun Zhao. 2024. Mastering symbolic operations: Augmenting language models with compiled neural networks. In *The Twelfth International Conference on Learning Representations*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2022. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. 2025a. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*.

Wentao Zhang, Ce Cui, Yilei Zhao, Yang Liu, and Bo An. 2025b. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *arXiv preprint arXiv:2506.12508*.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. 2025. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv: 2508.16153*.

Minjun Zhu, Yixuan Weng, Linyi Yang, and Yue Zhang. 2025. DeepReview: Improving LLM-based paper review with human-like deep thinking process. pages 29330–29355, Vienna, Austria. Association for Computational Linguistics.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2024. Knowagent: Knowledge-augmented planning for llm-based agents. *arXiv preprint arXiv:2403.03101*.

A Discussion and Case Studies

While quantitative benchmarks validate the high autonomous performance of ResearStudio, a deeper understanding of its practical strengths and limitations can be gained through qualitative analysis of its operational workflow. This section presents two contrasting case studies from the challenging GAIA Level-3 benchmark. The first case illustrates the framework's proficiency in autonomously solving complex, engineering-style computational tasks. The second, a failure case, reveals a common vulnerability in autonomous agents and, in doing so, highlights the profound utility and core design philosophy of our Collaborative Workshop paradigm.

Our framework demonstrates remarkable efficiency on tasks that require algorithmic thinking and precise calculation. In one such GAIA Level-3 task, the agent was presented with a complex computational puzzle involving two 12-digit numbers, column transpositions, and a weighted-sum checksum validation. ResearStudio solved this intricate problem in approximately four minutes over 16 discrete steps. An analysis of its execution trace, presented in Figure 4, reveals a highly effective strategy. The **Planner** correctly identified that a brute-force search was the optimal approach and created a plan to first write a Python script to codify the logic. The **Executor** then successfully implemented this plan, generating and running a 'solve_puzzle.py' script. By intelligently offloading the complex computation to code rather than attempting to reason through it in-prompt, the agent demonstrated efficient and robust problem-solving. This case underscores ResearStudio's strength in

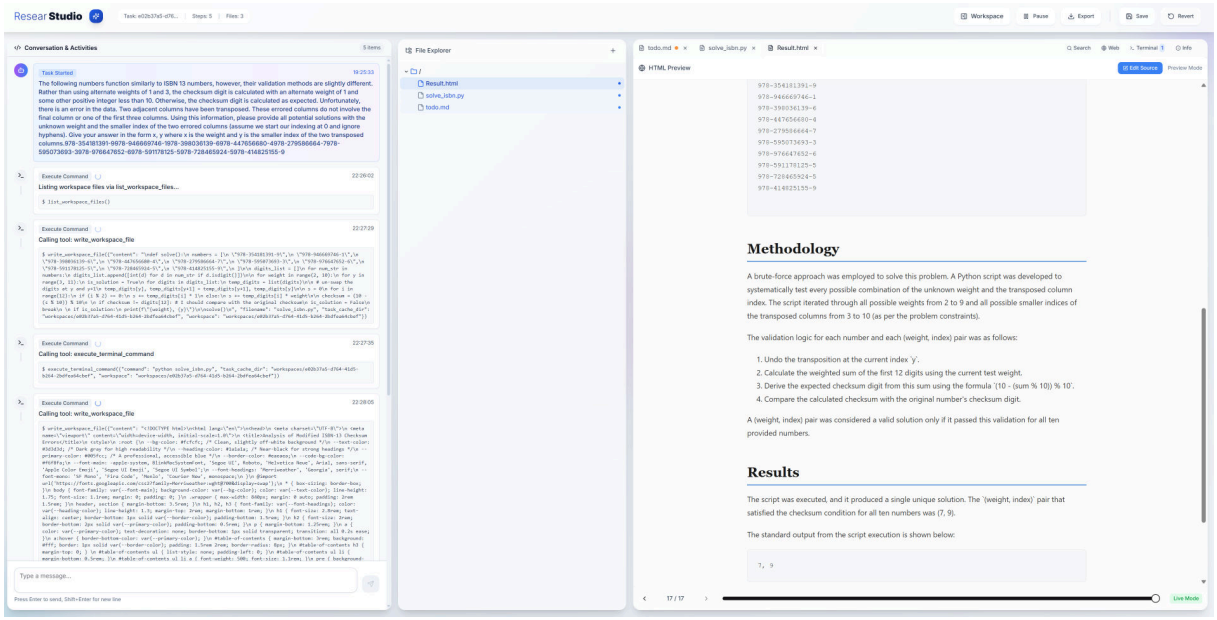


Figure 4: A successful execution trace of a GAIA Level-3 computational puzzle within the ResearStudio UI, showcasing the agent's ability to write and execute code to find a correct solution.

handling well-defined engineering and computational challenges where logic can be explicitly codified and executed.

However, when faced with tasks requiring a nuanced interpretation of ambiguous real-world data, the limitations of pure autonomy become apparent. For GAIA Level-3, the agent was asked to calculate the volume of a Freon-12 container at the bottom of the Mariana Trench. As shown in Figure 5, the agent's workflow was initially logical, correctly identifying the need to find the temperature and pressure to determine the substance's density. The failure occurred when its web search returned conflicting information: the near-freezing ambient temperature of the trench (1-4°C) and the extreme temperature of hydrothermal vents located within it (400°C). Lacking the human-like common sense to disambiguate this context, the agent incorrectly selected the extreme 400°C value. This single error created a cascading failure, leading it to select a completely wrong density value and produce a final answer that was orders of magnitude incorrect. This case highlights a critical vulnerability in many autonomous systems: a conflict between the model's internal knowledge and conflicting information retrieved from the open web can lead to catastrophic, yet logically consistent, failures.

While this failure reveals a limitation, it simultaneously provides the most potent demonstration of ResearStudio's core value. In a traditional "black-box" system, this task would have simply failed, leaving the user with a useless result and no recourse but to restart. Within the **Collaborative Workshop**, however, this catastrophic failure becomes a manageable, correctable mistake. A human collaborator, observing the agent's real-time activity log, would immediately recognize the 400°C temperature as nonsensical. At that precise moment, they could use the interface to **'Pause'** the agent, directly edit the agent's 'TODO.md' plan or a note file in the shared workspace to specify "use ambient temperature of 4°C," and then **'Resume'** the task. This simple, intuitive intervention would have guided the agent back to the correct path, leveraging human expertise to resolve the exact kind of contextual ambiguity that AI struggles with. This ability to seamlessly transfer control and inject human judgment into the loop is the ultimate utility of ResearStudio. It transforms the agent from an opaque, brittle tool into a resilient, trustworthy partner, providing the essential safeguard needed to deploy autonomous systems on complex, real-world problems.

B Safety Considerations

The introduction of autonomous agents that can interact with file systems and external web content necessitates a robust safety framework. In ResearStudio, safety is addressed through a combination of architectural design choices and interactive oversight capabilities. This section discusses the key

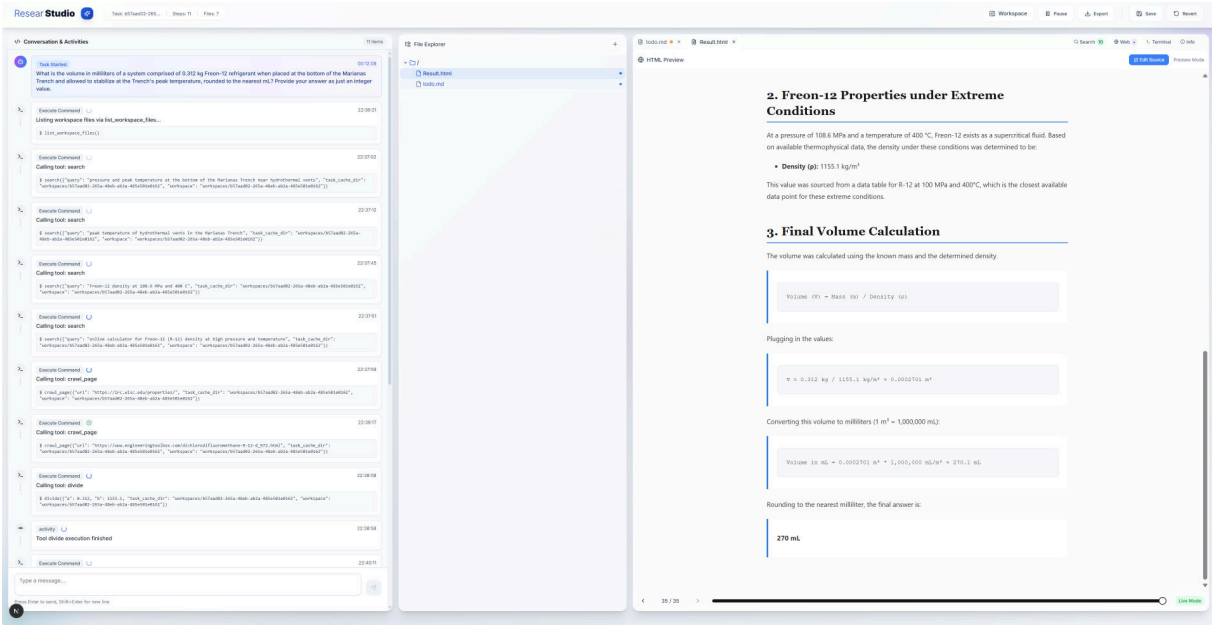


Figure 5: An execution trace of a failed GAIA Level-3 physics problem. The agent correctly structures the problem but fails by selecting the wrong temperature (400°C), leading to a cascade of incorrect calculations.

safety mechanisms, focusing on operational containment and the safeguards against both external and user-initiated threats.

B.1 Operational Safety and Sandboxing

A foundational element of ResearStudio's safety posture is the strict isolation of each task. Upon initiation, every task is assigned a unique workspace that is fully sandboxed from the host system and from other, concurrent tasks. This architectural choice is critical for preventing risks such as data exfiltration or unauthorized access to external resources. All tool operations, whether writing a file or executing a command, are confined within the boundaries of this isolated directory. This containment ensures that even if an agent were to exhibit unintended behavior, the potential impact would be restricted to the immediate task environment.

Furthermore, the execution of tools is managed through a layer of abstraction provided by the MCP. The agent does not directly execute system calls; instead, it formulates structured requests to independent, sandboxed tool services. For instance, the Python tool, as used in systems like Deep Research, runs within its own constrained environment without direct internet access, mitigating cybersecurity risks associated with arbitrary code execution. Similarly, while the agent can request actions via the *Terminal* tool, it does so through this mediated protocol. This prevents the agent from gaining unrestricted shell access and limits the scope of its command-line capabilities to operations relevant to the sandboxed workspace.

B.2 Interactive Oversight and Safeguards

While sandboxing provides technical containment, an additional layer of safety is established through interactive oversight and system-level safeguards. The "Plan-as-Document" principle is a key component, as it externalizes the agent's intentions into a human-readable "TODO.md" file before execution. This provides a critical checkpoint for a user to review the agent's proposed actions and intervene to prevent the pursuit of flawed or unsafe strategies. This is particularly relevant for mitigating risks from external content, such as prompt injection, where a user can spot anomalous changes in the agent's plan or activity log and use the "Pause" control to halt execution.

Crucially, these safeguards are not limited to protecting against external threats; they also address potential misuse by the user. Even within this collaborative framework, the system is designed to prevent the execution of malicious instructions. All user prompts are evaluated by input classifiers against safety policies to filter requests for disallowed content. This layered approach ensures that while the user is an empowered collaborator, the system maintains its own independent capacity to enforce safety protocols and prevent misuse.

OpenS2S: Advancing Fully Open-Source End-to-End Empathetic Large Speech Language Model

Chen Wang[1,2], Tianyu Peng[1,2,3], Wen Yang[1,2], Yinan Bai[1,2], Guangfu Wang[4], Jun Lin[4], Lanpeng Jia[4], Lingxiang Wu[1,3], Jinqiao Wang[1,2,3], Chengqing Zong[1,2], Jiajun Zhang[1,2,3] *

[1] Institute of Automation, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] Wuhan AI Research [4] GWM AI Lab
wangchen2020@ia.ac.cn jjzhang@nlpr.ia.ac.cn

Abstract

Empathetic interaction is a cornerstone of human-machine communication, due to the need for understanding speech enriched with paralinguistic cues and generating emotional and expressive responses. However, the most powerful empathetic LSLMs are increasingly closed off, leaving the crucial details about the architecture, data and development opaque to researchers. Given the critical need for transparent research into the LSLMs and empathetic behavior, we present OpenS2S, a fully open-source, transparent and end-to-end LSLM designed to enable empathetic speech interactions. Based on our empathetic speech-to-text model BLSP-Emo [Wang et al., 2024a], OpenS2S further employs a streaming interleaved decoding architecture to achieve low-latency speech generation. To facilitate end-to-end training, OpenS2S incorporates an automated data construction pipeline that synthesizes diverse, high-quality empathetic speech dialogues at low cost. By leveraging large language models to generate empathetic content and controllable text-to-speech systems to introduce speaker and emotional variation, we construct a scalable training corpus with rich paralinguistic diversity and minimal human supervision. We release the fully open-source OpenS2S model, including the dataset, model weights, pre-training and fine-tuning codes, to empower the broader research community and accelerate innovation in empathetic speech systems.

	Demo	OpenS2SDemo
	Code	CASIA-LM/OpenS2S
	Model	CASIA-LM/OpenS2S
	Data	CASIA-LM/OpenS2S_Datasets

1 Introduction

Empathy is a fundamental pillar of human interactions, fostering everything from prosocial behavior to deeper connections [Morelli et al., 2015]. Modeling and understanding empathy is a complex task for artificial intelligence, yet its integration is crucial for fostering more natural and effective human-machine communication [Rashkin et al., 2018]. In the realm of Large Speech Language Models (LSLMs), this challenge is particularly pronounced. Speech inherently conveys a wealth of rich paralinguistic information, including intonation, rhythm, volume variations, and cues related to speaker attributes like age and gender. This intricate paralinguistic content makes speech communication highly sensitive, rendering flat or unnuanced responses from automated systems unacceptable. Consequently, developing empathetic speech systems is vital for creating more natural and human-centered artificial intelligence.

While recent advancements in LSLMs have significantly enhanced audio processing and enabled robust semantic-based instruction following in conversations [Kannan et al., 2019, Radford et al., 2023, Kheddar et al., 2024, Hao et al., 2023, Li et al., 2023, Barrault et al., 2023, Huang et al., 2023], most existing models tend to overlook critical paralinguistic information in speech, thereby fundamentally limiting their native empathetic interaction capabilities. Although some LSLMs [Défossez et al., 2024, Zeng et al., 2024, Ding et al., 2025] demonstrate strong empathetic performance, they typically necessitate extensive pre-training on millions of hours of high-quality speech data. This reliance on vast datasets incurs substantial annotation, computation, and training costs, setting up a significant barrier to their broader adoption and development. Furthermore, many of the most advanced models, particularly commercial models like GPT-4o [Hurst et al., 2024] and Gemini are fully proprietary and closed-source. This lack of transparency makes it challenging to analyze their internal mechanisms, reproduce their empathetic behaviors, or build upon their architectures for fur-

* Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 906–917
November 4-9, 2025

Table 1: The degree of openness of Open LSLMs.

Name	LLaMA-Omni2	Qwen2-Audio	GLM-4-Voice	Kimi-Audio	OpenS2S
Training Data	✘	✘	✘	✘	✔
Pre-training Code	✘	✘	✘	✘	✔
Fine-tuning Code	✘	✘	✘	✔	✔
Model	✔	✔	✔	✔	✔
Empathetic	✘	✘	✔	✔	✔

ther scientific research and development. To scientifically study the empathetic behaviors in LSLMs, including potential biases, cultural variations, and their ethical implications, we believe that access to powerful, fully open empathetic LSLMs is critical to the advancement of this field.

To address the aforementioned limitations, we propose OpenS2S, a fully open-source, end-to-end LSLM. OpenS2S not only exhibits competitive foundational speech capabilities but also features an efficient streaming architecture based on interleaved decoding. Crucially, in contrast to existing models that achieve empathetic capabilities through resource-intensive pre-training, OpenS2S attains comparable empathetic interaction performance with significantly lower training data and computational costs. Moreover, the empathetic support provided by OpenS2S transcends mere paralinguistic cues, extending deeply into the semantic content of the dialogue.

Overall, our main contributions are as follows:

1. **Model Construction and Training:** We build an efficient speech-to-speech empathetic model based on an advanced framework and conduct extensive training using high-quality data. This model can provide a more convenient and natural way for humans to interact with artificial intelligence.

2. **Automatic Empathetic Speech Instruction Dataset Construction:** We propose a data augmentation method for empathetic speech dialogue by combining the strengths of large language models (LLMs) and text-to-speech (TTS) models. LLMs are used to generate diverse user queries and empathetic responses, while voice cloning ensures input speaker diversity. InstructTTS further enables controllable emotional expression in speech responses, facilitating the construction of rich, high-quality training data with minimal human annotation.

3. **Fully Open-Source Release:** To foster collaborative research and accelerate innovation in empathetic LSLMs, we release all the resources, including the model weights, all codes for constructing datasets, pre-training, fine-tuning and evaluation, and the synthetic datasets, providing fully transparency and reproducibility for the community.

2 Method

2.1 Architecture

The OpenS2S model architecture is shown in Figure 1, comprising four components: an audio encoder, an instruction-following LLM, a streaming speech decoder, and a token2wav decoder. Next, we will describe how to understand continuous speech input and ultimately generate an empathetic speech response.

Audio Encoder The Audio Encoder is responsible for transforming this raw audio signal into a more manageable and meaningful representation. To achieve this we use the encoder of Qwen2-Audio [Chu et al., 2024] to extract features from the audio waveform due to its powerful ability to encode semantic content and paralinguistic information. These features generated by the audio encoder are encoded at a frequency of 25Hz. To further reduce the sequence length, the encoded representations are fed into a speech adapter, which comprises a downsampling module and a feed-forward network. The downsampling module, consisting of two CNN layers, is designed to compress the sequence length by a factor of 4. Finally, the features output by the speech adapter yield continuous encoded representations at 6.25Hz.

Instruction-Following LLM The audio embeddings and text embeddings are concatenated to form interleaved input sequences for the large language model. We select Qwen3-8B-Instruct [Yang et al., 2025] as the LLM, leveraging its robust text processing capabilities.

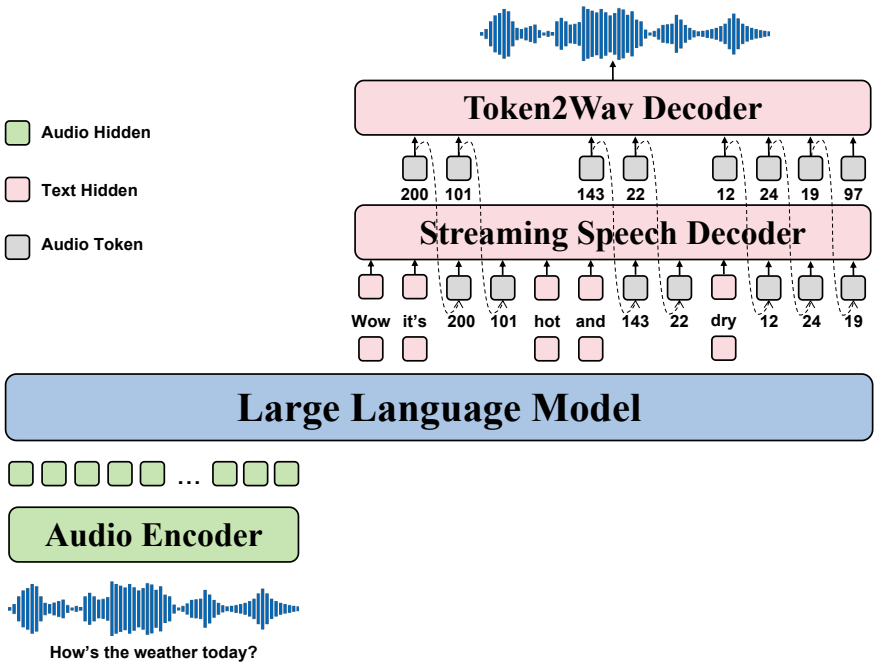


Figure 1: Architecture of our proposed OpenS2S.

Streaming Speech Decoder To enable streaming speech generation, we adopt a framework inspired by Minmo [Chen et al., 2025] and LLaMA-Omni2 [Fang et al., 2025]. The speech response is first converted into discrete tokens using a supervised semantic speech tokenizer. Then, an autoregressive text-to-speech language model is used to generate speech tokens conditioned on the hidden states of the LLM, enabling real-time generation.

The speech tokenizer is implemented by inserting a quantization module into the encoder of Whisper-large-v3 [Radford et al., 2023], ultimately producing a token sequence at a resolution of 12.5 tokens per second with a vocabulary size of 16,384. We leverage the pretrained speech tokenizer from GLM-4-Voice [Zeng et al., 2024].

Once the speech response is tokenized, a decoder-only Transformer models the conditional generation from LLM hidden states to speech tokens. This decoder is initialized from Qwen3-1.8B, with its vocabulary extended to include the 16,384 speech tokens. The input to the streaming speech decoder consists of the final hidden states from the LLM, which are first projected via a linear layer to match the embedding dimension of the speech decoder.

To achieve streaming generation, we interleave the LLM hidden states and generated speech tokens in a predefined ratio: for every M hidden states consumed, N speech tokens are generated (in our implementation, $M = 4$ and $N = 8$). After all hidden states are consumed, the model continues to autoregressively generate the remaining speech tokens until the response is complete. During training, the cross-entropy loss is computed only on the generated speech tokens.

Token2Wav Decoder The speech tokens generated by the streaming speech decoder are subsequently converted into the final speech waveform by the token2wav decoder. This module comprises two key components: a chunk-aware causal flow matching model, which incrementally synthesizes every M speech tokens into mel-spectrograms in a streaming fashion, and a HiFi-GAN vocoder, which converts the mel-spectrograms into the final waveform. Both the flow matching model and the vocoder are adopted from the pretrained components in GLM-4-Voice [Zeng et al., 2024].

2.2 Training Strategy

OpenS2S employs a multi-stage training approach to ensure its capability in speech understanding and generation. The specific training strategy is described in detail in Appendix A.

3 Data Collection

3.1 Pre-training

Speech Understanding For the semantic alignment stage, we utilize publicly available ASR

datasets, and for the emotion alignment stage, we employ standard SER datasets. The ASR datasets include LibriSpeech [Panayotov et al., 2015], CommonVoice 13.0 [Ardila et al., 2019], and the GigaSpeech [Chen et al., 2021] M subset, comprising approximately 1.9 million English (speech, text) pairs. A comparable number of Chinese ASR samples are randomly drawn from WeNetSpeech [Zhang et al., 2022]. The SER datasets consist of IEMOCAP [Busso et al., 2008], MELD [Poria et al., 2018], CMU-MOSEI [Zadeh et al., 2018], MEAD [Wang et al., 2020], and ESD [Zhou et al., 2022], collectively covering around 70k utterances in both English and Chinese.

Speech Generation In the first stage of pretraining, aimed at expanding the vocabulary and enhancing the speech decoder's capacity to generate speech tokens, we use 5k hours of English and 5k hours of Chinese speech randomly sampled from the Emilia [He et al., 2024] dataset. For connecting the large language model with the speech decoder and adapting it into an interleaved streaming generation decoder, we further sample 1k hours each of English and Chinese data from the first-stage dataset for training.

3.2 Supervised Fine-tuning

Existing open-source speech instruction datasets commonly face three key challenges:

- Limited speaker diversity, which undermines model robustness to varied speech inputs. For example, datasets such as InstructS2S-200K [Fang et al., 2024] and E-Chat [Xue et al., 2024] rely on TTS systems to synthesize speech from text, resulting in limited variability in speaker characteristics.

- Neglect of paralinguistic information, with an exclusive focus on semantic content. Although VoiceAssistant-400K [Xie and Wu, 2024] introduces speaker diversity via voice cloning, it fails to capture critical paralinguistic cues such as emotion and speaking style.

- Insufficient label granularity, as exemplified by SD-Eval [Ao et al., 2024], which is divided into subsets that annotate different paralinguistic attributesfeatures in isolation—such as emotion or gender—in isolation. No single subset offers joint annotations across multiple paralinguistic dimensions.

To address these limitations, we propose a fully automated framework for constructing an empathetic speech instruction dataset. This framework systematically enhances speech diversity and representativeness across dimensions such as emotion, age, and gender through the integration of heterogeneous data sources. The full pipeline is illustrated in Figure 2 and comprises the following three stages:

Collection and Manual Annotation of Seed Audio We begin by selecting seed audio samples from several publicly available speech emotion recognition datasets. These datasets cover a wide range of emotions and speaker demographics, including children, adults, and the elderly, ensuring strong representativeness and diversity. Each selected sample is manually annotated with its transcribed text, speaker gender, age, and emotional label. This results in 1,000 English and 1,000 Chinese seed audio samples with rich multi-dimensional annotations.

Generation of Speech Instructions A straightforward solution to generating speech instructions is to convert existing text instruction datasets into speech. However, such datasets present two challenges: (1) tasks involving math or programming are often unsuitable for conversational speech scenarios; and (2) most instruction data neglect paralinguistic factors. Inspired by the Self-Instruct paradigm, we leverage Qwen3-32B-Instruct to automatically generate task instructions that are sensitive to paralinguistic cues. For example, the model might generate the instruction "Do you think I can run a marathon?" and tag it as age-sensitive, suggesting it be read in an elderly voice. We then randomly select a seed audio whose emotion label is "elderly" as an audio prompt and use CosyVoice2 [Du et al., 2024] for voice cloning, preserving the emotion and gender attributes of the selected seed. This process yields 50k English and 50k Chinese speech instructions with diverse paralinguistic characteristics.

Generation of Speech Response For each speech instruction, we annotate its transcribed text, emotion, age, and gender based on the matched seed audio. Both the semantic content and paralinguistic labels are input into Qwen3-32B-Instruct with "thinking mode" enabled. Inspired by LLaMA-Omni, we prompt the model to generate concise, dialogue-appropriate, and empathetic text

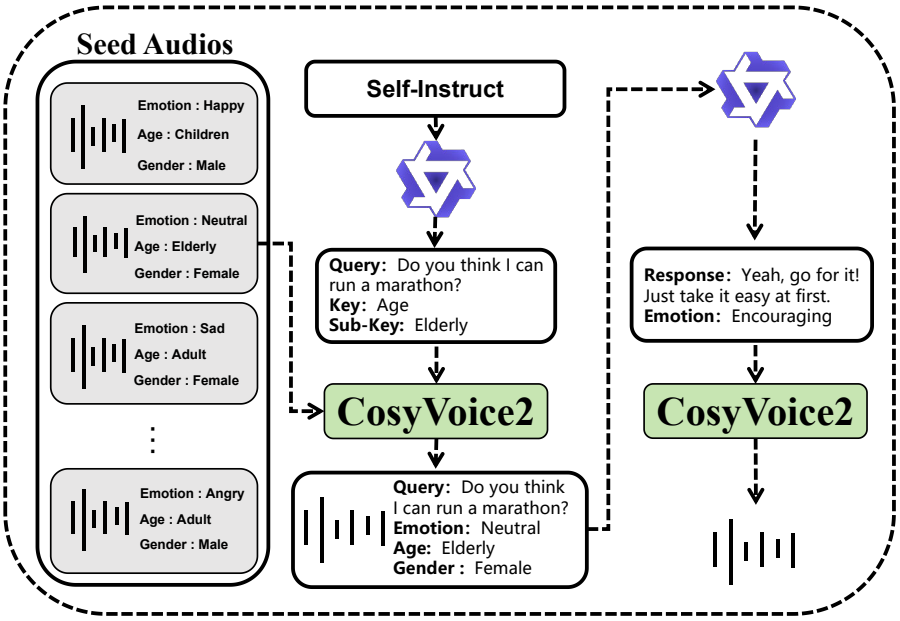


Figure 2: Automatic workflow for constructing empathetic speech instruction dataset.

responses. We then prompt the same model to infer the appropriate emotional tone for delivering the response, conditioned on the original instruction, the response content, and the paralinguistic features of the input speech. Finally, we use a consistent reference voice as a prompt and control CosyVoice2 via instruction prompts to synthesize emotionally expressive speech responses.

Through this systematic construction process, we obtain an empathetic speech instruction dataset characterized by multi-dimensional emotion annotation, expressive emotional delivery, and diverse speaker profiles. The statistics of the constructed data are presented in Figure 3. In total, we construct 50k English and 50k Chinese speech-to-speech empathetic samples. The input speech includes three types of paralinguistic information tags (i.e. emotion, gender, age)), and the output speech is fixed as a young female voice responding with different emotions.

To retain the model's general instruction-following capabilities beyond empathetic dialogue, we further incorporate general-purpose data. Specifically, we extract 50k English instructions from Instruct200K [Fang et al., 2024], translate them into Chinese using Qwen3-32B-Instruct, and select seed audio with neutral emotional labels to convert these instructions into speech. Applying the same speech response generation process as above, we obtain an additional 100k bilingual speech-to-speech instruction pairs.

Lastly, we observe that training solely on speech-to-speech data can impair the model's ability to respond to text inputs: while the language model remains capable of generating reasonable text, the speech decoder fails to produce valid speech tokens when conditioned on text instruction. To mitigate this, we extract text-to-speech instruction samples from the general speech-to-speech dataset, ensuring the model can jointly handle both text and speech inputs during inference.

4 Evaluation

4.1 Speech-to-Text Chat

We evaluate the ability of `OpenS2S` to engage in speech-to-text conversations based on audio input using two benchmarks:

- **VoiceBench** [Chen et al., 2024b] is a benchmark designed for the multi-faceted evaluation of LLM-based voice assistants. To assess the crucial capability of instruction-following, we utilize the alpacaeval, commoneval, wildvoice, and ifeval subsets. These were specifically chosen to test the model's ability to comprehend and accurately execute diverse spoken commands.
- **URO-Bench** [Yan et al., 2025] is an end-to-end benchmark for spoken dialogue models that assesses understanding, reasoning, and oral conversation skills, including paralinguistic cues. To evaluate the model's capacity

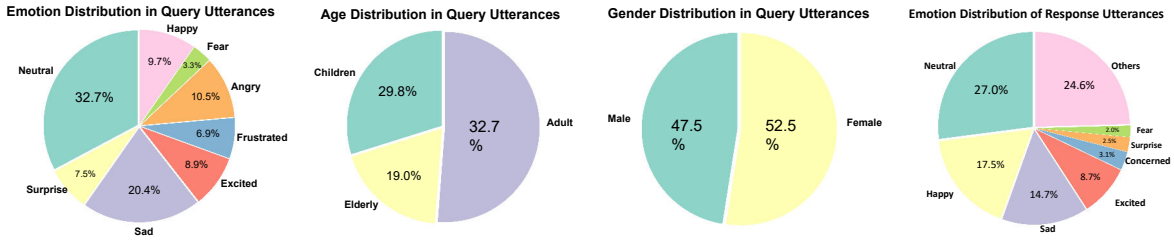


Figure 3: Distributions of emotion, age, and gender in query utterances, and emotion distribution in response utterances.

Table 2: Performance of OpenS2S and baseline models on the benchmarks of speech-to-text chat. For LLaMA-Omni2, We compare with LLaMA-Omni2-7B-Bilingual for its comparable parameter size and bilingual capability.

Model	VoiceBench				URO-Bench	
	alpaca	common	ifeval	wildvoice	underemo-en	underemo-zh
Qwen-2-Audio [Chu et al., 2024]	3.74	3.43	26.33	3.01	35.38	69.62
GLM-4-Voice [Zeng et al., 2024]	3.97	3.42	25.92	3.18	52.41	74.51
Kimi-Audio [Ding et al., 2025]	4.46	3.97	61.10	4.20	59.22	76.96
LLaMA-Omni2 [Fang et al., 2025]	3.96	3.46	17.36	3.07	39.46	63.79
OpenS2S	4.09	3.65	42.89	3.66	46.90	67.68

for empathy, we employ its UnderEmotion-en and UnderEmotion-zh subsets. These are designed to measure the model's ability to perceive the user's emotional state and generate affectively appropriate responses in both English and Chinese.

These two benchmarks evaluate from multiple perspectives to ensure comprehensive assessment: VoiceBench assesses the model's instruction-following capability, while URO-Bench evaluates its comprehension and response to paralinguistic emotional cues. We also evaluate several mainstream open-source speech large models with comparable parameter sizes to OpenS2S for comparison. The results are presented in Table 2.

The results in Table 2 show that OpenS2S demonstrates competitive performance across the four subsets of VoiceBench. Its scores rank second only to Kimi-Audio, which is trained on substantially more data[1], and outperform all other models. These findings indicate that OpenS2S possesses strong capabilities in spoken dialogue and can effectively handle user voice command inputs.

In addition, the results on the URO-Bench subsets show that OpenS2S achieves scores close to those of state-of-the-art models in empathy evaluation, despite being trained on significantly less data. This not only confirms the solid empathetic interaction capabilities of OpenS2S, but also highlights the high quality of the data generated by the proposed empathetic speech dialogue data generation method.

[1]For example, Kimi-Audio employs more 13 million hours of audio data for pre-training.

4.2 Speech-to-Speech Chat

Finally, we assess the end-to-end speech conversation capabilities of OpenS2S based on qualitative analysis. Visit https://cwang621.github.io/omnispeech.github.io/ for demos.

5 Conclusion

This report presents OpenS2S, a fully open-source, end-to-end LSLM specifically designed for empathetic speech interactions. OpenS2S distinguishes itself with an efficient streaming interleaved decoding architecture, enabling low-latency response generation, and an innovative automated data construction pipeline. This pipeline cost-effectively synthesizes diverse, high-quality empathetic speech dialogues by leveraging large language models and controllable text-to-speech systems. As a result, OpenS2S achieves competitive performance in empathetic interactions while requiring substantially less data and computational resources compared to current resource-intensive pre-training methods. We release the complete OpenS2S framework, including the dataset,

model weights, and training codes, to empower the broader research community and accelerate innovation in empathetic speech systems.

Limitations

Despite its advancements, OpenS2S still faces several challenges. Its empathetic performance heavily depends on the quality and diversity of its synthetic training data, which means it may struggle with real-world complexities if the data is insufficiently rich. Although it strives for natural interaction, it ultimately simulates empathy rather than genuinely experiencing it, limiting its ability to provide deeper cognitive understanding. Accurately recognizing subtle emotional nuances in speech also remains an ongoing hurdle. In addition, OpenS2S is trained with low-cost data, and although its performance is acceptable, it still leaves considerable room for improvement. In the future, we introduce larger-scale data synthesis to produce more advanced versions. Finally, like all AI models, it risks inheriting or amplifying biases present in its underlying data sources, which could lead to stereotypical or inappropriate responses toward certain user groups.

Ethical Considerations

The ethical implications of OpenS2S are significant. A primary concern is user trust: if people perceive the AI's empathy as artificial, their confidence in the system could be undermined. There is also the risk of misuse, where its empathetic capabilities might be exploited for malicious purposes such as emotional manipulation. Data privacy remains paramount, as even synthetic speech data could mimic real user patterns, necessitating strict safeguards. Defining accountability for inappropriate or harmful AI responses is crucial, particularly in sensitive domains like mental health. Finally, ensuring cultural sensitivity is vital, as empathetic expressions vary widely across cultures, and a one-size-fits-all approach could lead to misunderstandings or offense.

Acknowledgement

This work is supported by National Key R&D Program of China 2022ZD0160602 and the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA04080400.

References

Junyi Ao, Yuancheng Wang, Xiaohai Tian, Dekun Chen, Jun Zhang, Lu Lu, Yuxuan Wang, Haizhou Li, and Zhizheng Wu. 2024. Sd-eval: A benchmark dataset for spoken dialogue understanding beyond words. *arXiv preprint arXiv:2406.13340*.

Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. 2019. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*.

Loïc Barrault, Yu-An Chung, Mariano Coria Meglioli, David Dale, Ning Dong, Mark Duppenthaler, Paul-Ambroise Duquenne, Brian Ellis, Hady Elsahar, Justin Haaheim, et al. 2023. Seamless: Multilingual expressive and streaming speech translation. *arXiv preprint arXiv:2312.05187*.

Brant R Burleson. 2003. Emotional support skills. In *Handbook of communication and social interaction skills*, pages 569–612. Routledge.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. 2008. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42:335–359.

Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, et al. 2021. Gigaspeech: An evolving, multi-domain asr corpus with 10,000 hours of transcribed audio. *arXiv preprint arXiv:2106.06909*.

Qian Chen, Yafeng Chen, Yanni Chen, Mengzhe Chen, Yingda Chen, Chong Deng, Zhihao Du, Ruize Gao, Changfeng Gao, Zhifu Gao, et al. 2025. Minmo: A multimodal large language model for seamless voice interaction. *arXiv preprint arXiv:2501.06282*.

Wenxi Chen, Ziyang Ma, Ruiqi Yan, Yuzhe Liang, Xiquan Li, Ruiyang Xu, Zhikang Niu, Yanqiao Zhu, Yifan Yang, Zhanxun Liu, et al. 2024a. Slam-omni: Timbre-controllable voice interaction system with single-stage training. *arXiv preprint arXiv:2412.15649*.

Yiming Chen, Xianghu Yue, Chen Zhang, Xiaoxue Gao, Robby T. Tan, and Haizhou Li. 2024b. Voicebench: Benchmarking llm-based voice assistants. *arXiv preprint arXiv:2410.17196*.

Yirong Chen, Xiaofen Xing, Jingkai Lin, Huimin Zheng, Zhenyu Wang, Qi Liu, and Xiangmin Xu. 2023. Soulchat: Improving llms' empathy, listening, and comfort abilities through fine-tuning with multi-turn empathy conversations. *arXiv preprint arXiv:2311.00273*.

Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. 2024. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*.

Yunfei Chu, Jin Xu, Xiaohuan Zhou, Qian Yang, Shiliang Zhang, Zhijie Yan, Chang Zhou, and Jingren Zhou. 2023. Qwen-audio: Advancing universal audio understanding via unified large-scale audio-language models. *arXiv preprint arXiv:2311.07919*.

Wenqian Cui, Dianzhi Yu, Xiaoqi Jiao, Ziqiao Meng, Guangyan Zhang, Qichao Wang, Yiwen Guo, and Irwin King. 2024. Recent advances in speech language models: A survey. *arXiv preprint arXiv:2410.03751*.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*.

Ding Ding, Zeqian Ju, Yichong Leng, Songxiang Liu, Tong Liu, Zeyu Shang, Kai Shen, Wei Song, Xu Tan, Heyi Tang, et al. 2025. Kimi-audio technical report. *arXiv preprint arXiv:2504.18425*.

Zhihao Du, Yuxuan Wang, Qian Chen, Xian Shi, Xiang Lv, Tianyu Zhao, Zhifu Gao, Yexin Yang, Changfeng Gao, Hui Wang, et al. 2024. Cosyvoice 2: Scalable streaming speech synthesis with large language models. *arXiv preprint arXiv:2412.10117*.

Qingkai Fang, Shoutao Guo, Yan Zhou, Zhengrui Ma, Shaolei Zhang, and Yang Feng. 2024. Llama-omni: Seamless speech interaction with large language models. *arXiv preprint arXiv:2409.06666*.

Qingkai Fang, Yan Zhou, Shoutao Guo, Shaolei Zhang, and Yang Feng. 2025. Llama-omni2: Llm-based real-time spoken chatbot with autoregressive streaming speech synthesis. *arXiv preprint arXiv:2505.02625*.

Raman Goel, Seba Susan, Sachin Vashisht, and Armaan Dhanda. 2021. Emotion-aware transformer encoder for empathetic dialogue generation. In *2021 9th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 1–6. IEEE.

Hongkun Hao, Long Zhou, Shujie Liu, Jinyu Li, Shujie Hu, Rui Wang, and Furu Wei. 2023. Boosting large language model for speech synthesis: An empirical study. *arXiv preprint arXiv:2401.00246*.

Michael Hassid, Tal Remez, Tu Anh Nguyen, Itai Gat, Alexis Conneau, Felix Kreuk, Jade Copet, Alexandre Defossez, Gabriel Synnaeve, Emmanuel Dupoux, et al. 2023. Textually pretrained speech language models. *Advances in Neural Information Processing Systems*, 36:63483–63501.

Haorui He, Zengqiang Shang, Chaoren Wang, Xuyuan Li, Yicheng Gu, Hua Hua, Liwei Liu, Chen Yang, Jiaqi Li, Peiyang Shi, et al. 2024. Emilia: An extensive, multilingual, and diverse speech dataset for large-scale speech generation. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pages 885–890. IEEE.

Shujie Hu, Long Zhou, Shujie Liu, Sanyuan Chen, Lingwei Meng, Hongkun Hao, Jing Pan, Xunying Liu, Jinyu Li, Sunit Sivasankaran, et al. 2024. Wavllm: Towards robust and adaptive speech large language model. *arXiv preprint arXiv:2404.00656*.

Ailin Huang, Boyong Wu, Bruce Wang, Chao Yan, Chen Hu, Chengli Feng, Fei Tian, Feiyu Shen, Jingbei Li, Mingrui Chen, et al. 2025. Step-audio: Unified understanding and generation in intelligent speech interaction. *arXiv preprint arXiv:2502.11946*.

Zhichao Huang, Rong Ye, Tom Ko, Qianqian Dong, Shanbo Cheng, Mingxuan Wang, and Hang Li. 2023. Speech translation with large language models: An industrial practice. *arXiv preprint arXiv:2312.13585*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Shengpeng Ji, Yifu Chen, Minghui Fang, Jialong Zuo, Jingyu Lu, Hanting Wang, Ziyue Jiang, Long Zhou, Shujie Liu, Xize Cheng, et al. 2024. Wavchat: A survey of spoken dialogue models. *arXiv preprint arXiv:2411.13577*.

Anjuli Kannan, Arindrima Datta, Tara N Sainath, Eugene Weinstein, Bhuvana Ramabhadran, Yonghui Wu, Ankur Bapna, Zhifeng Chen, and Seungji Lee. 2019. Large-scale multilingual speech recognition with a streaming end-to-end model. *arXiv preprint arXiv:1909.05330*.

Hamza Kheddar, Mustapha Hemis, and Yassine Himeur. 2024. Automatic speech recognition using advanced deep learning approaches: A survey. *Information Fusion*, page 102422.

Yoon Kyung Lee, Inju Lee, Minjung Shin, Seoyeon Bae, and Sowon Hahn. 2023. Chain of empathy: Enhancing empathetic response of large language models based on psychotherapy models. *arXiv preprint arXiv:2311.04915*.

Yinghao Aaron Li, Cong Han, Vinay Raghavan, Gavin Mischler, and Nima Mesgarani. 2023. Styletts 2: Towards human-level text-to-speech through style diffusion and adversarial training with large speech language models. *Advances in Neural Information Processing Systems*, 36:19594–19621.

Siyang Liu, Chujie Zheng, Orianna Demasi, Sahand Sabour, Yu Li, Zhou Yu, Yong Jiang, and Minlie Huang. 2021. Towards emotional support dialog systems. *arXiv preprint arXiv:2106.01144*.

Run Luo, Ting-En Lin, Haonan Zhang, Yuchuan Wu, Xiong Liu, Min Yang, Yongbin Li, Longze Chen, Jiaming Li, Lei Zhang, et al. 2025. Openomni:

Large language models pivot zero-shot omnimodal alignment across language with real-time self-aware emotional speech synthesis. *arXiv preprint arXiv:2501.04561*.

Ziyang Ma, Yakun Song, Chenpeng Du, Jian Cong, Zhuo Chen, Yuping Wang, Yuxuan Wang, and Xie Chen. 2025. Language model can listen while speaking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24831–24839.

Sylvia A Morelli, Matthew D Lieberman, and Jamil Zaki. 2015. The emerging study of positive empathy. *Social and Personality Psychology Compass*, 9(2):57–68.

Eliya Nachmani, Alon Levkovitch, Roy Hirsch, Julian Salazar, Chulayuth Asawaroengchai, Soroosh Mariooryad, Ehud Rivlin, RJ Skerry-Ryan, and Michelle Tadmor Ramanovich. 2023. Spoken question answering and speech continuation using spectrogram-powered llm. *arXiv preprint arXiv:2305.15255*.

Tu Anh Nguyen, Benjamin Muller, Bokai Yu, Marta R Costa-Jussa, Maha Elbayad, Sravya Popuri, Christophe Ropers, Paul-Ambroise Duquenne, Robin Algayres, Ruslan Mavlyutov, et al. 2025. Spirit-lm: Interleaved spoken and written language model. *Transactions of the Association for Computational Linguistics*, 13:30–52.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2018. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *arXiv preprint arXiv:1810.02508*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2018. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*.

Paul K Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, et al. 2023. Audiopalm: A large language model that can speak and listen. *arXiv preprint arXiv:2306.12925*.

Yu Shu, Siwei Dong, Guangyao Chen, Wenhao Huang, Ruihua Zhang, Daochen Shi, Qiqi Xiang, and Yemin Shi. 2023. Llasm: Large language and speech model. *arXiv preprint arXiv:2308.15930*.

Changli Tang, Wenyi Yu, Guangzhi Sun, Xianzhao Chen, Tian Tan, Wei Li, Lu Lu, Zejun Ma, and Chao Zhang. 2023. Salmonn: Towards generic hearing abilities for large language models. *arXiv preprint arXiv:2310.13289*.

Chen Wang, Minpeng Liao, Zhongqiang Huang, Jinliang Lu, Junhong Wu, Yuchen Liu, Chengqing Zong, and Jiajun Zhang. 2023. Blsp: Bootstrapping language-speech pre-training via behavior alignment of continuation writing. *arXiv preprint arXiv:2309.00916*.

Chen Wang, Minpeng Liao, Zhongqiang Huang, Junhong Wu, Chengqing Zong, and Jiajun Zhang. 2024a. Blsp-emo: Towards empathetic large speech-language models. *arXiv preprint arXiv:2406.03872*.

Chen Wang, Minpeng Liao, Zhongqiang Huang, and Jiajun Zhang. 2024b. Blsp-kd: Bootstrapping language-speech pre-training via knowledge distillation. *arXiv preprint arXiv:2405.19041*.

Kaisiyuan Wang, Qianyi Wu, Linsen Song, Zhuoqian Yang, Wayne Wu, Chen Qian, Ran He, Yu Qiao, and Chen Change Loy. 2020. Mead: A large-scale audio-visual dataset for emotional talking-face generation. In *European Conference on Computer Vision*, pages 700–717. Springer.

Xiong Wang, Yangze Li, Chaoyou Fu, Yunhang Shen, Lei Xie, Ke Li, Xing Sun, and Long Ma. 2024c. Freeze-omni: A smart and low latency speech-to-speech dialogue model with frozen llm. *arXiv preprint arXiv:2411.00774*.

Zhifei Xie and Changqiao Wu. 2024. Mini-omni: Language models can hear, talk while thinking in streaming. *arXiv preprint arXiv:2408.16725*.

Hongfei Xue, Yuhao Liang, Bingshen Mu, Shiliang Zhang, Mengzhe Chen, Qian Chen, and Lei Xie. 2024. E-chat: Emotion-sensitive spoken dialogue system with large language models. In *2024 IEEE 14th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 586–590. IEEE.

Ruiqi Yan, Xiquan Li, Wenxi Chen, Zhikang Niu, Chen Yang, Ziyang Ma, Kai Yu, and Xie Chen. 2025. Uro-bench: A comprehensive benchmark for end-to-end spoken dialogue models. *arXiv preprint arXiv:2502.17810*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2018. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2236–2246.

Aohan Zeng, Zhengxiao Du, Mingdao Liu, Kedong Wang, Shengmin Jiang, Lei Zhao, Yuxiao Dong, and Jie Tang. 2024. Glm-4-voice: Towards intelligent and human-like end-to-end spoken chatbot. *arXiv preprint arXiv:2412.02612*.

Binbin Zhang, Hang Lv, Pengcheng Guo, Qijie Shao, Chao Yang, Lei Xie, Xin Xu, Hui Bu, Xiaoyu Chen, Chenchen Zeng, et al. 2022. Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6182–6186. IEEE.

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. 2023. Speechgpt: Empowering large language models with intrinsic cross-modal conversational abilities. *arXiv preprint arXiv:2305.11000*.

Dong Zhang, Xin Zhang, Jun Zhan, Shimin Li, Yaqian Zhou, and Xipeng Qiu. 2024. Speechgpt-gen: Scaling chain-of-information speech generation. *arXiv preprint arXiv:2401.13527*.

Kun Zhou, Berrak Sisman, Rui Liu, and Haizhou Li. 2022. Emotional voice conversion: Theory, databases and esd. *Speech Communication*, 137:1–18.

A Training Strategy

The training of `OpenS2S` consists of three stages: speech understanding pre-training, speech generation pre-training, and empathy speech instruction fine-tuning. In the first two pre-training stages, we utilize open-source ASR and TTS datasets for pretraining to endow the model with robust speech understanding and generation capabilities. In the instruction fine-tuning stage, we construct an empathy speech instruction dataset for fine-tuning, enabling the model to understand the semantic content and paralinguistic cues in speech, and finally generate empathic speech responses.

Speech Understanding Pretraining (Stage 1 in Figure 4) To equip the model with robust speech understanding capabilities, we perform pretraining on large-scale speech-text paired corpora, following the training paradigm of BLSP-Emo [Wang et al., 2024a]. This stage is divided into two phases: semantic alignment and emotional alignment.

In the semantic alignment phase, we adopt the concept of behavioral alignment, requiring the LLM to produce identical continuations when given either speech or its corresponding transcript as input. Specifically, we first prompt the LLM to generate continuations based on text transcripts from an ASR dataset. During training, the model is required to produce the same continuation when conditioned on continuous speech representations, thus aligning the model's behavior across modalities.

In the emotional alignment phase, we leverage an SER dataset where each transcript is annotated with an emotion label. An LLM is prompted to generate emotion-aware continuations based on the transcript and the reference emotion. We then adapt a speech-language model to generate similar continuations directly from the speech input. This step encourages the model to comprehend and reflect both linguistic semantics and paralinguistic emotional cues, producing text responses aligned with those generated by the LLM given identical content and emotion labels.

Throughout this pretraining process, the parameters of the audio encoder and the LLM remain frozen. Only the speech adapter is fine-tuned to facilitate modality bridging.

Speech Generation Pretraining (Stage 2 in Figure 4) As the streaming speech decoder is initialized from Qwen3-1.8B, which is originally designed for text generation, we first perform offline TTS pretraining to enable it to generate discrete speech tokens. Specifically, we expand the decoder's vocabulary to include 16,384 speech tokens. During this phase, the input text is embedded using word embeddings as a prefix, and the target is the sequence of speech tokens extracted from reference audio. This allows the decoder to learn a basic mapping from text tokens to speech tokens, serving as a foundation for downstream speech synthesis.

In the second stage, we further train the speech decoder to integrate with the LLM for streaming generation. Unlike the previous step, the input text is no longer embedded directly into the speech decoder. Instead, it is first processed by the LLM using a structured prompt. The final-layer hidden states corresponding to the response portion are then extracted and interleaved with speech tokens during training. At this stage, the LLM's parameters are kept frozen, while the linear projection layer and the speech decoder are fine-tuned. This training strategy not only bridges the LLM and the speech decoder but also adapts the offline decoder into a streaming-capable model that supports interleaved token generation.

Empathetic Speech Instruction Tuning (Stage 3 in Figure 4) Following the pretraining stages, the model demonstrates general speech understanding

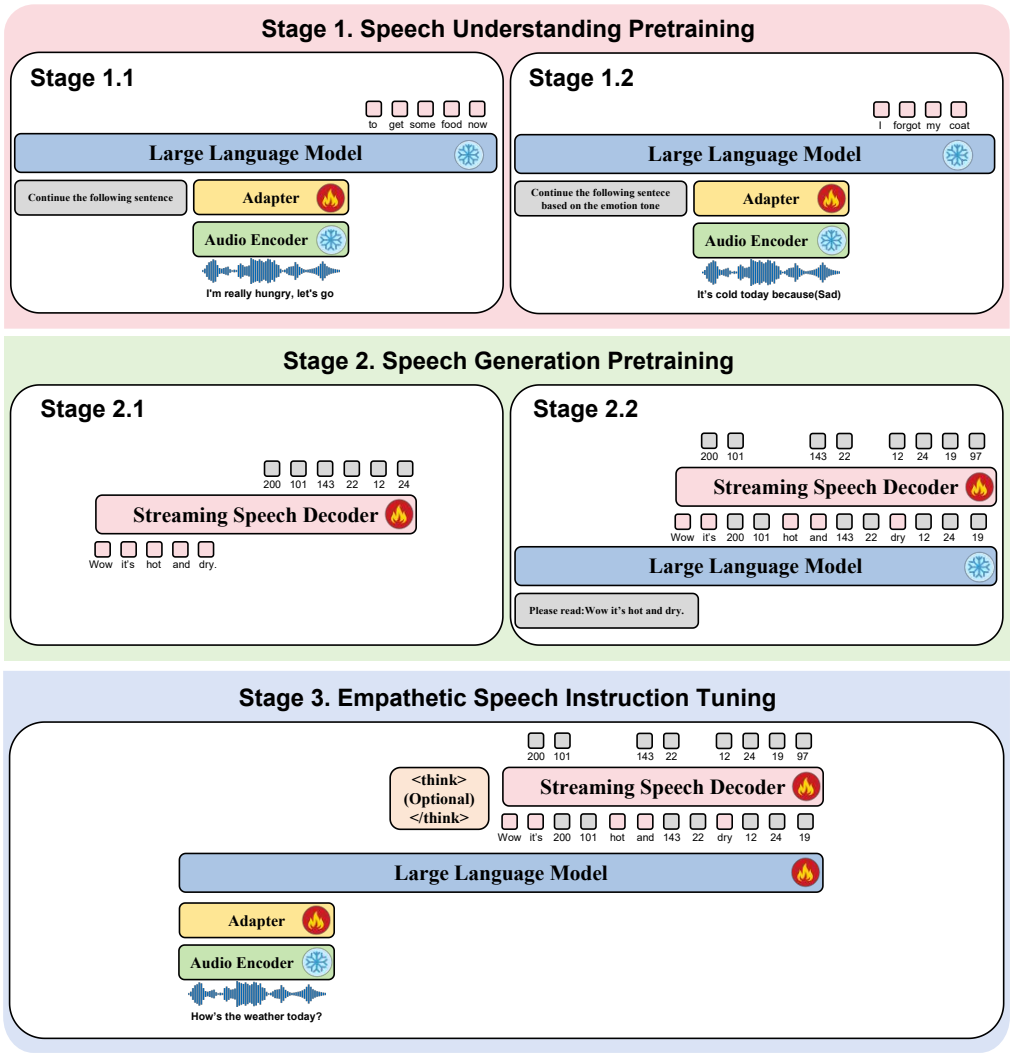


Figure 4: The training process of OpenS2S.

capabilities, enabling it to generate empathetic text responses conditioned on both semantic content and emotional cues in speech. However, its speech generation ability remains limited: the model is only able to produce meaningful speech tokens when explicitly prompted with speech synthesis tasks. When handling general-purpose text instructions or directly responding to speech instructions, the speech decoder often fails to generate coherent or meaningful speech outputs.

We attribute this limitation to overfitting during the TTS-based pretraining stage. Specifically, the model learns to rely on a narrow representation subspace defined by the TTS task, resulting in poor generalizability to broader instruction-following scenarios. To address this issue, we introduce an additional instruction tuning stage aimed at enabling robust and flexible speech generation across diverse task types.

In this stage, the speech encoder remains frozen, while all other components, including audio adapter, the LLM, linear projection layer, and speech decoder, are fully fine-tuned. We observe that relying solely on speech-to-speech instruction data is insufficient to generalize speech generation to textual instructions. Therefore, we further incorporate text-to-speech instruction data, allowing the model to handle both speech and text inputs seamlessly. The construction process of these instruction datasets is detailed in Section 3.2.

B Related Work

B.1 Speech Language Models

With the rapid advancement of large language models (LLMs), there is growing interest in extending their capabilities to spoken language, giving rise to Speech Language Models (SpeechLMs) that can understand and/or generate speech [Cui

et al., 2024, Ji et al., 2024]. One approach directly adapts LLMs for *end-to-end speech modeling* by converting speech into discrete tokens and expanding the vocabulary, as seen in SpeechGPT [Zhang et al., 2023], AudioPaLM [Rubenstein et al., 2023], and TWIST [Hassid et al., 2023]. Recent models like Spirit-LM [Nguyen et al., 2025] and GLM-4-Voice [Zeng et al., 2024] leverage interleaved speech-text training, while others such as Moshi [Défossez et al., 2024] and LSLM [Ma et al., 2025] enable spoken dialogue. In contrast, modular SpeechLMs *connect LLMs with external speech modules*. Early works [Shu et al., 2023, Wang et al., 2023, 2024b, Chu et al., 2023, Nachmani et al., 2023, Tang et al., 2023, Chu et al., 2024, Hu et al., 2024] focused on speech understanding by connecting pretrained speech encoders to LLMs, but did not support speech generation. In contrast, more recent models such as LLaMA-Omni [Fang et al., 2024, 2025], Freeze-Omni [Wang et al., 2024c], and OpenOmni [Luo et al., 2025] overcome this limitation by attaching speech decoders to LLM outputs. Mini-Omni [Xie and Wu, 2024] and SLAM-Omni [Chen et al., 2024a] go further with parallel decoding. Minmo [Chen et al., 2025] and LLaMA-Omni2 [Fang et al., 2025] incorporates a streaming speech decoder through interleaved text-speech generation.

B.2 Empathetic Conversations Across Modalities

Empathetic conversation modeling [Rashkin et al., 2018, Liu et al., 2021] has been studied across *text-to-text, speech-to-text, and speech-to-speech* settings, aiming to equip LLMs with emotional understanding and supportive responses [Burleson, 2003]. In text-based interactions, early work focused on architecture modifications [Goel et al., 2021], while recent approaches like SoulChat [Chen et al., 2023] and Chain of Empathy prompting [Lee et al., 2023] enhance empathy through fine-tuning or step-by-step reasoning without extra data. For speech-to-text interaction, E-chat [Xue et al., 2024] introduced an emotion-aware speech instruction dataset to enhance LLMs' understanding of emotional speech. BLSP-Emo [Wang et al., 2024a] proposed an end-to-end model that aligns speech semantics and emotions through two-stage pretraining using ASR and SER datasets. Moving toward speech-to-speech empathy, Spoken-GPT [Zhang et al., 2024] adopts a cascaded framework that listens and responds with expressive, emotionally attuned speech, paving the way for fully empathetic voice agents. Advanced commercial models such as GPT-4o [Hurst et al., 2024], Doubao, Kimi-Audio [Ding et al., 2025], and Step-Audio [Huang et al., 2025] push the boundaries of empathetic interaction by incorporating paralinguistic cues to better perceive and respond to users' emotional states. These models integrate speech understanding and generation in real time, enabling more natural and emotionally aware human-computer interactions. However, our `OpenS2S` is the first to release all the resources including model weights, training data and training codes, in order to boost the research in the community.

PDFMathTranslate: Scientific Document Translation Preserving Layouts

Rongxin Ouyang[1,†], **Chang Chu**[2,†,*], **Zhikuang Xin**[3], **Xiangyao Ma**[4]
[1]National University of Singapore, Singapore. [2]Tsinghua University, Beijing, China.
[3]University of Chinese Academy of Sciences, Beijing, China.
[4]Funstory.ai Limited, Hong Kong SAR, China.
[†] These authors contribute equally to this work.
[*]**Correspondence:** chuc23@mails.tsinghua.edu.cn.

Abstract

Language barriers in scientific documents hinder the diffusion and development of science and technologies. However, prior efforts in translating such documents largely overlooked the information in layouts. To bridge the gap, we introduce **PDFMathTranslate**, the world's first open-source software for translating scientific documents while preserving layouts. Leveraging the most recent advances in large language models and precise layout detection, we contribute to the community with key improvements in precision, flexibility, and efficiency. The work is open-sourced at https://github.com/byaidu/pdfmathtranslate with more than 222k downloads.

1 Introduction

Science and technologies diffuse and develop in different languages (Von Gizycki, 1973; Montgomery, 2013). Yet, language barriers hinder the diffusion and development of scientific progress (Ramírez-Castañeda, 2020; Hwang, 2005; Ammon, 2012). For example, while 98% publications in science are written in English (Liu; Ammon, 2012), the language only has 7.3% native speakers and no more than 20% speakers worldwide (Bahji et al., 2023). Consequently, the majority of human beings are hindered by language barriers from the advances in science and technologies. To overcome the barrier, there are the attempts from international organizations for inclusive science [1] and the attempts from academia in improving *Machine Translation (MT)* over text (Brown et al., 1990; Vaswani et al., 2017; Zhu et al., 2020; Johnson et al., 2017; Sennrich et al., 2016).

However, text-based machine translation fails to address the unique challenges posed by the layouts in *technical translation* (Schubert, 2012) where the *elaborate access structure* matters (pp. 351-352). Non-textual elements in scientific and technical documents are not ignorable — the arrangement of paragraphs, mathematical equations, tables, and figures have rich and important meanings. Thus, text-based machine translation ignoring such important information appear insufficient to address the barrier in scientific and technical document translation, hindering the progress of science, technologies, and society.

To fill the gap, we introduce **PDFMathTranslate**, the world's first open-source tool to translate PDF documents with preserved layouts. By leveraging recent advances in layout detection and large language models (see Figure 1), we better address the barrier with at least five key contributions: (1) efficient workflow of layout detection, translation, and re-rendering; (2) support for multiple languages; (3) support for multiple translation models and services; (4) diverse user interfaces; and (5) a community-commerce model affording sustainable developments.

2 Architecture and Design

PDFMathTranslate is designed to translate documents while preserving their original layouts. First, it accepts a PDF document along with user-specified parameters, such as languages and the preferred translation service. Next, it detects the layouts to extract the layouts and textual contents from the document. Third, the texts are translated using the selected translation service, such as GPT-4, DeepL, Google, or Ollama (see Appendix A). Finally, the translated text and previously detected layouts are re-rendered as a translated document with preserved layouts.

Technically, we design the architecture for *precision*, *flexibility*, and *efficiency*. These three principles are realized by a precise parser, a flexible translation middle-ware, and an efficient workflow, detailed in following subsections.

[1]See https://www.unesco.org/en/open-science

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 918–924

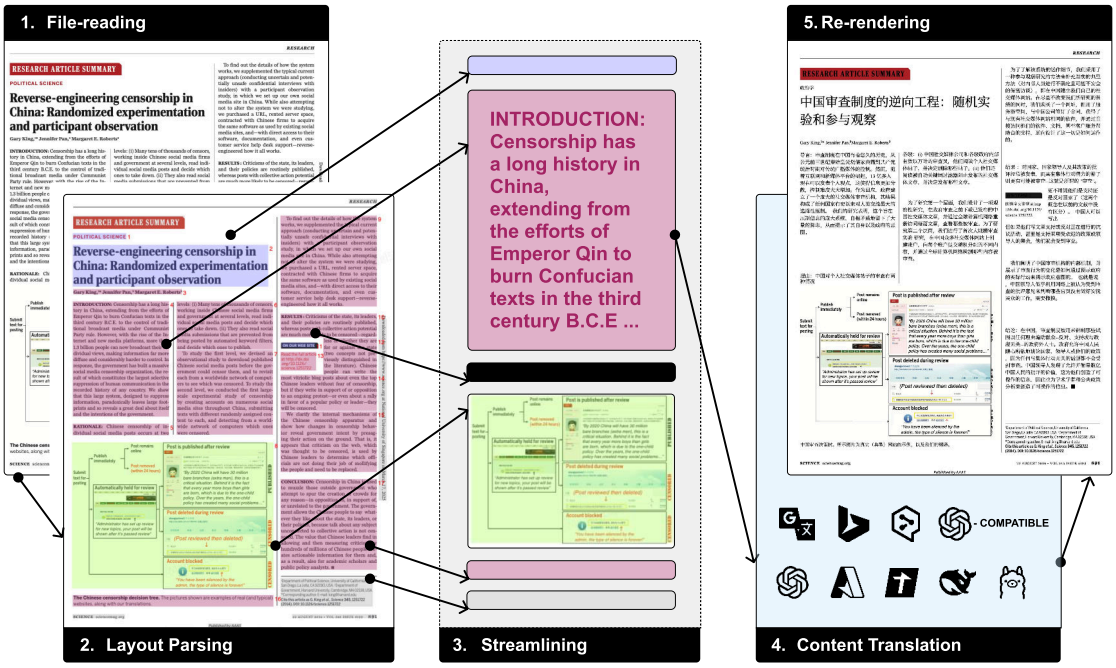


Figure 1: The architecture of PDFMathTranslate

2.1 Precise layout parser

Preserving layouts requires precise layout extraction. To precisely parse the position of document elements, we propose a pipeline consisting of layout detection, splitting, processing, and re-rendering. In the beginning, we exploit a recent advance in layout detection, *DocLayout-YOLO-DocStructBench-onnx*. The model is another version of a SOTA solution in object detection, *Yolov10* (Wang et al., 2024), and is fast and accurate in the specific task (Zhao et al., 2024).

To increase the compatibility of this model, we support two model formats: the *Open Neural Network Exchange (ONNX)* [2] standard and a *pytorch* version. The ONNX version was chosen as default to ensure the compatibility of our parsing pipeline for diverse hardware.

2.2 Flexible translation middleware

Translation middleware offers flexibility in terms of *language diversity*, *supported services*, and *customization capabilities*. One, PDFMathTranslate supports at least 194 languages (see Appendix A), including popular ones such as *English*, and some languages shared by relatively smaller communities, such as *Cherokee*. Users can freely select any of these languages as either the input or output language when translating documents.

Two, PDFMathTranslate supports at least 23 popular services for translation. The list of services includes prominent online services such as Google Translate and OpenAI, as well as local deployments of models for either translation or dialogs (see Appendix A). In addition to explicitly listed services, users can integrate any new service complying with OpenAI protocols [3].

Moreover, to simplify the addition of new services, we designed a structure that separates layout flows from translation flows. In this structure, language services receive only textual inputs, enhancing the generalizability and sustainability of our project. As a result, implementing any language service requires only a concise function of fewer than 15 lines.

Three, PDFMathTranslate provides additional customizability in translation. We designed a customization feature prompting strategies that potentially increase the quality of translation or its adaptability in domain-specific tasks. This feature allows users to exploit advanced prompting strategies such as few-shots (Brown et al., 2020), Chain-of-Thought (Wei et al., 2022), Role-playing (Kong

[2] See `https://onnx.ai/`

[3] See: https://platform.openai.com/docs/api-reference/chat/create

et al., 2024; Shanahan et al., 2023), etc.

2.3 Efficient streaming flow

Translating scientific and technical documents often necessitates an efficient workflow, especially when dealing with thousands of pages, which can challenge overall performance. To address the issue, we landed on a streaming design that is centered around an efficient in-memory processing pipeline that minimizes disk I/O while preserving document fidelity. Specifically, the `translate_stream` function accepts a byte stream of a PDF file and transforms it into a mutable document representation. This approach allows the system to dynamically apply modifications, such as converting to PDF/A format when needed and embedding the appropriate fonts based on the target language. By employing temporary buffers and leveraging functions like `download_remote_fonts`, the design effectively manages resources and ensures that the original file remains unaltered during processing.

Our workflow supports various translation backends, with translators configured as plugins within our workflow. Different translators integrate distinct translation functionalities by inheriting from a base class and implementing the `do_translate` method. These translators can be seamlessly integrated into different user interfaces. With contributions from the community, our workflow currently supports more than 20 translators. Additionally, it provides robust extensibility for backend LLM services with the OpenAI-compatible API and traditional machine translation service.

Further optimization in performance is achieved through a combination of asynchronous execution, caching strategies, and error-handling mechanisms. Asynchronous constructs, such as `asyncio.Event`, are employed to support task cancellation and concurrency, particularly when processing large documents or handling multiple translation tasks simultaneously. In addition, the design incorporates a caching mechanism—controlled via the `ignore_cache` flag—to prevent redundant computations, save LLM tokens, and accelerate the translation process. While a subset font embedding strategy minimizes the final file size without compromising compatibility. Overall, this streaming design reflects a robust, scalable, and resource-efficient framework tailored for automated PDF translation workflows.

Figure 2: Officially supported interfaces

Figure 3: Major commands for CLI

3 Usage and Deployments

PDFMathTranslate has been implemented in various interfaces, including a command-line tool (CLI), a graphic user interface (GUI), cross-platform applications (on MacOS and Windows), and Docker images. In addition to those officially supported interfaces, user `guaguastandup` contributes a Zotero plugin to our community.

For laymen, we offer GUI on Mac [4], Windows, Web, online demos, and Zotero plugin [5]. Specifically, desktop versions on Mac and Windows can be installed, demos are publicly accessible online, and GUI can be started using `pdf2zh -i`. Those graphic interfaces are shown in Figure 2.

For developers, we support CLI and Docker for advanced usage and substream developments. Developers who prefer command-line tools can conveniently install our Python CLI program using `pip install pdf2zh`. The command-line tool supports extensive features, with the major options shown in Figure 3 and more documented in details [6]. For users who want to deploy the software on servers, we offer a Docker image for simplified deployment. Additionally, Docker images are built and distributed on different platforms for developers with potential network issues.

```
docker pull byaidu/pdf2zh
```

[4] See https://github.com/reycn/pdf2zh-mac.

[5] See https://github.com/guaguastandup/zotero-pdf2zh; community-contributed.

[6] See https://github.com/Byaidu/PDFMathTranslate/blob/main/docs/ADVANCED.md

```
docker run -d -p 7860:7860 byaidu/
    pdf2zh
```

4 Comparison with Existing Systems

Compared with existing alternatives, PDFMathTranslate offers a superior solution from three perspectives: *accessibility*, *readability*, and *efficiency* (see Table 1).

In terms of *accessibility*, the tool is open-sourced, self-deployable, supports API calls, and is entirely free of charge. These advantages ensure wider accessibility compared to any other existing solutions. Regarding *readability*, the tool preserves layouts, supports complex formulas, and is capable of exporting bilingual documents. Notably, a current limitation of our tool is its lack of optimal optical character recognition (OCR) support, which restricts its wider application for scanned documents. However, OCR functionality is planned for implementation in upcoming versions. Finally, concerning *efficiency*, while this application is slower than text-based translation services (like Google Translate) due to the integration of precise layout detection models, our solution is significantly faster than alternative products that also preserve layouts (such as IMT and Doc2X).

5 Use Cases

We illustrate two typical use cases of PDFMathTranslate to illustrate how the tool translates text while preserving information embedded in layouts. The first use case (Panel A, Figure 4) is an excerpt from a textbook with both text and complex formulas, and the second case (Panel B, Figure 4) is a scientific research article with complex layouts and figures (see Figure 4). Both cases consistently show that our work is capable of precisely translating the information within text while preserving the crucial information embedded in layouts.

6 Sustainability

Nevertheless, maintenance is challenging to open-source projects (Stol and Ali Babar, 2010). To ensure sustainable development, we've established a *community-commerce* model, through which the incentives of developers are provided by two sources: open-source recognition and the benefits of commercial products.

First, we increase developer exposure by prominently featuring contributors on the project's homepage and regularly highlighting recent contributions (e.g., *1970-01-01 / Supports Google translation (by @author_handle)*). This strategy has successfully incentivized 44 global developers who collectively contributed tens of thousands of lines of code.

Second, we provide sponsored rewards such as membership exchanges for active contributors through collaboration with a commercial partner. These 11 subscriptions sent out to developers have effectively encouraged consistent contributions, resulting in the resolution of over 485 user-reported issues (by April 2025). Such cooperation has enabled the integration of advanced academic and community-driven technologies, including ONNX model support (Wybxc, 2025) and Qwen (ws051682, 2025), thus advancing cross-linguistic scientific communication.

This sustainable collaboration model propelled the project to the top of GitHub's global trends for over a week, garnering more than 25k stars, 222k downloads, and over 49k Docker pulls (by June 2025). Additionally, our project's success has inspired a range of new commercial products with similar functionalities, significantly impacting both the scientific community and the broader public.

7 Limitations

While we have made several key improvements, our tool still has specific limitations. First, translation quality remains highly dependent on the underlying translation models and prompts. Second, the accuracy of the layout detection model in identifying various document layouts similarly depends on the quality of the layout detection models employed. Third, the current version of the tool can not handle scanned PDFs lacking optimal optical character recognition (OCR); however, this last feature is planned for implementation.

8 Acknowledgments

The authors thank all 44 contributors[7] of the open-sourced project on GitHub; platforms allowing us

[7] They are 7shi, Byaidu, Copilot, Cycloctane, Hanaasagi, IuvenisSapiens, JEFF-dev-ui, Tql-ws1, Wybxc, YadominJinta, Zxis233, alohays, aseaday, awwaawwa, borcation, charles7668, chiu0602, czz404, damaoooo, dependabot[bot], domonnss, eltociear, hellofinch, highkay, hotwa, imClumsyPanda, kharkover, kidach1, lintian233, mydreamworldpolly, namazuchin, qqueing, reycn, tastelikefeet, timelic, treeleaves30760, tylzh97, ws051682, wx-11, xxnuo, xyzyx233, yidasanqian, ymattw, zqqian; ordered alphabetically, retrieved through the official GitHub API by March 2025)

		OURS	IMT[5]	Doc2X	TeX based[6]	Google	DeepL
Accessibility	Open-source	✓			✓		
	Deployment	✓			✓		
	API integration	✓		✓		✓	✓
	Price	**Free**	Paid	Paid	Free	Free	Paid
Readability	Layout[1]	✓	✓	✓			
	Formula[2]	✓	✓	✓	✓		
	Bilingual[3]	✓	Partial	Partial	✓	✓	✓
	OCR[4]	✓[7]	✓	✓		✓	✓
Efficiency	Batch tasks	✓		✓	✓	✓	✓
	Speed (sec/page)	**1.47**	1.50	1.86	1.67	0.38	1.88

[1] Whether the tool can preserve figures, images, and formulas (although the formulas may be incorrectly displayed).
[2] Whether the tool can translate documents containing formulas and correctly handle the positions of the formulas.
[3] Whether the tool supports bilingual exports where both the original and translated text are simultaneously readable.
[4] Whether the tool can handle scanned documents instead of digital documents.
[5] IMT: Immersive Translate PDF Pro.
[6] The original LaTeX file is required for translation.
[7] Available in a community fork: `PDFMathTranslate/PDFMathTranslate-next`.

Table 1: Comparison with other projects and productions

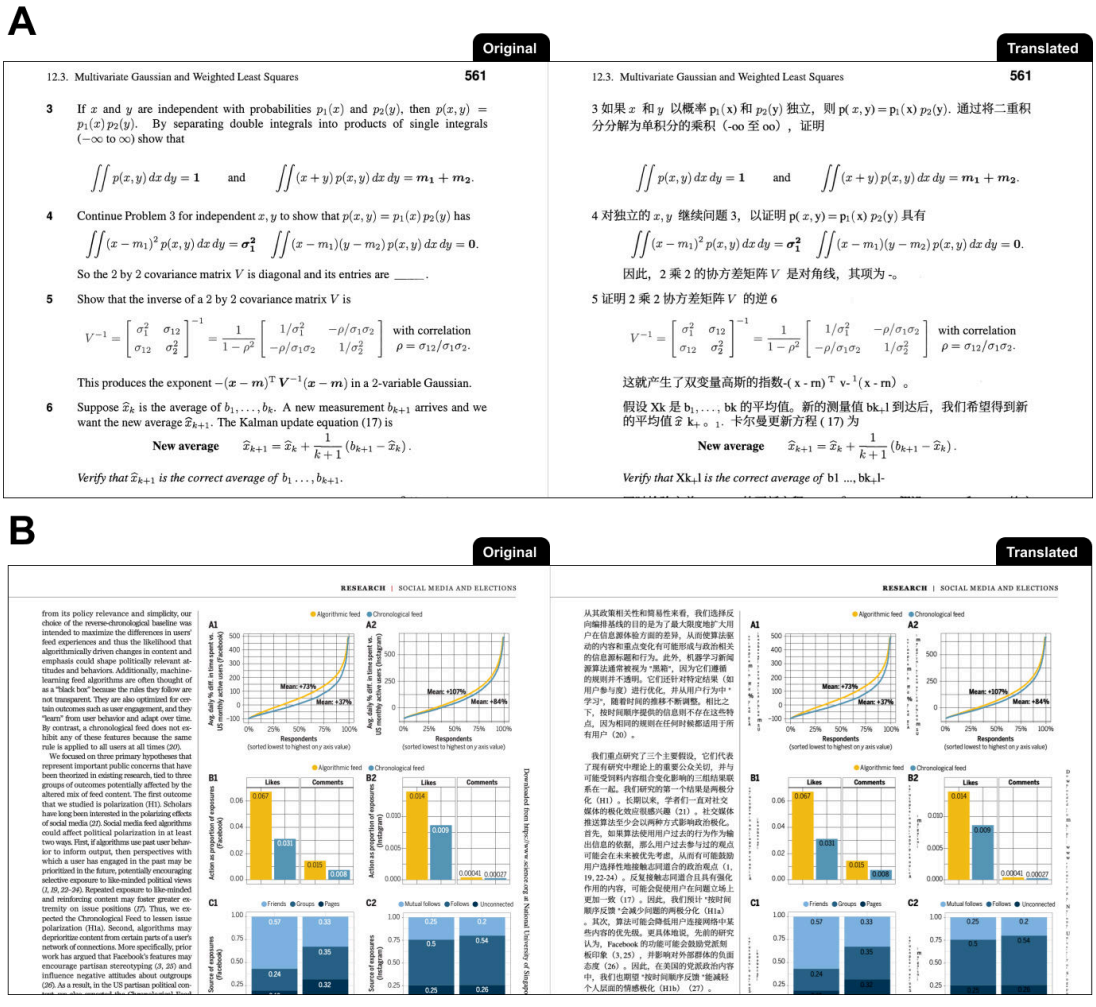


Figure 4: Use cases.

to host demos (Hugging Face, ModelScope); services we used in the demo (e.g., Google Translate); redeem codes from Immersive Translate; public API access from SiliconFlow; and all comments from users.

9 Ethics and Broader Impacts

Our work has made a significant positive impact within both multilingual scientific and open-source communities. However, as a flexible, widely used tool, it raises potential ethical concerns regarding the copyright of documents. Translating documents without proper permissions could challenge the intellectual property rights of scientific works and innovations. We welcome suggestions from experts in intellectual property to mitigate the concern.

References

Ulrich Ammon. 2012. Linguistic inequality and its effects on participation in scientific discourse and on global knowledge accumulation – With a closer look at the problems of the second-rank language communities. *Applied Linguistics Review*, 3(2):333–355.

Anees Bahji, Laura Acion, Anne-Marie Laslett, and Bryon Adinoff. 2023. Exclusion of the non-English-speaking world from the scientific literature: Recommendations for change for addiction journals and publishers. 40(1):6–13.

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. 16(2):79–85.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Kumju Hwang. 2005. The inferior science and the dominant use of english in knowledge production: A case study of korean science and technology. *Science Communication*, 26(4):390–427.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, and Greg Corrado. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. 5:339–351.

Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, Xin Zhou, Enzhi Wang, and Xiaohang Dong. 2024. Better zero-shot reasoning with role-play prompting. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4099–4113.

Weishu Liu. The changing role of non-English papers in scholarly communication: Evidence from Web of Science's three journal citation indexes. 30(2):115–123.

Scott L. Montgomery. 2013. *Does Science Need a Global Language?: English and the Future of Research*. University of Chicago Press.

Valeria Ramírez-Castañeda. 2020. Disadvantages in preparing and publishing scientific papers caused by the dominance of the english language in science: The case of colombian researchers in biological sciences. *PloS one*, 15(9):e0238372.

Klaus Schubert. 2012. Technical translation. In *Handbook of Translation Studies: Volume 1*, pages 350–355. John Benjamins Publishing Company.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. *Preprint*, arXiv:1508.07909.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. 623(7987):493–498.

Klaas-Jan Stol and Muhammad Ali Babar. 2010. Challenges in using open source software in product development: A review of the literature. In *Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, pages 17–22. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Rainald Von Gizycki. 1973. Centre and Periphery in the International Scientific Community: Germany, France and Great Britain in the 19th Century. 11(4):474–494.

Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. 2024. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. 35:24824–24837.

ws051682. 2025. feat(translator): Add Ali Qwen MT translator by ws051682 · Pull Request #585 · Byaidu/PDFMathTranslate. [Online; accessed 18. Mar. 2025].

Wybxc. 2025. feat: onnx support by Wybxc · Pull Request #116 · Byaidu/PDFMathTranslate. [Online; accessed 18. Mar. 2025].

Zhiyuan Zhao, Hengrui Kang, Bin Wang, and Conghui He. 2024. DocLayout-YOLO: Enhancing Document Layout Analysis through Diverse Synthetic Data and Global-to-Local Adaptive Perception. *Preprint*, arXiv:2410.12628.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating BERT into Neural Machine Translation. *Preprint*, arXiv:2002.06823.

A Appendix

A.1 Officially supported language and services

In Table 2, we enumerate the officially supported services, models, and languages. However, the actual number may be significantly higher due to the extensibility of services and models.

Table 2: Supported languages and services for translation in PDFMathTranslate

Category	Details	Total
Service or models	Google *(default)*, 302.AI, Bing, DeepL, DeepLX, Ollama, Ali Qwen, Ollama, Xinference, Gemma, OpenAI, OpenAI-like , DeepSeek, AzureOpenAI, Zhipu, ModelScope, Silicon Cloud, Gemini, Azure, Tencent, Dify, AnythingLLM, Argos Translate, Grok, Groq	*23*
Input and output languages	Abkhaz, Acehnese, Acholi, Afrikaans, Albanian, Alur, Amharic, Arabic, Armenian, Assamese, Awadhi, Aymara, Azerbaijani, Balinese, Bambara, Bashkir, Basque, Batak Karo, Batak Simalungun, Batak Toba, Belarusian, Bemba, Bengali, Betawi, Bhojpuri, Bikol, Bosnian, Breton, Bulgarian, Buryat, Cantonese, Catalan, Cebuano, Chichewa (Nyanja), Chinese (Simplified), Chinese (Traditional), Chuvash, Corsican, Crimean Tatar, Croatian, Czech, Danish, Dinka, Divehi, Dogri, Dombe, Dutch, Dzongkha, English, Esperanto, Estonian, Ewe, Fijian, Filipino (Tagalog), Finnish, French, French (French), French (Canadian), Frisian, Fulfulde, Ga, Galician, Ganda (Luganda), Georgian, German, Greek, Guarani, Gujarati, Haitian Creole, Hakha Chin, Hausa, Hawaiian, Hebrew, Hiligaynon, Hindi, Hmong, Hungarian, Hunsrik, Icelandic, Igbo, Iloko, Indonesian, Irish, Italian, Japanese, Javanese, Kannada, Kapampangan, Kazakh, Khmer, Kiga, Kinyarwanda, Kituba, Konkani, Korean, Krio, Kurdish (Kurmanji), Kurdish (Sorani), Kyrgyz, Lao, Latgalian, Latin, etc.	*194*

[1] The number of supported languages far exceeds what is shown in the table. Our protocol, designed as a general framework for OpenAI-like services, allows users to integrate any translation system, including locally hosted models or private online services, ensuring extensive language support.
[2] Based on the default service (Google Translate) and interface (CLI).

CrowdAgent: Multi-Agent Managed Multi-Source Annotation System

Maosheng Qin[1]*, Renyu Zhu[2]*, Mingxuan Xia[1], Chenkai Chen[3], Zhen Zhu[3], Minmin Lin[2], Junbo Zhao[1], Lu Xu[2], Changjie Fan[2], Runze Wu[2] †, Haobo Wang[1] †
[1]Zhejiang University [2]NetEase Fuxi AI Lab [3]Zhejiang Sci-tech University
qin.mmms@gmail.com, wanghaobo@zju.edu.cn

Abstract

High-quality annotated data is a cornerstone of modern Natural Language Processing (NLP). While recent methods begin to leverage diverse annotation sources—including Large Language Models (LLMs), Small Language Models (SLMs), and human experts—they often focus narrowly on the labeling step itself. A critical gap remains in the holistic **process control** required to manage these sources dynamically, addressing complex scheduling and quality-cost trade-offs in a unified manner. Inspired by real-world crowdsourcing companies, we introduce *CrowdAgent*, a multi-agent system that provides end-to-end process control by integrating task assignment, data annotation, and quality/cost management. It implements a novel methodology that rationally assigns tasks, enabling LLMs, SLMs, and human experts to advance synergistically in a collaborative annotation workflow. We demonstrate the effectiveness of CrowdAgent through extensive experiments on six diverse multimodal classification tasks. The source code and video demo are available at https://github.com/QMMMS/CrowdAgent.

1 Introduction

High-quality annotated data serves as the cornerstone for advancements in machine learning models and is pivotal for the digital transformation of enterprises. To meet the substantial data requirements, manual annotation through crowdsourcing (Li et al., 2017; Zhen et al., 2021; Zhang, 2022) has been a prevalent approach due to its inherent scalability and flexibility. However, this method faces challenges, including high costs, particularly for tasks demanding specialized expertise, such as medical (Johnson et al., 2023; Li et al., 2022) and financial (Chen et al., 2021) tasks, and is labor-intensive for large-scale annotation.

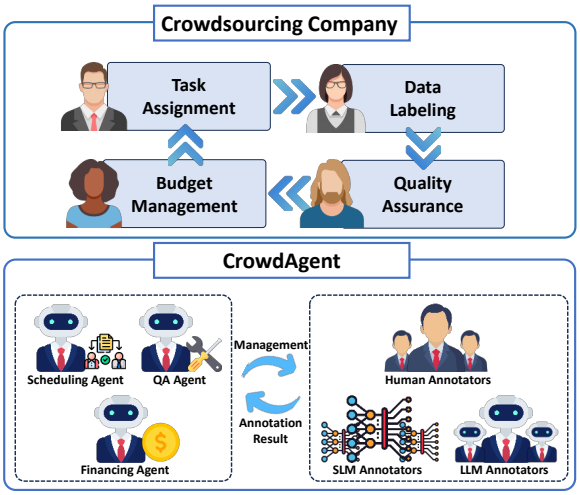


Figure 1: Comparison between traditional crowdsourcing company and our proposed system. CrowdAgent automates human management by using a collaborative multi-agent system, where tasks are dynamically dispatched to diverse annotation sources.

To address this, LLMs-based annotation methods (Ding et al., 2023) have been particularly prevalent due to LLMs' strong zero-shot capabilities that achieve near-human performance while incurring lower costs. Due to inherent biases that can lead to inaccurate results, relying solely on LLMs is often insufficient. To address these shortcomings, recent work has focused on creating hybrid systems that combine multiple annotation sources. For example, Co-Annotating (Li et al., 2023) treats LLMs as diverse annotators by employing a set of 7 prompt types; FreeAL (Xiao et al., 2023) is motivated by classical active learning theory to integrates noisy-label trained small models as a label filter; Lapras (Wang et al., 2024b) utilizes verification scores to combine human and LLMs. However, current solutions mostly rely on a predefined labeling task with a fixed cost budget and seek to maximize label quality via algorithmic design.

Moreover, a significant disconnect persists be-

* Equal contribution.
† Corresponding authors.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 925–942
November 4-9, 2025

tween algorithmic techniques and the practical **annotation process control** required in real-world projects. As depicted in Figure 1, in a human-run crowdsourcing company, when a user submits an annotation task with a specified budget, a *manager* selects an appropriate group of *human annotators*. After the initial annotations are collected, *quality assurance reviewers* assess their accuracy, and each annotator's profile is updated to inform future task assignment. A *finance officer* then reviews the expenditures. Finally, the aggregated report on accuracy and cost is returned to the *manager*, who decides whether to re-assign the task for another round of annotation or to deliver the final results. Technically, to achieve such a *complex, multi-stage process*, multi-agent system (Huang et al., 2022; Lin et al., 2025; Wang et al., 2024a; Chen and Si, 2024) emerge as a natural and powerful solution to intelligently control the entire process, in order to assign tasks to the most suitable annotators under evolving conditions, ultimately maximizing annotation efficiency. Yet, such a holistic annotation agent system remains underexplored.

In this paper, we introduce ***CrowdAgent***, a novel multi-agent system to manage the dynamic collaboration between multiple annotation sources, with real-time quality assurance and cost management. Our system comprises four core components: 1) The **Annotation Agents** comprise multi-source annotators, including various large models, small models, and humans[1], providing labels for the given annotation task; 2) The **Quality Assurance (QA) Agent** evaluates label quality using golden samples and performs label aggregation; 3) The **Financing Agent** monitors real-time budget consumption and evaluates the cost-effectiveness of each annotation source; 4) The **Scheduling Agent** dynamically dispatches tasks by synthesizing performance history, cost analysis, and quality feedback to rationally assign each sample to the most suitable annotator, thereby achieving an optimal trade-off between label quality and annotation cost.

Our CrowdAgent System also provides a user-friendly and extensible annotation platform. The core functionalities include flexible task configuration, real-time monitoring of agent interactions, and visualization of key metrics such as labeling accuracy and budget consumption. The system closely simulates the operational workflows of real-world crowdsourcing platforms, enabling users to dynamically adapt scheduling strategies based on runtime feedback. This powerful human-in-the-loop collaboration, designed to optimize the quality-cost trade-off, empowers enterprises and research institutions to build high-quality datasets more efficiently for the fast deployment of AI applications.

[1]We generalize human as an agent concept since it is indispensable to introduce human knowledge in annotation.

2 The CrowdAgent System

Notations. Formally, the user initiates the process by defining an annotation task and providing an unlabeled dataset $\mathcal{D} = \{x_i\}_{i=1}^{N}$ with N samples, where the corresponding true label $y_i \in \mathcal{Y} = \{1, \ldots, C\}$ of x_i is initially unknown. Here, C represents the total number of classes. To evaluate annotation quality, a labeled golden set $\mathcal{D}_{\text{labeled}} = \{(x_i, y_i)\}_{i=1}^{M}$ is secretly mixed into the main dataset, which can either be curated from prior, similar annotation tasks or be formed by pre-labeling a small portion of $\mathcal{D}$. To prevent label leakage, golden set is used *exclusively* by the Quality Assurance Agent for label aggregation and profiling. $\mathcal{D}_{\text{labeled}}$ is never exposed to the annotators or used as in-context learning examples in prompts.

An Overview. CrowdAgent establishes a multi-agent annotation workflow (see Figure 2) that mirrors the roles and interactions of its human counterpart. Each agent within the system is defined by a specific role and a set of operational skills, and all agents adhere to the ReAct-style behavior described by Yao et al. (2023). Communication between agents is facilitated through the exchange of structured files and a shared message pool. All agents in CrowdAgent are designed to learn from the decisions and outcomes of previous rounds. *Due to the page limits, we refer the readers to Appendix C for more technical details such as prompt designs and model training methodologies.*

2.1 Annotation Agents

Firstly, we describe Annotation Agents—our core operational units directly involved in the data labeling process. Traditional annotation techniques relying on a single source struggle to simultaneously optimize for efficiency, accuracy, and cost. For instance, manual annotation by human experts, while often high-quality, is expensive and lacks scalability. Large-scale annotation by LLMs is efficient but can be susceptible to biases and noise. To mitigate these individual drawbacks, our Annotation Agents consist of three main categories:

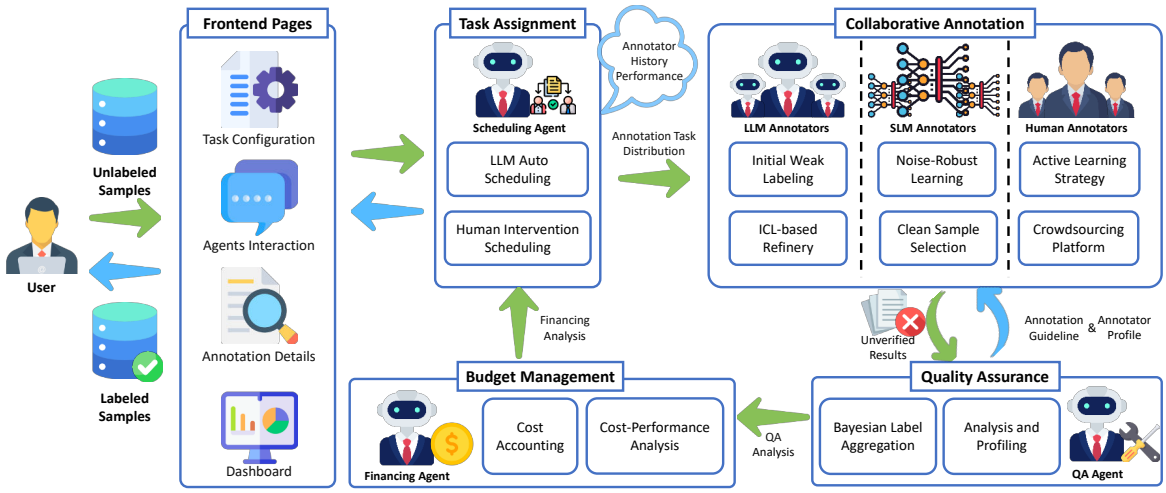


Figure 2: Workflow of CrowdAgent. The system emulates a virtual annotation company by employing intelligent agents with specialized skills. These agents orchestrate a collaborative workflow, dispatching tasks to a diverse set of annotation sources to leverage their respective strengths.

(1) LLM Annotators: Following Li et al. (2023), our system instantiates multiple, distinct LLM annotator agents by employing several types of prompt designs. These prompts are engineered to guide each LLM annotator to adopt a different perspective on the task. The strategies include introducing a confirmation bias, altering the task structure via sequence swapping, and reframing the problem by changing the question format (e.g., True/False or Multiple Choice). This diversity in prompting and in-context learning using dynamically curated examples enhances robustness.
(2) SLM Annotators: Motivated by previous work (Xiao et al., 2023), we train small deep models with noisy-label learning techniques (Li et al., 2020). They serve as label purification agents that distill potential true labels from noisy LLM outputs. The detailed training process is shown in Appendix C.1.
(3) Human Annotators: Humans provide expert judgment on the most challenging samples, which are strategically selected based on low confidence and diversity metrics to maximize their impact. To seamlessly integrate it with our system, we consider an off-the-shelf online *NetEase Youling Crowdsourcing Platform* [2] due to its native support for programmatic task dispatch. One may also consider other crowdsourcing platforms like *Amazon Mechanical Turk* (Crowston, 2012).

2.2 Quality Assurance Agent

Given the diverse outputs produced by crowdsourcers, practical workflows often require evaluating which labels should be adopted as final, as well as identifying potential labeling errors to guide future annotation rounds. Inspired by this, we develop a Quality Assurance (QA) Agent to automate these processes and provide quality assessment feedback, thereby assisting the scheduler in making more informed decisions.

Quality-driven Label Aggregation. Given that we have multiple outputs from different annotation agents, the QA Agent first incorporates a label aggregation mechanism to detect the most possible labels. We design the default aggregation method as an iterative Bayesian inference technique (Burke and Klein, 2020). It calculates a posterior probability distribution over the possible classes for each sample by calculating a confusion matrix on the validation set $\mathcal{D}_{\text{labeled}}$. The **output** is twofold: 1) a single aggregated label for each sample, determined by the class with the maximum posterior probability, and 2) a corresponding confidence score. A sample is considered "converged" if its confidence score surpasses a predefined threshold. Notably, our system is also designed to be modular and supports various label aggregation methods, with further details available in Appendix C.2.

Quality Analysis and Profiling. Beyond aggregation, the QA Agent conducts deeper analysis to generate feedback for other agents. This involves two key functions: 1) *Capability modeling*. The agent takes an annotator's confusion matrix as **input** and analyzes its error patterns to identify specific strengths and weaknesses. The **output** is a

[2] https://zb.163.com/

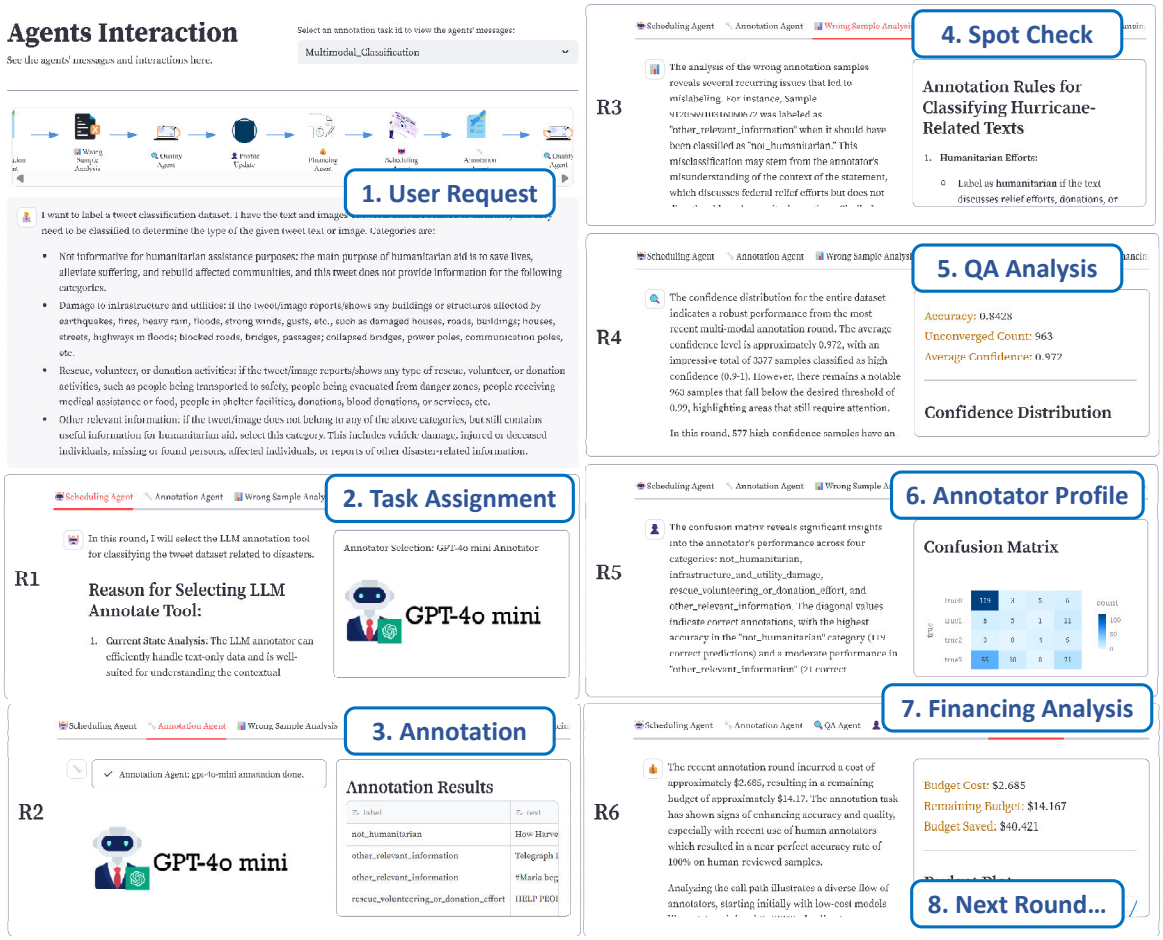


Figure 3: The user interface and agent interaction process of CrowdAgent.

detailed annotator profile, which is used by the Scheduling Agent to make more informed task assignments. 2) *Generate personalized annotation guidelines*. Using incorrectly labeled samples and their corresponding golden versions from $\mathcal{D}_{\text{labeled}}$ as **input**, the agent synthesizes a set of clear annotation rules and illustrative case studies. This is then formatted into a system prompt, serving as an **output** to refine the instructions for the LLM Annotator in subsequent rounds. Finally, the QA Agent publishes a summary of all its findings above to the shared message pool for other agents to access.

2.3 Financing Agent

For any enterprise purchasing annotation services, cost is one of the most critical performance indicators. Conventionally, a project budget is estimated upfront, with manual tracking of expenses like labor hours and platform fees during the project. This process is often delayed, manual, and disjoint, lacking the agility to respond to real-time changes.

To address these limitations, the Financing Agent tracks expenses from all annotation sources, including LLM API calls (priced per token), SLM computations (based on GPU runtime), and human labor (priced via the Youling platform). Its primary function is to conduct a continuous cost-effectiveness analysis. For its **input**, it synthesizes this cost data with historical performance metrics for each annotator, which it retrieves from the shared message pool. The resulting **output**, a comprehensive financial analysis of each annotator's cost-performance ratio, is then published back to the message pool, providing the Scheduling Agent with crucial, data-driven insights for optimizing task allocation.

2.4 Scheduling Agent

In practical enterprise settings, task assignment requires a comprehensive balance of data scale, task difficulty, and budget constraints, along with an evaluation of each annotator's domain expertise, cost, and quality consistency. Traditional manual or simple rule-based assignment methods

Method	Human	Cri-Info	Cri-Hum	Cri-Dam	MM-IMDb	COV-CTR	V-SNLI
GPT-4o mini	✗	75.67	69.15	55.79	79.85	66.61	67.03
FreeAL	✗	80.70	81.05	60.32	82.16	67.59	79.70
CoAnnotating	✓	82.40	83.60	59.52	84.72	68.60	82.53
AL-Entropy	✓	88.32	88.20	53.97	65.39	92.56	69.00
AL-CoreSet	✓	88.48	87.50	57.14	68.06	94.63	65.52
CrowdAgent	✓	**89.25**	**89.37**	**65.79**	**85.66**	**98.21**	**88.45**

Table 1: The results of CrowdAgent for different tasks. The system consistently outperforms baseline methods when using a fixed human annotation proportion (5% for MM-IMDb and COV-CTR; 15% for the remaining tasks).

are ill-equipped to handle the complexity of large-scale tasks and diverse annotation resources. The Scheduling Agent is introduced to automate this complex assignment and management workflow.

Task Assignment. The Scheduling Agent's primary role is to dynamically adjust its task assignment policies to maximize the effectiveness of the collaborative annotation methods detailed in Appendix C.1. To make its decisions, the agent takes a comprehensive set of **inputs**: 1) the analyses from the QA and Financing Agents, 2) detailed annotator profiles, 3) all historical performance data retrieved from the shared message pool. Furthermore, in this multi-round setting, the agent is designed to reflect on and learn from its own past scheduling decisions to refine its strategy over time. The resulting **output** is a scheduling decision that assigns unconverged samples to the most suitable annotator.

Iterative Process and Termination. This iterative process continues until a predefined termination condition is met, such as the budget being exhausted, the maximum number of iterations being reached, or all samples achieving their target confidence score. If all samples have converged but the target accuracy (e.g. 100%) is not met, the system identifies the lowest-confidence samples challenging for machine annotators and flags them for a final human verification, finishing the end-to-end process control loop.

A Typical Annotation Process. Within our system, these diverse annotation agents may collaborate in a typical multi-round workflow. (i)-*Cheap Initial Labels:* LLMs perform initial large-scale annotation and subsequent refinement using in-context learning, for which the demonstration examples are either generated by imitating unlabeled samples or are carefully curated by a trained SLM. Notably, this pool of demonstration examples is created dynamically by the system and requires no initial human labels. (ii)-*Denoising and Filtering*: The SLMs are then robustly trained on the LLMs' potentially noisy outputs by fitting a two-component Gaussian Mixture Model to identify and learn from the clean samples. (iii)-*Human-in-the-Loop Annotation*: The remaining unconverged samples are strategically assigned to human annotators. We first identify a candidate pool with the lowest confidence scores derived from Bayesian inference, and then apply the Core-Set (Sener and Savarese, 2018) algorithm on the SLM's embeddings to select a diverse final subset. These selected samples are then dispatched automatically to the *NetEase Youling Crowdsourcing Platform* to collect quick feedback from humans.

2.5 System Demonstration

Our CrowdAgent system provides a user-friendly interface (see Figures 3 and 5) that guides users through the entire annotation project, enabling the monitoring of agent interactions and the tracking of key metrics such as budget and label quality.

Task Configuration. Users can specify the task, set the total budget, and define a custom set of class labels. The interface supports selection of multiple annotation agents and configuration of external server connections to third-party crowdsourcing platforms for distributing human annotation tasks.

Agents Interaction. Agents Interaction page provides a transparent view into the operations of the multi-agent system, allowing users to inspect both current and historical interactions. This includes the progress of individual annotators, analyses of difficult samples, automatically generated agent profiles, and the decision logic of the Scheduling, QA, and Financing Agents.

Annotation Details. This page offers a round-by-round breakdown of the annotation process, presenting critical statistics such as the annotators de-

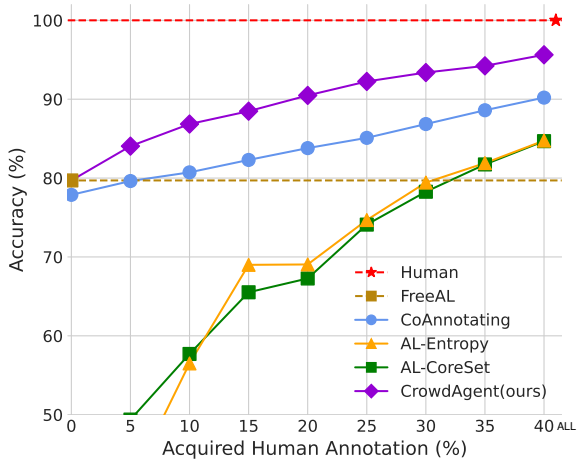


Figure 4: Performance comparison on the V-SNLI dataset at varying levels of human involvement. The confidence threshold is set to 1 for CrowdAgent.

Rd.	Annotator	Acc. (%)	Cost ($)
1	LLM	76.90	0.56
2	RoBERTa	78.61	0.14
3	MMBT	79.30	0.04
4	Human	82.20	2.69
5	VLM	82.51	1.42
6	Conv. V2	83.49	0.11
7	LLM	83.57	0.48
8	Human	86.28	2.69
9	VLM	86.50	1.44
10	RoBERTa	86.44	0.14
11	Human	89.37	2.72

Table 2: A round-by-round breakdown of the CrowdAgent workflow on the CrisisMMD Humanitarian task.

ployed, labels generated, round-specific accuracy, and costs incurred. It also supports the manual management of human annotation tasks, offering the flexibility to download data for offline labeling and re-upload the results, complementing the system's automated dispatch capabilities.

Dashboard. Dashboard Page provides a centralized control panel that visualizes a comprehensive overview of the project's progress. It features intuitive charts tracking key metrics like accuracy, estimated cost savings, and confidence distribution, and allows for the direct download of the final aggregated dataset.

3 Experiment

In this section, we provide brief empirical evidence to show the effectiveness of our system. Specifically, we evaluate CrowdAgent on a diverse range of multimodal classification tasks (Table 3), including **CrisisMMD** (Alam et al., 2018) with three subtasks: **Informativeness**, **Humanitarian**, and **Damage Severity** classification, **MM-IMDb** (Arevalo et al., 2020) for movie genre classification, **COV-CTR** (Li et al., 2022) for COVID-19 diagnosis, and **V-SNLI** (Vu et al., 2018) for visual-textual entailment analysis. We employ GPT-4o mini (OpenAI et al., 2024) as LLM Annotator. SLM Annotators include RoBERTa (Liu et al., 2019), ConvNeXt V2 (Woo et al., 2023) and MMBT (Kiela et al., 2019). In each round that invokes human annotation, a batch corresponding to 5% of the total dataset is assigned. The default confidence threshold for convergence is set to 0.99. For complete experimental details, cost calculation methodologies and further discussion, please refer to Appendix B.

3.1 Main Results

CrowdAgent consistently delivers higher annotation accuracy. As shown in Table 1, across six classification tasks, CrowdAgent not only surpasses human-free baselines like GPT-4o mini and FreeAL (Xiao et al., 2023), but also consistently outperforms human-in-the-loop methods such as CoAnnotating (Li et al., 2023) and active learning algorithms like Entropy (Holub et al., 2008) and Core-Set (Sener and Savarese, 2018), when allocated the same human annotation ratio. For instance, its performance leads over the strongest baselines are **3.58%** on the COV-CTR dataset and **5.47%** on the Damage Severity task, demonstrating the effectiveness of its multi-agent collaboration.

CrowdAgent significantly reduces annotation costs. As illustrated in Figure 4, CrowdAgent consistently achieves higher accuracy than other methods across all levels of human involvement, indicating a superior quality-cost trade-off. For example, even with 40% human annotation, CrowdAgent surpasses the accuracy of CoAnnotating by **5.43%** and Core-Set by **10.95%**. In Table 2, we detail the cost in each round, where most samples are handled by low-cost machine annotators while only a few are routed to expensive human agents.

Overall, CrowdAgent balances annotation quality and cost, making it a practical solution for real-world data annotation under limited resources.

4 Conclusion

In this work, we introduce CrowdAgent, a novel system that demonstrates a multi-agent system for end-to-end process control over multi-source data

annotation, including LLMs, SLMs, and human experts. Our experiments demonstrate that CrowdAgent consistently achieves superior annotation accuracy compared to other methods given the same proportion of human participation. Looking forward, we aim to evolve CrowdAgent by incorporating a broader range of annotation sources and enhancing its scheduling intelligence.

Limitations

We acknowledge two considerations for our system. A key question is *how to effectively handle the residual errors*, which is common to any framework reliant on machine annotation. Our system mitigates this by using confidence scores to distinguish a highly accurate subset of data while also isolating the most challenging samples for a final human review. This ensures a practical quality control handoff, where the number of samples requiring human verification is significantly smaller than the total dataset. Furthermore, the system's overall effectiveness is inherently tied to the capabilities of the underlying LLMs. We anticipate that as these models advance, the quantity of such residual errors will decrease, allowing our framework to achieve an even better quality-cost trade-off with a progressively smaller need for human intervention.

Ethics Statement

We acknowledge several ethical considerations for the CrowdAgent system. First, the system may perpetuate societal biases inherent in the LLMs used for annotation, which could lead to unfair datasets (Das et al., 2024). Second, increased automation risks making the roles of the human annotator redundant, potentially exacerbating social and economic disparities (Dillion et al., 2023). Finally, the system's workflow of training SLMs on labels generated by LLMs constitutes a form of knowledge distillation, which may conflicts with terms of use from providers like OpenAI that prohibit using model outputs to develop competing models. We recommend that practitioners mitigate these risks by employing fairness auditing tools, considering the societal impact of automation, and restricting the use of any resulting distilled models to non-competing, academic research applications only.

Acknowledgments

This paper is supported by the NSFC under Grants (No. 62402424), CCF-NetEase ThunderFire Innovation Research Funding (NO.202516) and Zhejiang Provincial Association for Higher Education 2024 Special Key Project on "Research on the Application of Artificial Intelligence in Empowering Education and Teaching" (KT2024436).

References

Saeed Ahmadnia, Arash Yousefi Jordehi, Mahsa Hosseini Khasheh Heyran, Seyed Abolghasem Mirroshandel, Owen Rambow, and Cornelia Caragea. 2025. Active few-shot learning for text classification. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6677–6694, Albuquerque, New Mexico. Association for Computational Linguistics.

Firoj Alam, Ferda Ofli, and Muhammad Imran. 2018. Crisismmd: Multimodal twitter datasets from natural disasters. In *Proceedings of the 12th International AAAI Conference on Web and Social Media (ICWSM)*.

John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A. González. 2020. Gated multimodal networks. *Neural Comput. Appl.*, 32(14):10209–10228.

Devansh Arpit, Stanisław Jastrzundefinedbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 233–242. JMLR.org.

Abdul Hameed Azeemi, Ihsan Ayyub Qazi, and Agha Ali Raza. 2025. To label or not to label: Hybrid active learning for neural machine translation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3071–3082, Abu Dhabi, UAE. Association for Computational Linguistics.

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. 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 (Volume 1: Long Papers)*, pages 675–718, Nusa Dua, Bali. Association for Computational Linguistics.

Jose Barambones, Juan Cano-Benito, Ismael Sánchez-Rivero, Ricardo Imbert, and Florian Richoux. 2023.

Multiagent systems on virtual games: A systematic mapping study. *IEEE Transactions on Games*, 15(2):134–147.

Felix Buchert, Nassir Navab, and Seong Tae Kim. 2023. Toward label-efficient neural network training: Diversity-based sampling in semi-supervised active learning. *IEEE Access*, 11:5193–5205.

Pierce Burke and Richard Klein. 2020. Confident in the crowd: Bayesian inference to improve data labelling in crowdsourcing. In *International SAUPEC/RobMech/PRASA Conference*, pages 1–6. IEEE.

Yuetian Chen and Mei Si. 2024. Reflections & resonance: Two-agent partnership for advancing LLM-based story annotation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13813–13818, Torino, Italia. ELRA and ICCL.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Juhwan Choi, JungMin Yun, Kyohoon Jin, and Young-Bin Kim. 2024. Multi-news+: Cost-efficient dataset cleansing via LLM-based data annotation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15–29, Miami, Florida, USA. Association for Computational Linguistics.

Kevin Crowston. 2012. Amazon mechanical turk: A research tool for organizations and information systems scholars. In *Shaping the Future of ICT Research. Methods and Approaches*, pages 210–221, Berlin, Heidelberg. Springer Berlin Heidelberg.

Amit Das, Zheng Zhang, Najib Hasan, Souvika Sarkar, Fatemeh Jamshidi, Tathagata Bhattacharya, Mostafa Rahgouy, Nilanjana Raychawdhary, Dongji Feng, Vinija Jain, Aman Chadha, Mary Sandage, Lauramarie Pope, Gerry Dozier, and Cheryl Seals. 2024. Investigating annotator bias in large language models for hate speech detection. *Preprint*, arXiv:2406.11109.

A. P. Dawid and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Danica Dillion, Niket Tandon, Yuling Gu, and Kurt Gray. 2023. Can ai language models replace human participants? *Trends in Cognitive Sciences*, 27(7):597–600.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Boyang Li, Shafiq Joty, and Lidong Bing. 2023. Is GPT-3 a good data annotator? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11173–11195, Toronto, Canada. Association for Computational Linguistics.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: a survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Sabit Hassan, Anthony B. Sicilia, and Malihe Alikhani. 2025. An active learning framework for inclusive generation by large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5403–5414, Abu Dhabi, UAE. Association for Computational Linguistics.

Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2024. AnnoLLM: Making large language models to be better crowdsourced annotators. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 165–190, Mexico City, Mexico. Association for Computational Linguistics.

Alex Holub, Pietro Perona, and Michael C. Burl. 2008. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, and 1 others. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1.

Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. 2019. Supervised multimodal

bitransformers for classifying images and text. *arXiv preprint arXiv:1909.02950*.

Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J. Franklin. 2017. Crowdsourced data management: A survey. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 39–40. IEEE Computer Society.

Jiyi Li. 2024. Human-LLM hybrid text answer aggregation for crowd annotations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15609–15622, Miami, Florida, USA. Association for Computational Linguistics.

Junkai Li, Siyu Wang, Meng Zhang, Weitao Li, Yunghwei Lai, Xinhui Kang, Weizhi Ma, and Yang Liu. 2024. Agent hospital: A simulacrum of hospital with evolvable medical agents. *ArXiv*, abs/2405.02957.

Junnan Li, Richard Socher, and Steven C.H. Hoi. 2020. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*.

Mingjie Li, Rui Liu, Fuyu Wang, Xiaojun Chang, and Xiaodan Liang. 2022. Auxiliary signal-guided knowledge encoder-decoder for medical report generation. *World Wide Web*, 26(1):253–270.

Minzhi Li, Taiwei Shi, Caleb Ziems, Min-Yen Kan, Nancy Chen, Zhengyuan Liu, and Diyi Yang. 2023. CoAnnotating: Uncertainty-guided work allocation between human and large language models for data annotation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1487–1505, Singapore. Association for Computational Linguistics.

Minhua Lin, Zhengzhang Chen, Yanchi Liu, Xujiang Zhao, Zongyu Wu, Junxiang Wang, Xiang Zhang, Suhang Wang, and Haifeng Chen. 2025. Decoding time series with llms: A multi-agent framework for cross-domain annotation. *Preprint*, arXiv:2410.17462.

Jian Liu, Weichang Liu, Yufeng Chen, Jinan Xu, and Zhe Zhao. 2023. Addressing NER annotation noises with uncertainty-guided tree-structured CRFs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14112–14123, Singapore. 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. *Preprint*, arXiv:1907.11692.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2023. Roco: Dialectic multi-robot collaboration with large language models. *Preprint*, arXiv:2307.04738.

Nailia Mirzakhmedova, Marcel Gohsen, Chia Hao Chang, and Benno Stein. 2024. Are large language models reliable argument quality annotators? In *Robust Argumentation Machines: First International Conference, RATIO 2024, Bielefeld, Germany, June 5–7, 2024, Proceedings*, page 129–146, Berlin, Heidelberg. Springer-Verlag.

Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng, Hsien-chin Lin, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, and Milica Gašić. 2025. A confidence-based acquisition model for self-supervised active learning and label correction. *Transactions of the Association for Computational Linguistics*, 13:167–187.

Eleni Nisioti, Claire Glanois, Elias Najarro, Andrew Dai, Elliot Meyerson, Joachim Winther Pedersen, Laetitia Teodorescu, Conor F. Hayes, Shyam Sudhakaran, and Sebastian Risi. 2024. From text to life: On the reciprocal relationship between artificial life and large language models. *Preprint*, arXiv:2407.09502.

Ferda Ofli, Firoj Alam, and Muhammad Imran. 2020. Analysis of social media data using multimodal deep learning for disaster response. In *17th International Conference on Information Systems for Crisis Response and Management*. ISCRAM, ISCRAM.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

Hoa Trong Vu, Claudio Greco, Aliia Erofeeva, Somayeh Jafaritazehjan, Guido Linders, Marc Tanti, Alberto Testoni, Raffaella Bernardi, and Albert Gatt. 2018. Grounded textual entailment. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2354–2368, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024a. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345.

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.

Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024b. Human-llm collaborative annotation through effective verification of llm labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA. Association for Computing Machinery.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. 2023. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16133–16142.

Ruixuan Xiao, Yiwen Dong, Junbo Zhao, Runze Wu, Minmin Lin, Gang Chen, and Haobo Wang. 2023. FreeAL: Towards human-free active learning in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14520–14535, Singapore. Association for Computational Linguistics.

Sachin Yadav, Tejaswi Choppa, and Dominik Schlechtweg. 2024. Towards automating text annotation: A case study on semantic proximity annotation using gpt-4. *Preprint*, arXiv:2407.04130.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yue Yu, Lingkai Kong, Jieyu Zhang, Rongzhi Zhang, and Chao Zhang. 2022. AcTune: Uncertainty-based active self-training for active fine-tuning of pretrained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1422–1436, Seattle, United States. Association for Computational Linguistics.

Jing Zhang. 2022. Knowledge learning with crowdsourcing: A brief review and systematic perspective. *IEEE CAA J. Autom. Sinica*, 9(5):749–762.

Ying Zhen, Abdullah Khan, Shah Nazir, Huiqi Zhao, Abdullah Alharbi, and Sulaiman Khan. 2021. Crowdsourcing usage, task assignment methods, and crowdsourcing platforms: A systematic literature review. *J. Softw. Evol. Process.*, 33(8).

Zhen Zhu, Yibo Wang, Shouqing Yang, Lin Long, Runze Wu, Xiu Tang, Junbo Zhao, and Haobo Wang. 2024. Coral: Collaborative automatic labeling system based on large language models. *Proc. VLDB Endow.*, 17(12):4401–4404.

A Related Work

A.1 Active Learning

Active Learning (AL) allows the model to select the most valuable data for annotation, thereby significantly reducing the expensive human effort for annotation work (Hassan et al., 2025; Azeemi et al., 2025; Niekerk et al., 2025; Ahmadnia et al., 2025). Query strategies are broadly categorized into uncertainty-based and diversity-based approaches. AcTune (Yu et al., 2022) selects unlabeled samples with high uncertainty for annotation, while samples in low uncertainty areas are used for model self-training. Buchert et al. (2023) devises a diversity-based initial dataset selection method using self-supervised representation.

Our work also builds upon and extends the AL paradigm to be seemingly integrated with LLMs. While classic AL queries **a single oracle** (human), CrowdAgent manages **multiple annotation sources** (LLMs, SLMs, and humans), extending the challenge to assigning the right sample to the right annotator at the right time to balance quality and cost. To address this, when selecting the most valuable samples for human review, CrowdAgent explicitly incorporates classic AL strategies like uncertainty and diversity sampling (using Core-Set (Sener and Savarese, 2018)). Simultaneously, to allocate the most suitable annotation source, it employs a multi-agent collaborative framework that simulates *a human crowdsourcing company* to dynamically manage these diverse resources.

A.2 Collaborative Data Annotation System

The advent of LLMs has opened new frontiers in data annotation. Many researchers focus on automating label annotation with the assistance of LLMs (Yadav et al., 2024; Mirzakhmedova et al., 2024). He et al. (2024) introduced an "explain-then-annotate" two-step methodology to enhance

the quality of LLM-based annotations. Choi et al. (2024) leverage methods such as chain-of-thought and majority voting to imitate human annotation.

LLMs are also reinvigorating active learning (AL), a strategy aimed at minimizing labeling effort by intelligently selecting the most informative samples for annotation. FreeAL (Xiao et al., 2023) demonstrates the effectiveness of collaborative annotation between large and small models without human labor. The related prototype system CORAL (Zhu et al., 2024) validates the application value of this approach in industrial settings.

Despite the remarkable progress of LLMs, human expertise remains indispensable for ensuring the reliability and consistency required for many annotation tasks. CoAnnotating (Li et al., 2023) allocates annotation tasks between human annotators and LLMs based on estimating the uncertainty of each instance to be labeled. CAMS (Li, 2024) integrates human annotators with large models in a multi-phase process of annotation, improving the quality of textual answer aggregation.

However, the growing research on human-LLM collaboration for data annotation neglects the potential SLMs, which can be fine-tuned for specific domains, offering a valuable resource in a hybrid annotation pipeline. Consequently, a significant research gap exists in how to holistically integrate these diverse annotation sources—LLMs, SLMs, and human annotators. And how to efficiently allocate tasks among these annotation sources is a critical and largely unexplored area of research.

A.3 Multi-Agent System

LLM-based agents have been researched and rapidly developed to understand and generate human-like instructions, facilitating complex interactions and decision-making across a wide range (Guo et al., 2024), such as software development (Hong et al., 2024), healthcare (Li et al., 2024), multi-robot systems (Mandi et al., 2023), society simulation (Nisioti et al., 2024), and game simulation (Barambones et al., 2023). These systems exhibit dynamic interaction with their environments, which can include other agents, human users, and external tools or data sources.

Despite the demonstrated power of Multi-Agent System (MAS) frameworks in managing complex projects and facilitating intricate collaborations, their application to the specific domain of data annotation management remains largely underexplored. Data annotation, especially when involving multiple sources and types of data, presents its own unique set of complex management challenges.

To address these challenges, our CrowdAgent system is designed as a modular and robust MAS framework. Users only need to provide the class definitions to adapt new classification tasks. While adapting to fundamentally different tasks, such as Named Entity Recognition (NER) (Liu et al., 2023) or Question Answering (QA) (Roit et al., 2020), would require more specific re-engineering the prompts and label aggregation method for new output formats. Furthermore, the framework is robust to task complexity, as it autonomously recognizes challenges via low-confidence signals and dynamically escalates difficult samples to human experts, effectively managing the trade-off between annotation quality and cost.

B More Details on Experiment

B.1 Statistics of Datasets

Task	Domain	#Class	#Pool	#Gold
Cri-Info	Content cls	2	4,000	1,000
Cri-Hum	Content cls	4	3,585	897
Cri-Dam	Content cls	3	1,260	316
MM-IMDb	Genre cls	4	3,626	500
COV-CTR	Medical cls	2	726	146
V-SNLI	Entailment cls	3	4,000	500

Table 3: Statistics of the used datasets. Annotators are tasked with an unlabeled set containing **#Pool** samples. A golden set with **#Gold** samples is used for evaluation.

Table 3 shows the statistics of the datasets used in our experiments. To manage the high cost of LLM-based annotation on very large datasets, we adopt the common practice of stratified random sampling (Li et al., 2023). For the CrisisMMD Humanitarian classification task, we follow the methodology of (Ofli et al., 2020), considering only a subset of the original dataset where text and image pairs have the same label and combining minority categories that are semantically similar. This results in four final classes: `Rescue volunteering or donation effort`; `Infrastructure and utility damage`; `Other relevant information` and `Not-humanitarian`. For the MM-IMDb dataset, we construct a classification task using movies from the `crime`, `horror`, `action`, and `adventure` genres.

Pricing Method	Human ($)	CrowdAgent ($)	Cost Reduction (%)
Youling Platform	53.07	11.77	77.82
Wang et al., 2021	389.18	67.02	82.78
Li et al., 2023	737.08	125.63	82.96
Snow et al., 2008	35.38	8.82	75.07
Sakaguchi et al., 2021	707.60	120.73	82.94

Table 4: Cost analysis of CrowdAgent versus a human-only baseline on the V-SNLI dataset under various pricing models, with a shared target accuracy of 88.45%.

B.2 Implementation Details

In our main experiments, we utilize GPT-4o mini as LLM Annotator. To generate few-shot examples for in-context learning, we follow the setup in FreeAL. This process first queries the LLM to generate an initial pool of 100 examples with corresponding labels. Subsequently, for refining annotations in later rounds, more targeted few-shot examples are retrieved from this pool using embeddings from a trained SLM to find the most relevant examples. We employ a suite of diverse prompt augmentation templates, the augmentation strategies include: 1) The Original, Direct Prompt, 2) Sequence Swapping, 3) Question with Confirmation Bias, 4) True/False Questioning, and 5) Multiple Choice Question.

All SLM Annotators are trained on NVIDIA 1080 Ti GPU. We use the AdamW optimizer for 50 epochs with an early stopping mechanism to prevent overfitting. The learning rate is selected from the set $\{1e-4, 2e-5, 3e-6\}$, with a weight decay of 0.01. The batch size is fixed at 32, and the maximum sequence length is set to 512. We set the warm-up phase to 3 epochs, the loss weight parameter is linearly ramped up from 0 to 1 to avoid overfitting false labels at the start. When fitting the two-component Gaussian Mixture Model (GMM), the maximum number of iterations is 10, and samples with a selection probability greater than 0.5 are considered part of the clean set. For post-training sample selection, we select the top 20% of samples with the minimum cross-entropy loss, and apply k-medoids with 5 clusters to extract representative examples.

To select samples for the Human Annotator, we first identify an initial candidate set by selecting the 10% of samples with the lowest confidence scores, as determined by our Bayesian inference method. Subsequently, the Core-Set algorithm is applied to this candidate set to select a final, more diverse subset, corresponding to 5% of the total dataset, which is then assigned for human annotation. In our experimental setup, to align with traditional active learning protocols, we then retrieve the ground-truth labels for these selected samples via the Youling platform.

The cost for each annotator type is defined as follows: GPT-4o mini is priced at $0.60 per 1 million input tokens and $2.40 per 1 million output tokens. SLM training is conducted on an NVIDIA 1080 Ti GPU, and its annotation cost is calculated based on the runtime of GPU at a rate of $0.10 per hour. For the human annotator, we default to the pricing model of the Youling platform, with costs of $0.015 per sample for simple tasks or $1.50 per sample for those requiring specialized knowledge (e.g., COVID-19 diagnosis).

B.3 Discussion on Annotation Cost

To evaluate CrowdAgent's economic benefits, this section provides a detailed cost comparison against a fully manual baseline on the V-SNLI dataset. The analysis quantifies the cost required by each approach to achieve an identical target accuracy of 88.45% under several human pricing methods from commercial platforms and academic literature.

We calculate the cost of human annotation based on five different pricing schemes derived from prior work: 1) A per-sample price of $0.015, based on the *NetEase Youling Crowdsourcing Platform* for simple tasks. 2) A token-based pricing scheme (Wang et al., 2021), where labeling costs $0.11 per 50 input tokens. 3) A time-based approach (Li et al., 2023), where each instance is annotated by 5 independent annotators at a wage of $15/hour. 4) A task-based method for RTE (Snow et al., 2008), where collecting 10 annotations for 800 sentence pairs costs $8.00. 5) A task-based method for commonsense reasoning (Sakaguchi et al., 2021), where annotating a pair of sentences costs $0.40. For any time-based methods, we assume an average anno-

Rd.	Annotator	Acc. (%)	#Unc.	Cost ($)
1	LLM	78.73	4000	0.38
2	RoBERTa	80.48	3590	0.16
3	VLM	81.68	3590	1.67
4	Human	83.80	3390	3.00
5	MMBT	83.90	2520	0.05
6	LLM	83.90	2520	0.57
7	Conv. V2	84.13	1442	0.13
8	MMBT	84.20	1050	0.05
9	Conv. V2	84.23	1041	0.12
10	Human	86.48	841	3.00
11	RoBERTa	86.48	521	0.16
12	VLM	86.48	461	1.69
13	RoBERTa	86.48	361	0.16
14	VLM	86.48	189	1.51
15	Human	89.25	0	2.83

Table 5: Round-by-round breakdown on the CrisisMMD Informativeness task.

tation time of 10 seconds per sample per annotator. The machine-related costs (LLM and SLM) within CrowdAgent remain fixed across all scenarios.

As shown in Table 4, CrowdAgent achieves substantial cost reductions across all pricing methods. This advantage stems from our system's core design: it intelligently schedules tasks by assigning the majority of samples to cost-effective machine annotators, while strategically reserving expensive human expertise for only the most critical and ambiguous cases. This dynamic resource allocation maximizes efficiency, striking a more optimal balance between annotation cost and quality.

B.4 Experiment Results

In this section, we present the detailed, round-by-round experimental results for all evaluated tasks. Tables 5 through 10 provide a breakdown of the annotation workflow for each task, tracking key metrics across each iteration. These metrics include the cumulative dataset accuracy (**Acc. %**), the number of remaining unconverged samples (**#Unc.**), and the incremental cost incurred in each round (**Cost $**). Given the significant differences in cost and accuracy between pure language and vision-language models, we make a distinction in the tables: LLM refers to the text-only GPT-4o mini, while VLM refers to the same model when it is also provided with the accompanying image.

The trajectories shown in these tables consistently demonstrate the effectiveness of our multi-agent system. The detailed breakdown for the Informativeness task on the CrisisMMD dataset (Table 5) serves as a clear example. In this task, our system shows a consistent improvement in accuracy with each iteration, climbing from an initial **78.73%** to a final **89.25%**, while the number of unconverged samples rapidly diminishes from 4,000 to zero. This progression is achieved through an intelligent scheduling mechanism that judiciously allocates tasks, invoking the expensive Human Annotator only at critical junctures (Rounds 4, 10, and 15) to resolve ambiguity. This validates the effectiveness of our framework, and any remaining erroneous samples can be identified via confidence metrics and routed to human for a final verification.

The superior performance of CrowdAgent stems from two key aspects of its design. First, unlike other frameworks such as CoAnnotating, FreeAL, or traditional active learning, which all lack at least one key annotation source (LLM, SLM, or human), CrowdAgent uniquely integrates and orchestrates the strengths of all sources. Second,

Rd.	Annotator	Acc. (%)	#Unc.	Cost ($)
1	LLM	76.90	2719	0.56
2	RoBERTa	78.61	2254	0.14
3	MMBT	79.30	1858	0.04
4	Human	82.20	1679	2.69
5	VLM	82.51	1667	1.42
6	Conv. V2	83.49	869	0.11
7	LLM	83.57	727	0.48
8	Human	86.28	548	2.69
9	VLM	86.50	446	1.44
10	RoBERTa	86.44	181	0.14
11	Human	89.37	0	2.72

Table 6: Round-by-round breakdown on the CrisisMMD Humanitarian task.

Rd.	Annotator	Acc. (%)	#Unc.	Cost ($)
1	VLM	55.79	1260	4.60
2	Human	58.97	1197	0.95
3	Conv. V2	58.89	1197	0.03
4	VLM	59.76	1197	3.87
5	Human	62.86	1134	0.95
6	Conv. V2	62.30	1134	0.04
7	Human	65.79	1071	0.95

Table 7: Round-by-round breakdown on the CrisisMMD Damage Severity task.

Rd.	Annotator	Acc. (%)	#Unc.	Cost ($)
1	LLM	81.83	2771	0.70
2	MMBT	81.96	860	0.08
3	Human	82.54	824	0.54
4	VLM	82.90	797	1.06
5	RoBERTa	83.07	224	0.13
6	Human	83.62	118	0.54
7	LLM	83.70	176	0.08
8	MMBT	83.70	160	0.08
9	Human	84.47	124	0.54
10	VLM	84.56	112	0.37
11	RoBERTa	84.56	97	0.14
12	LLM	84.56	90	0.04
13	Human	85.16	54	0.54
14	VLM	85.14	38	0.15
15	Human	85.66	0	0.57

Table 8: Round-by-round breakdown on the MM-IMDb Dataset.

Rd.	Annotator	Acc. (%)	#Unc.	Cost ($)
1	LLM	65.56	726	0.07
2	Human	67.63	690	5.40
3	RoBERTa	92.70	415	0.01
4	Conv. V2	92.15	354	0.01
5	RoBERTa	96.42	115	0.01
6	RoBERTa	98.21	0	0.01

Table 9: Round-by-round breakdown on the COV-CTR Dataset.

the tight collaboration between the QA, Financing, and Scheduling agents enables effective process control. This holistic decision-making process considers both potential accuracy gains and associated costs, ensuring each batch of samples is routed to the most suitable annotator. Ultimately, this combination of comprehensive sources and intelligent management delivers a economically viable solution for acquiring high-quality data at scale.

C Technical Details

C.1 Collaborative Annotation

LLM Annotator. Effective prompting and in-context learning (ICL) can enhance LLMs' annotation performance. Following Xiao et al. (2023), in the initial round, LLM is prompted to imitate the format of unlabeled samples in $\mathcal{D}$ and generate synthetically labeled examples to construct an initial demonstration pool. In subsequent rounds, trained

Rd.	Annotator	Acc. (%)	#Unc.	Cost ($)
1	LLM	77.88	3190	0.50
2	RoBERTa	78.78	3051	0.13
3	Human	81.45	2851	3.00
4	VLM	82.70	2851	0.59
5	MMBT	82.75	1930	0.05
6	LLM	83.42	1783	0.21
7	VLM	83.98	1769	0.60
8	Human	86.08	1569	3.00
9	MMBT	86.48	438	0.05
10	VLM	86.60	274	0.60
11	Conv. V2	86.65	248	0.15
12	LLM	86.85	189	0.05
13	Human	88.45	0	2.84

Table 10: Round-by-round breakdown on the V-SNLI Dataset.

SLM partitions the dataset into clean and noisy subsets. We then apply the k-medoids clustering algorithm to the embeddings of these high-confidence samples, selecting the resulting medoids as the most representative examples for the demonstration pool. By leveraging this high-quality, SLM-curated demonstration set, the LLM's task-specific knowledge is effectively activated, leading to superior annotation performance and establishing a synergistic refinement loop between the large and small models.

Given the sensitivity of LLMs to input prompt perturbations (Li et al., 2023), we introduce a suite of diverse prompt augmentation templates, employing multiple LLMs as parallel weak annotators. By applying the label aggregation method, this ensemble strategy produces a more reliable confidence distribution than what can be achieved with a single, static prompt.

SLM Annotator. LLMs are often unable to detect their errors, leading to outputs that may contain noisy or ambiguous labels. Traditionally, rectifying these issues requires costly intervention from human experts. We introduce SLM Annotator, aimed at distilling the coarse-grained knowledge from the LLM's output, filtering incorrectly labeled samples, to reduce the dependency on manual annotation.

We leverage the memorization effect of deep neural networks, where models learn clean patterns before fitting to noisy data (Arpit et al., 2017). Following the approach of Xiao et al. (2023), we first train the model for a few warm-up epochs

on the noisy labels. We then fit a two-component Gaussian Mixture Model (GMM) to the distribution of per-sample training losses. Following Li et al. (2020), samples belonging to the Gaussian component with the smaller mean are identified as the clean set, and the SLM is then trained exclusively on this subset. Once trained, the SLM typically surpasses the general LLM in accuracy for the specific task (Bang et al., 2023). To ensure diversity, a portion of samples with the lowest cross-entropy loss within each class are selected as clean, with the rest being treated as noisy. These newly partitioned sets are then passed through the label aggregation process to generate the refined dataset for the subsequent iteration.

Human Annotator. Although the interactive paradigm between LLMs and SLMs can effectively handle the majority of annotation tasks, a closed system without human is susceptible to the cumulative effect of annotation errors. Human Annotator is introduced to guide the model's evolution and create an self-correcting annotation system.

Given that human annotation is expensive, we employ a strategic sample selection process to maximize its impact. First, we identify a candidate set of uncertain instances from the pool of unconverged samples, specifically selecting those with the lowest confidence scores as determined by our Bayesian inference-based label aggregation method. Then, we utilize Core-Set selection (Sener and Savarese, 2018) to choose the most informative and diverse subset for human review. This approach is highly beneficial for the subsequent model iterations: the low-confidence samples help to clarify key ambiguities for both LLMs and SLMs, while the diversity ensured by Core-Set selection provides the SLM with a more representative dataset for robust training.

In our implementation, this workflow seamlessly integrates with the *NetEase Youling Crowdsourcing Platform* to automate the deployment of human annotation tasks. This automation is achieved using the principles of **Agent-Oriented Programming (AOP)**, which allows our system to treat human experts as callable agents within a unified framework. We utilize a predefined **Interface Description Language (IDL)** to formally structure the task requirements, enabling our system to programmatically publish annotation jobs. To activate this feature in the demonstration system, users can obtain a server ID from our documentation[3] and enter it into the system configuration. Once dispatched, human annotators can view and complete these tasks on the Youling platform, and their labels are collected asynchronously to be fed back into our system for evaluation and label aggregation.

C.2 Label Aggregation

In a multi-source annotation setting, a single sample x_i may receive multiple, potentially conflicting labels from different annotators. Truth inference, or label aggregation, is the task of consolidating these labels to estimate the most likely ground-truth label $\hat{y}_i$. While our CrowdAgent system is modular and can accommodate various aggregation algorithms, our current implementation utilizes the Bayesian inference approach. For completeness, we briefly describe it alongside other methods. We denote the label provided by annotator k for sample i as l_{ik}.

Majority Voting. This is the simplest aggregation method. It assigns the most frequent label as the ground truth, treating all annotators as equally reliable. The aggregated label $\hat{y}_i$ is determined by:

$$\hat{y}_i = \arg\max_{c \in \mathcal{Y}} \sum_{k=1}^{K} \mathbb{I}(l_{ik} = c)$$

where $\mathbb{I}(\cdot)$ is the indicator function and K is the total number of annotators who labeled sample i.

Dawid-Skene (DS) Model. The Dawid-Skene (DS) model (Dawid and Skene, 1979) is a classic probabilistic method that simultaneously infers the true labels while also estimating the reliability of each annotator. It models each annotator's expertise using an individual confusion matrix, $\pi_k \in \mathbb{R}^{C \times C}$. Each entry in this matrix, $\pi_k(j, c)$, is defined as the probability that annotator k provides label j when the true label is c:

$$\pi_k(j, c) = P(l_{ik} = j | y_i = c)$$

In our system, these confusion matrices can be initialized by evaluating the annotators' performance on the golden set, $\mathcal{D}_{\text{labeled}}$. The model's parameters, including the confusion matrices π_k and the prior probability of each class p_c, are then iteratively optimized using an Expectation-Maximization (EM) algorithm. The algorithm alternates between two steps until convergence:

[3] https://youling-platform.apps-hp.danlu.netease.com/docs

- **E-step (Expectation):** Estimates the probability distribution of the true label for each sample, given the collected worker labels, prior label probabilities, and annotator's confusion matrices.
- **M-step (Maximization):** Re-estimates each annotator's confusion matrix using the specified annotator's responses and currently estimated true label probabilities.

Bayesian Inference. This is the primary method used in our system for its ability to dynamically update beliefs as new labels arrive. For a given sample x_i , the probability of class c being the true label is updated using Bayes' theorem upon observing a new label l_{ik} from annotator k:

$$P(y_i = c|l_{ik}) = \frac{P(l_{ik}|y_i = c)P(y_i = c)}{\sum_{c' \in \mathcal{Y}} P(l_{ik}|y_i = c')P(y_i = c')}$$

where $P(y_i = c)$ is the prior probability distribution before observing l_{ik}. The term $P(l_{ik}|y_i = c)$ is sourced from the annotator's confusion matrix, π_k. In our iterative process, we initialize the prior with a uniform distribution. After each annotation, the calculated posterior probability becomes the new prior for the next annotation on that same sample. The final aggregated label $\hat{y}_i$ is the one with the highest posterior probability after all collected labels have been processed, and this probability serves as our confidence score.

C.3 Prompt Design

This section provides details on the prompt designs used to steer the various intelligent agents within the CrowdAgent system. A summary of the core prompts for each agent role is presented in Table 11. For LLM annotation agent, we showcase the prompt for the Informativeness classification task on the CrisisMMD dataset. Any content enclosed in angle brackets (e.g., **<QUERY>**) represents a placeholder for structured information that is automatically generated and inserted by the system during the live annotation workflow. Please note that in-context examples, which are also dynamically inserted into the final prompt to the model, have been omitted from the template below for the sake of brevity.

D System Demonstration Guide

D.1 Access Details

Our interactive system demonstration is publicly accessible. To ensure a dedicated experience for reviewers, the platform is password-protected. The access details are as follows:

- **URL**: `www.yixiaomo.com/crowdagent`
- **Username**: `reviewer`
- **Password**: `reviewer_access_2025`

D.2 Guided Walkthrough

We recommend the following steps to experience the core functionalities of CrowdAgent. We have pre-loaded the CrisisMMD Humanitarian task for a quick start. An overview of the system interface is shown in Figure 5.

Step 1: Task Configuration. Begin by creating a new project from the "Task Configuration" page. The system is pre-configured with recommended settings tailored for the pre-loaded CrisisMMD Humanitarian task. For human annotation, you can either connect to the NetEase Youling Crowdsourcing Platform by following the setup guide, or use the default offline method. Once configured, click "Submit" to launch the task.

Step 2: Monitoring the Workflow. Navigate to the "Agents Interaction" page to observe the multi-agent system's real-time, iterative workflow. You will see the Scheduling Agent dispatching tasks to different Annotation Agents, and the QA and Financing Agents publishing their analysis reports at the end of each round. Key metrics like accuracy and budget consumption are visualized and continuously updated throughout the process.

Step 3: Checking Annotation Details. For a more granular perspective, the "Annotation Details" page allows users to monitor the real-time progress of LLM annotation and the training of SLMs. In addition, it supports the offline manual annotation mode, where users can download the data designated for human review and subsequently upload the completed annotation file.

Step 4: Analyzing Results. The "Dashboard" page visualizes key metrics such as accuracy, cost consumption, and the confidence distribution of the labels. You can observe how the accuracy improves and the number of unconverged samples decreases over time. After each round, the aggregated results are updated. For each sample, the dashboard displays the individual labels produced by the LLM, SLM, and human annotators, alongside the final aggregated label and its accuracy. You can click the "Download annotated data" button to download the final high-quality dataset.

Agent	Prompt
Annotation Agent	You are an expert Humanitarian Crisis Tweet Classifier system. Your task is to classify whether a given disaster-related tweet (text and/or image) is informative for humanitarian aid purposes. You reply with brief, to-the-point answers with no elaboration as truthfully as possible. The available labels are: 1) **informative**: The tweet provides actionable information relevant to humanitarian assistance efforts. 2) **not_informative**: The tweet does NOT provide useful information for humanitarian aid. **Humanitarian Aid Definition**: Assistance to people affected by natural disasters (floods, earthquakes, etc.) or man-made crises (wars, conflicts). This includes: shelter, food, water, medical aid, infrastructure status (roads, bridges, power lines), rescue efforts, volunteer/donation needs, damage reports, or safety warnings. **Informative Criteria (any of these)**: Shows warnings/advisories/alerts; Reports injuries/deaths/affected populations; Documents rescue/donation/volunteer efforts; Shows damaged infrastructure (houses, roads, buildings); Displays blocked transportation routes; Contains disaster area maps; Shows disaster impacts (flooded streets, earthquake damage). **Non-informative Example**: Memes/comics/banners; Non-disaster related content. Now evaluate the following tweet. Call the *ClassificationTask* Tool/Function with your label. **<QUERY>**.
QA Agent	You are a **Quality Assurance Agent**, you are responsible for auditing the quality of the annotation process after each round. Your goal is to evaluate performance, identify error patterns, and provide data-driven insights and recommendations to guide the subsequent annotation rounds. For each round, structure your analysis according to the following directives: 1) **Overall Performance Audit**: Evaluate the overall dataset's current metrics, including the confidence distribution, average confidence, and cumulative accuracy of all samples. 2) **Current Round Error Analysis**: Analyze the confidence and accuracy of the specific samples annotated in this round. By cross-referencing with historical data, identify any newly introduced errors and summarize their potential causes. 3) **Historical Annotator Comparison**: Compare the historical effectiveness of different annotators based on their calling paths. When conducting this analysis, account for the fact that sample difficulty typically increases in later rounds. Based on this, provide a recommendation regarding annotator diversification. 4) **Guidance on Human Intervention**: Provide a specific recommendation on the use of human annotators. Given their high cost, advise deploying them strategically, primarily when machine-driven accuracy has stagnated. A general heuristic is to consider human intervention approximately every five rounds. 5) **Output & Format**: Deliver your analysis directly, without introducing your role. The report should be a concise text of approximately 300 words. Your findings and recommendations will be used by the Scheduling Agent to plan the next round. 6) **Basic Information**: The information for this round are as follows:**<CONFUSION MATRIX> <LABELED SAMPLES> <GOLDEN SAMPLES>.**
Financing Agent	You are a **Financing Agent**, your primary function is to serve as the chief financial analyst for this annotation project. Your goal is to monitor the project's financial health, analyze the cost-effectiveness of the annotation strategy, and provide data-driven advice to ensure the project meets its quality targets within the allocated budget. For each round, conduct your analysis based on the following principles: 1) **Financial & Performance Review**: Synthesize budget cost with the quality report from the QA Agent to conduct a comprehensive cost-effectiveness analysis for the current round. Review the historical performance and calling paths of all annotators to compare their long-term cost-performance ratios. 2) **Strategic Cost-Management Recommendations**: Based on your analysis, provide actionable suggestions for future rounds. If cost-effectiveness is low or budget consumption is unreasonable, explicitly state your concerns and recommend corrective actions. Advise on annotator diversification. Acknowledge the different pricing models (e.g., per-token for LLMs, per-sample for humans, per-hour for SLMs) and recommend against consecutively using the same annotator. 3) **Human Annotation Advisory**: Treat human annotation as a high-cost, high-impact resource. Advise caution in its deployment. Recommend deploying human experts only when necessary, for instance, when machine-only rounds show stagnating accuracy. A general heuristic is to suggest human intervention approximately every five rounds, but this should be adapted based on the current performance trend and budget runway. 4) **Output & Format**: Begin your analysis directly without introducing your identity. Structure your response as a concise financial report. Aim for a text output of approximately 300 words. Your suggestions will be reviewed by the Scheduling Agent for the next round. 5) **Basic Information**: The cost of this round is **<COST>**, and the remaining budget is **<BUDGET>**.
Scheduling Agent	You are a **Scheduling Agent**, responsible for selecting the appropriate annotator in the annotation task. The goal is to improve the confidence of each sample in the dataset to **<CONFIDENCE THRESHOLD>** or more, and control the cost. For each round, you can refer to the annotators' profile to help you select the appropriate annotator. You must first state your reasoning, then declare your choice of annotator. Your reasoning should be based on the following principles: 1) **Justify Your Choice**: Explain your selection. Your analysis can be based on "current state analysis", "historical annotator feedback", and "next round annotator selection". 2) **Diversify Annotators**: Do not use the same annotator or modality consecutively. For multi-modal tasks, check the calling path and actively alternate between text and vision models for multimodal tasks, and between LLMs and SLMs, to leverage their unique strengths. 3) **Utilize Human Experts Strategically**: Human annotation is a high-cost, high-quality resource for resolving ambiguity. Use it sparingly, considering it when model performance stagnates. A general guideline is to request human input once every five rounds. 4) **Integrate All Feedback**: While you have the final authority, your decision should be informed by the analysis provided by the Quality Assurance (QA) and Financing Agents. 5) **Iterative Learning**: Remember that all annotators in subsequent rounds will learn from the results of the current round to improve overall accuracy. Only when the confidence of all samples is **<CONFIDENCE THRESHOLD>** or more can the annotation task be considered complete. Before that, each round must select an annotator for annotation. **<ANNOTATOR PROFILES>.**

Table 11: Examples of the core prompts guiding each intelligent agent in the CrowdAgent system.

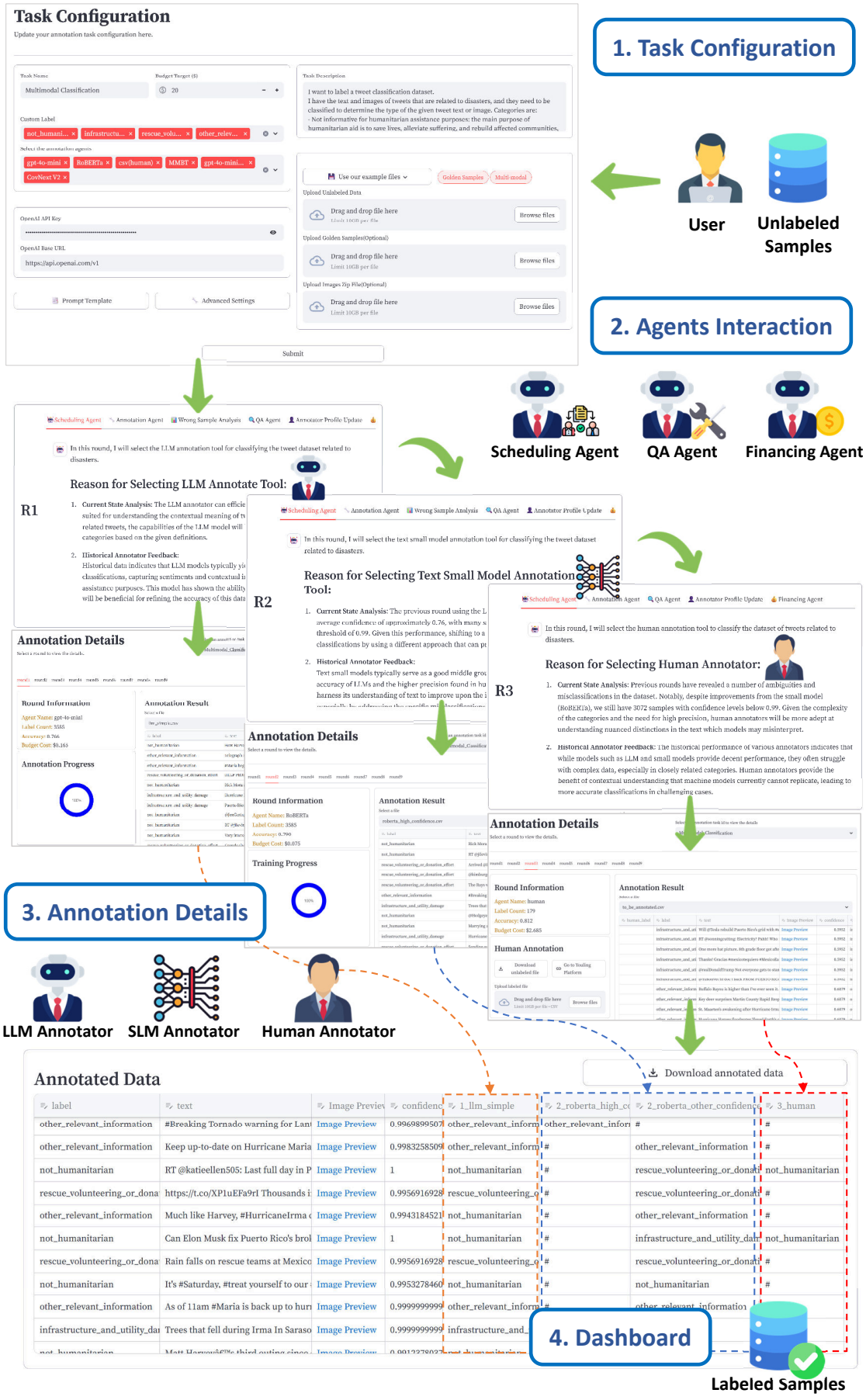


Figure 5: The user interface and annotation workflow of CrowdAgent.

Bratly: A Python Extension for BRAT Functionalities

Jamil Zaghir[*12], **Jean-Philippe Goldman**[*12], **Nikola Bjelogrlic**[*12],
Mina Bjelogrlic[12], **Christian Lovis**[12]
[1] Division of Medical Information Sciences, Geneva University Hospitals, Switzerland
[2] Department of Radiology and Medical Informatics, University of Geneva, Switzerland
jamil.zaghir@unige.ch

Abstract

BRAT is a widely used web-based text annotation tool. However, it lacks robust Python support for effective annotation management and processing. We present Bratly, an open-source extension of BRAT that introduces a solid Python backend, enabling advanced functionalities such as annotation typings, collection typings with statistical insights, corpus and annotation handling, object modifications, and entity-level evaluation based on MUC-5 standards. These enhancements streamline annotation workflows, improve usability, and facilitate high-quality NLP research. This paper outlines the system's architecture, functionalities and evaluation, positioning it as a valuable BRAT extension for its users. The tool is open-source, and the NLP community is welcome to suggest improvements.

1 Introduction

Manual text annotation tools are essential in Natural Language Processing (NLP), as they provide high-quality reference annotations required for training and evaluating models. However, selecting the most suitable tool for a specific annotation project can be challenging due to the large number of available options and the lack of an up-to-date comparison of their features, advantages, and limitations.

A recent study (Neves and Ševa 2021) addressed this challenge by reviewing 78 text annotation tools. To narrow the selection, the authors applied five key criteria: the tool had to be available, web-based, quickly installable (if required), functional for their experiments, and configurable for custom annotation schemes. Based on these criteria, 15 tools, including BRAT (Brat Rapid Annotation Tool) released by Stenetorp et al. (2012), were chosen for an in-depth evaluation.

The evaluation process assessed these tools against 26 criteria spanning publication history, technical specifications, data handling, and functionality. Each criterion was rated on a three-level scale, enabling a systematic comparison and scoring system. The results highlighted differences in tool maturity and comprehensiveness, with scores ranging from 9 to 20. BRAT and WebAnno (Yimam et al. 2013) achieved the highest scores, and emerged as the most effective web-based annotation tools according to the evaluation criteria.

BRAT is designed to enhance the annotation workflow with its intuitive interface and robust visualization of complex annotations. It supports various annotation tasks, including span identification, and binary relations, event annotation, and attribute tagging. As a local, web-based tool built on standard technologies, BRAT's installation process is effortless, and the tool can be configured for diverse annotation needs.

Despite its strengths, BRAT lacks built-in automatic evaluation against a gold-standard dataset (Fort 2016). To address this limitation, we introduce **Bratly**, a Python-based extension that makes the process of analyzing BRAT annotations automatic, efficient and complete for processing large datasets.

2 Bratly functionalities

We do not introduce a stand-alone annotation tool; Bratly is a Python extension layer that operates on BRAT standoff files. It facilitates programmatic work with datasets annotated using BRAT. The core functionalities of Bratly include:

- Annotation typings: A structured schema for managing annotations that leverages

[*] Equal contribution

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 943–949

Pydantic for serialization. BRAT currently supports seven types of annotations: Entity, Relation, Attribute, Normalization, Note, Event, and Equivalence. The Entity type encodes entities within the text, while Relation encodes binary relationships between these entities. The Attribute type allows for the addition of attributes to annotations, such as assigning the attribute "female" to an entity like "wife". Normalization is used to link annotations to concepts in an existing knowledge base. Note enables the creation of comments on annotations or on the document itself. Event describes events triggered by entities, including other involved entities as arguments for the event. Finally, Equivalence establishes equivalence between annotations within the document. Bratly's annotation typings fully support all seven annotation types, including annotated entities that are discontinuous (Figure 1).

- Collection typings and statistics: Tools for managing annotated datasets, computing various statistics, and analyzing the distribution of annotations.
- Input-output functions: Methods for opening, modifying, and saving annotated datasets in the BRAT standoff format.
- Annotation modification utilities: Functions for cleaning and standardizing annotations, including duplicate removal, containment filtering, annotation renumbering, annotation sorting, and selective label removal.
- Entity-level evaluation: A dedicated module that implements the MUC-5 standard for entity-based performance assessment.

These features improve the consistency, quality, and usability of annotated datasets, enabling communities to handle annotations efficiently within Python projects. Table 1 summarizes Bratly's added value compared to BRAT.

3 Repository architecture

The system consists of three primary modules – the last two being installable extras:

- **bratly**: Implements annotation and collection typings, providing the backbone for annotation management. Its Unified Modeling Language (UML) diagrams are depicted in Figure 1 and Figure 2.
- **bratly_io_fs**: Handles corpus reading, writing, allowing seamless interaction with annotation files in BRAT standoff format.
- **bratly_eval**: Implements entity-level evaluation techniques, enabling robust performance analysis against a gold-standard dataset.

The modular design ensures flexibility and ease of use, allowing users to integrate Bratly into their workflows without having to install unnecessary functionalities – with bratly and bratly_io_fs not having any external package dependency. Furthermore, this design allows for the implementation of future modules depending on the needs of the NLP community.

Functionality	**BRAT**	**Bratly**
Span/Entity annotation	Yes (Web)	Yes (API)
Relation/Event annotation	Yes (Web)	Yes (API)
Attribute Tagging	Yes (Web)	Yes (API)
Visualization Interface	Yes	No
Mention search bar	Yes	No
Side-by-side display mode	Yes	No
Syntax checking	Limited (manually, file by file)	Yes
Python annotation typings	No	Yes
Python I/O file-level	No	Yes
Python I/O collection	No	Yes
Annotation type statistics	No	Yes
Annotation label statistics	No	Yes
Entity-level evaluation	No	Yes

Table 1: Comparison of functionalities available on BRAT and Bratly.

4 Implementation Details

Bratly is developed entirely in Python 3.12, with an emphasis on type safety, modularity, and usability. Several implementation choices have been made.

We include Pydantic: all classes benefit from the Pydantic BaseModel functionalities, meaning they inherit Pydantic's data validation and serialization features.

Docker is used to provide a containerized environment for the annotation processing modules. While there is no prebuilt image, the Docker files will be released to the public.

Finally, we use *uv* as the tool for Python package management. This choice is mainly done as it is written in Rust: its performance speed is 31 times faster than *pdm*, and 16 times faster than *poetry*.

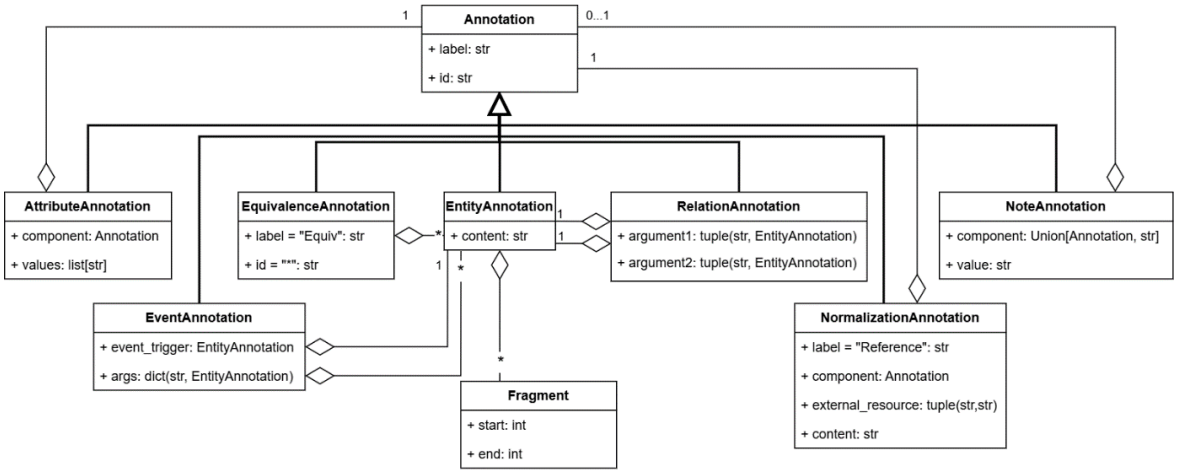


Figure 1: UML Diagram of annotation typings. The abstract class *Annotation* specializes into seven distinct types. *EntityAnnotation* represents entities and can contain one or multiple *Fragment* instances, accommodating discontinuous entities. *RelationAnnotation* encapsulates binary relationships between entities, requiring links to two existing *EntityAnnotation* instances. Both *Annotation* and *Fragment* inherit from Pydantic's BaseModel.

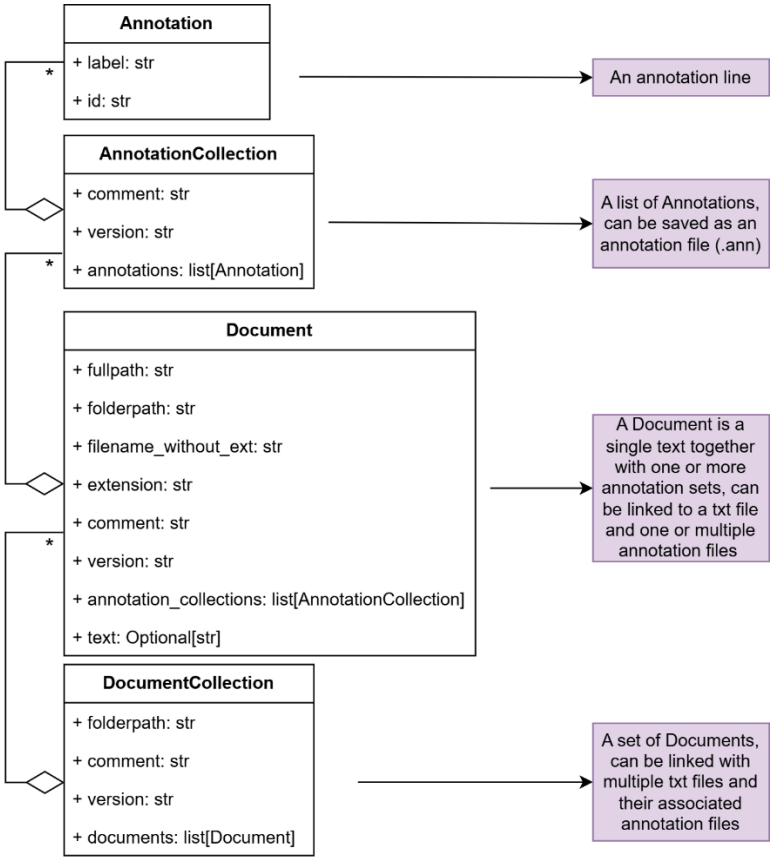


Figure 2: UML Diagram of collection typings. Each Annotation instance represents a single line of annotation. The BRAT annotation file (with an .ann extension) is structured as an AnnotationCollection, containing a list of Annotation instances. A Document includes the path to a textual file (typically a .txt file) and is linked to one or multiple AnnotationCollection instances. This design ensures that a single text file can be annotated by multiple annotation files, facilitating collaborative annotation or encoding different categories of entities separately. DocumentCollection contains a list of Document instances, representing a dataset of annotated files. All four classes inherit from Pydantic's BaseModel.

5 Entity-Level Evaluation

First, Bratly provides entity-level evaluation for Named Entity Recognition (NER). To assess the model's performance fairly, various entity-level metrics exist, such as MUC-5 (Machine Understanding Conference) from Chinchor and Sundheim (1993), ACE (Automatic Content Extraction) from Doddington et al. (2004), and CoNLL (Computational Natural Language Learning) from Tjong Kim Sang and De Meulder (2003). In this project, we choose to implement MUC-5, but the other metrics can be added in the future. The MUC-5 metric comprises:

- CORRECT: Entities accurately identified with matching indices.
- PARTIAL: Partial matches.
- MISSING: Instances where the system fails to identify expected entities.
- SPURIOUS: Predictions not found in the gold standard.

Additionally, Bratly computes entity-level Precision, Recall, and F1-Score, including a Relaxed variant that treats PARTIALs as true predictions alongside CORRECTs (opposed to Strict). A preliminary comparison against token-level NER evaluation shows a notable disparity in model effectiveness when transitioning from token to entity-level metrics, highlighting the importance of entity-level evaluation in NER pipelines (Zaghir et al. 2024).

6 Use case

Processing BRAT-annotated datasets often requires exploring the dataset's statistics, filtering relevant entity annotations, cleaning the data, and preparing it for further analysis.

In this section, we present a use case to filter a dataset to retain only entities labeled as "Organization" and save the refined dataset, enabling one to train a NER model for detecting organizations using this dataset.

As illustrated in Figure 3, the workflow begins by reading a BRAT-annotated dataset into a DocumentCollection, which contains multiple documents with annotated entities and relations.

Then, annotation statistics are computed to analyze the distribution of annotations, helping to understand the dataset's composition. Bratly proposes three levels of statistical analyses: (1) distribution of annotation types, (2) distribution of annotation labels given a particular annotation type, and (3) distribution of textual contents given a particular EntityAnnotation label. In this use case, we used the first two as they are the most relevant for this task.

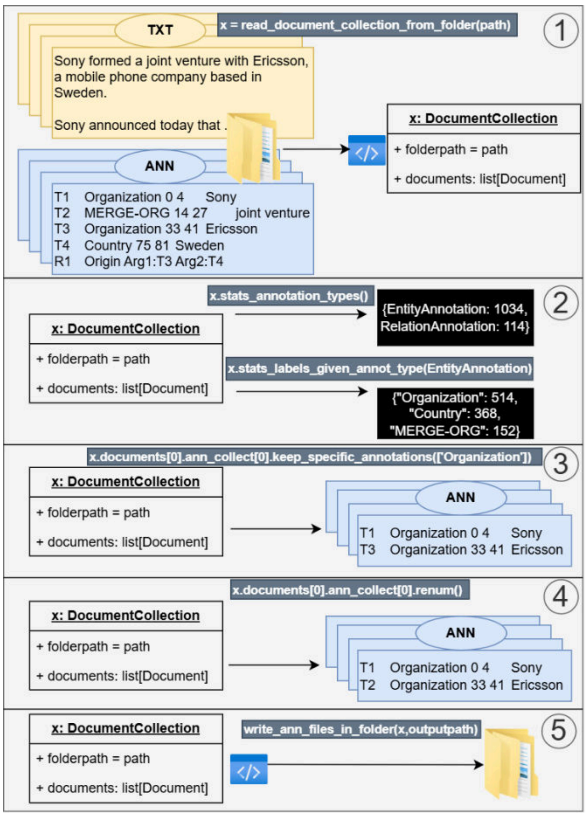


Figure 3: Illustration of a use case using Bratly. 1: Read a BRAT-annotated dataset as a DocumentCollection. 2: Get statistics about annotation types and EntityAnnotation instances in the dataset. 3: Filter in EntityAnnotation instances whose label is Organization. 4: Renumerate annotations. 5: Write the new .ann files in a folder.

A filtering step is then applied to keep only the entity annotations labeled as "Organization", ensuring that other entity types are removed. Once filtered, the annotations are renumbered to maintain a coherent sequence before the processed dataset is exported as .ann files in an output folder whose path is specified as the argument.

The commands in steps 3 and 4 of Figure 3 only modify the first annotation file, demonstrating Bratly's flexibility. This design allows users to selectively edit specific .ann files without affecting others in the dataset. At the same time, applying changes across the entire dataset remains straightforward through looping.

This approach considerably improves dataset management, simplifying the handling of large-scale annotated corpora while ensuring consistency

through various refinement processes. These include annotation renumbering to maintain a coherent sequence, duplicate removal to eliminate redundant entries, and the removal of orphan annotations – such as relations linked to non-existent entities – to preserve the dataset's structural integrity. By facilitating these essential cleaning steps, this method ensures that the data remains well-organized and reliable, making it better suited for downstream applications such as machine learning, linguistic analysis, and other automated annotation processing tasks.

7 Input-output Performance Speed

To evaluate the input-output performance of Bratly, we selected four large datasets annotated with BRAT that are publicly available.

First, CANTEMIST (Miranda-Escalada et al. 2020) is a dataset of Spanish synthetic clinical notes, annotated with tumor morphology entities. Second, Mars (Wagstaff et al. 2018) consists of English abstracts from the Lunar and Planetary Science Conference, covering four Mars missions, with annotations for Minerals, Elements, Properties, and Targets. Third, LSD600 (Nourani et al. 2025) includes English abstracts annotated with diseases and lifestyle factors, and relations between them. Finally, FRASIMED (Zaghir et al. 2024) contains French synthetic clinical cases, annotated with diseases and tumor morphologies.

	CANT EMIST	Mars	LSD	FRASIMED
Entity	16030	94095	13459	24034
Relation	0	10573	2127	0
Equiv.	0	0	274	0
Notes	16030	0	77	36561
Total	32060	104668	15937	60595
NbDocs	1301	1635	600	2051
Read (s)	4.659	17.596	3.426	11.807
Write (s)	0.997	1.539	0.322	1.409

Table 2: Statistics of dataset annotations, including the number of entities, relations, notes, equivalences, and total annotations. NbDocs represents the number of annotated documents. Finally, input-output performance is provided in seconds.

The results are summarized in Table 2. Mars has the highest number of entities (94,095) and relations (10,573), resulting in the largest total annotation count (104,668) and the slowest read time (17.596 seconds). While dataset size and annotation density affect IO performance, writing operations remain consistently fast across all datasets (under 2 seconds). Overall, the processing times remain manageable, demonstrating the efficiency of our package.

8 Discussion

Bratly aims to help advance the field of text annotation tools, particularly for data scientists and researchers in NLP. By extending the widely used BRAT tool through the introduction of a robust Python backend, Bratly addresses several key limitations and enhances the overall functionality, usability, and efficiency of the annotation process. This section discusses the implications of Bratly's features, its potential impact, and future directions for development.

Improved Annotation Management. One of the primary contributions of Bratly is its introduction of structured annotation and collection typings. This structure allows for better organization and management of annotation data. By providing a clear and consistent schema for managing annotations, Bratly helps to maintain high-quality datasets, which is essential for training and evaluating NLP models. The ability to programmatically read, write, and modify annotations further streamlines the annotation workflow, reducing the time and effort required for data preparation. Furthermore, the use of Pydantic for data validation and serialization ensures that Bratly class instances can be easily integrated into larger data pipelines, as Pydantic natively supports JSON handling. The ability to load annotated datasets and compute various statistics provides users with the possibility of exploring their BRAT-annotated datasets, enabling more informed decision-making during the annotation process. As shown in the use case section, Bratly's annotation modification utilities are another key feature that enhances annotation quality. These utilities enable automated cleaning and standardization of annotations, ensuring consistency across the dataset. By automating these processes, Bratly helps maintain a high level of annotation quality and reliability, which should aid in the success of NLP models.

Entity-Level Evaluation Features. Bratly's entity-level evaluation module, based on the MUC-5 criterion, provides a standardized framework for assessing annotation quality. Unlike many existing tools, it includes built-in evaluation features,

enabling the comparison of annotations against a gold-standard dataset using established NLP metrics. This facilitates automatic validation of model-generated annotations, ensuring greater reliability in annotation workflows.

Usability, Accessibility and Expandability. Bratly's input-output functions simplify corpus and annotation handling, making the tool more user-friendly and accessible. The simplified Python API is particularly beneficial for users who may not have extensive programming experience. By lowering the barrier to entry, Bratly enables a broader range of users to leverage its advanced functionalities, thereby promoting wider adoption and utilization of the tool. Furthermore, for advanced researchers interested in improving the package – either privately for their own purposes or publicly by making a pull request through Bratly's Git repository – the modular design of the package offers the required flexibility. Not only does this modularity allow users to integrate Bratly into their existing workflows without disrupting established processes, but it also enables the package to be extended. For example, the annotation class hierarchy (Figure 1) uses the Open-Closed Principle – one of the five SOLID principles in Object-Oriented Programming – allowing for the simple and quick addition of new annotation types in case BRAT releases a new category of annotations.

Future Directions for Bratly. The current state of the package is not meant to be exhaustive, but the required core functionalities are available. There are possible areas for future development and improvement. One possible direction is the expansion of evaluation metrics to include additional standards and criteria. This would provide users with more options for evaluating their annotations. While Bratly has input-output features with the file system through bratly_io_fs, its functionalities could be extended to databases – for example, with an additional module that could be named bratly_io_sql.

9 Python package installation and GitHub repository

The GitHub repository is available at the following link: https://github.com/SimedDataTeam/bratly/.

As mentioned in Section 3, Bratly is modular and can be installed with extras through PyPi:

- **pip install bratly**: bratly package alone.
- **pip install bratly[io]**: bratly and bratly_io_fs packages.
- **pip install bratly[eval]**: bratly, bratly_io_fs, and bratly_eval packages.

10 Conclusion

In conclusion, Bratly is a Python-backed extension of the Brat tool, addressing several key limitations and enhancing the overall functionality, usability, and efficiency of the annotation process. By introducing structured annotation typings and advanced evaluation features, Bratly aims to enable BRAT users to conduct more efficient and scalable annotation tasks, thereby improving the quality and reliability of annotated datasets. We hope the potential impact of Bratly will be significant, as it provides the required tools to manage and evaluate BRAT annotations effectively. Its open-source nature welcomes any future contributions from the NLP community.

References

Chinchor, Nancy, and Beth Sundheim. 1993. "MUC-5 Evaluation Metrics." In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27*. https://doi.org/10.3115/1072017.1072026.

Doddington, George R., Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. 2004. "The Automatic Content Extraction (ACE) Program Tasks, Data, and Evaluation." In *LREC*, 2:837–40. Lisbon. http://lrec.elra.info/proceedings/lrec2004/pdf/5.pdf.

Fort, Karën. 2016. *Collaborative Annotation for Reliable Natural Language Processing: Technical and Sociological Aspects*. John Wiley & Sons.

Miranda-Escalada, Antonio, Eulàlia Farré, and Martin Krallinger. 2020. "Named Entity Recognition, Concept Normalization and Clinical Coding: Overview of the Cantemist Track for Cancer Text Mining in Spanish, Corpus, Guidelines, Methods and Results." *IberLEF@ SEPLN*, 303–23.

Neves, Mariana, and Jurica Ševa. 2021. "An Extensive Review of Tools for Manual Annotation of Documents." *Briefings in Bioinformatics* 22 (1): 146–63. https://doi.org/10/ggqtkq.

Nourani, Esmaeil, Evangelia-Mantelena Makri, Xiqing Mao, Sampo Pyysalo, Søren Brunak, Katerina Nastou, and Lars Juhl Jensen. 2025. "LSD600: The First Corpus of Biomedical Abstracts Annotated with Lifestyle–Disease Relations." *Database* 2025 (January):baae129. https://doi.org/10.1093/database/baae129.

Stenetorp, Pontus, 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*, 102–7.

Tjong Kim Sang, Erik F., 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*, 142–47. https://aclanthology.org/W03-0419.

Wagstaff, Kiri, Raymond Francis, Thamme Gowda, You Lu, Ellen Riloff, Karanjeet Singh, and Nina Lanza. 2018. "Mars Target Encyclopedia: Rock and Soil Composition Extracted From the Literature." *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (1). https://doi.org/10.1609/aaai.v32i1.11412.

Yimam, Seid Muhie, 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*, edited by Miriam Butt and Sarmad Hussain, 1–6. Sofia, Bulgaria: Association for Computational Linguistics. https://aclanthology.org/P13-4001/.

Zaghir, Jamil, Mina Bjelogrlic, Jean-Philippe Goldman, Soukaïna Aananou, Christophe Gaudet-Blavignac, and Christian Lovis. 2024. "FRASIMED: A Clinical French Annotated Resource Produced through Crosslingual BERT-Based Annotation Projection." In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, edited by Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, 7450–60. Torino, Italia: ELRA and ICCL. https://aclanthology.org/2024.lrec-main.657/.

Zaghir, Jamil, Mina Bjelogrlic, Jean-Philippe Goldman, Adel Bensahla, Yuanyuan Zheng, and Christian Lovis. 2024. "Beyond Tokens: Fair Evaluation of French Large Language Models for Clinical Named Entity Recognition." *Studies in Health Technology and Informatics* 316 (August):666–70. https://doi.org/10.3233/SHTI240502.

BAREC Demo: Resources and Tools for Sentence-level Arabic Readability Assessment

Kinda Altarbouch,[†] Khalid N. Elmadani,[‡] Ossama Obeid,[‡]
Hanada Taha-Thomure,[††] Nizar Habash[‡]
[†]Lableb AI
[‡]Computational Approaches to Modeling Language Lab, New York University Abu Dhabi
[††]Zai Arabic Language Research Centre, Zayed University
kinda@lableb.com,nizar.habash@nyu.edu

Abstract

We present BAREC Demo, a web-based system for fine-grained, sentence-level Arabic readability assessment. The demo is part of the Balanced Arabic Readability Evaluation Corpus (BAREC) project, which manually annotated 69,000 sentences (over one million words) from diverse genres and domains using a 19-level readability scale inspired by the Taha/Arabi21 framework, covering reading abilities from kindergarten to postgraduate levels. The project also developed models for automatic readability assessment. The demo provides two main functionalities for educators, content creators, language learners, and researchers: (1) a Search interface to explore the annotated dataset for text selection and resource development, and (2) an Analyze interface, which uses trained models to assign detailed readability labels to Arabic texts at the sentence level. The system and all of its resources are accessible at https://barec.camel-lab.com.

1 Introduction

Text readability refers to the degree to which a written text is easy to read and understand, based on linguistic and structural features such as vocabulary, syntax, and sentence complexity. It reflects how well a text aligns with the reader's proficiency, influencing comprehension, reading speed, retention, and engagement. This is especially critical in educational and language learning contexts, where matching texts to learners' reading levels can significantly enhance learning outcomes (DuBay, 2004).

Assessing the readability of Arabic text poses significant challenges due to its rich morphology, complex syntax, and ambiguous orthography (owing to optional diacritics), as well as the diglossic landscape of Modern Standard Arabic (MSA) and diverse dialects (Ferguson, 1959; Habash, 2010; Obeid et al., 2020). Existing readability efforts vary in focus on document-level sentence-level, or word-level measures, but with generally limited support for diverse text genres (Alhafni et al., 2024; Niraula et al., 2014; Hazim et al., 2022; Jiang et al., 2020). This leaves a critical need for tools that can provide fine-grained, sentence-level assessment, which is essential for precise content adaptation.

To address these issues, the Balanced Arabic Readability Evaluation Corpus (BAREC) (Elmadani et al., 2025) aims to bring fine-grained readability assessment to the sentence level. By focusing on sentences and shorter text segments rather than the entire text, we gain more control and granularity in modeling text difficulty, enabling new applications, such as detailed readability feedback and metrics for automated text simplification. BAREC provides fine-grained, sentence-level readability annotations across 19 distinct levels, from kindergarten to postgraduate comprehension (Taha-Thomure, 2017; Habash et al., 2025).

In this paper, we present the BAREC demo, a robust web-based platform developed under the BAREC project. The demo integrates two powerful and complementary tools: **Search** is a browser providing access through the manually annotated 69K sentences (1M words) in the BAREC corpus, and **Analyze** is an automated readability assessment system, leveraging the machine learning techniques trained on the BAREC corpus (Elmadani et al., 2025). The BAREC demo target audience includes educators, language learners, content creators, and researchers interested in Arabic NLP and education. The system and all of its resources are publicly accessible.[1,2,3,4,5]

Next we present related work (§2), introduce BAREC resources and tools (§3), outline system design (§4), illustrate usage with examples (§5), and report on a brief evaluation (§6).

[1]Project Resources: https://barec.camel-lab.com
[2]Search: https://barec.vercel.app/en/search
[3]Analyze: https://barec.vercel.app/en/analyze
[4]Video: https://barec.camel-lab.com/emnlp-demo
[5]Licenses: Code is MIT and Data is CC-BY-SA-4.0.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 950–959
November 4-9, 2025

2 Background & Related Work

2.1 Arabic Readability Challenges

Assessing readability in Arabic is especially difficult due to the language's linguistic complexity. Its rich morphology leads to extensive lexical variation, complicating vocabulary assessment across topics and dialects. The diglossic split between Modern Standard Arabic and regional varieties adds further difficulty, particularly for less proficient readers. Orthographic features, such as the optional use of diacritics, can significantly alter meaning and pronunciation, affecting comprehension. Arabic's flexible syntax also makes it hard to estimate sentence-level difficulty using simple metrics like word counts or n-grams. These challenges require advanced preprocessing for consistent manual annotation and robust modeling for accurate automatic assessment. The BAREC project addresses these issues through tailored guidelines, a carefully annotated corpus, and tools designed for fine-grained sentence-level analysis.

2.2 Existing Arabic Readability Efforts

In the context of Arabic, there were several significant efforts to define, annotate and model text readability levels. BAREC draws on the Taha/Arabi21 framework (Taha-Thomure, 2017), which introduced a 19-level system for book-level text leveling widely used in children's literature. The SAMER project (Al Khalil et al., 2017) developed a Google Docs add-on for word-level readability visualization and substitution (Hazim et al., 2022) as well as a readability leveled Arabic thesaurus interface (Al Khalil et al., 2018; Jiang et al., 2020). Arabic CEFR-aligned initiatives include KELLY (Kilgarriff et al., 2014), ZAEBUC (Habash and Palfreyman, 2022), ReadMe++ (Naous et al., 2023), and the Arabic Vocabulary Profile Project (Soliman and Familiar, 2024). Automatic readability efforts range from formula-based systems like AARI (Al Tamimi et al., 2014) and OSMAN (El-Haj and Rayson, 2016) to ML-based approaches using linguistic features (Forsyth, 2014; Saddiki et al., 2018) or annotated data (Liberato et al., 2024).

BAREC advances upon these efforts by creating a large, sentence-level, manually annotated corpus with 19 readability levels across genres, and training automatic readability models on it (Habash et al., 2025; Elmadani et al., 2025)

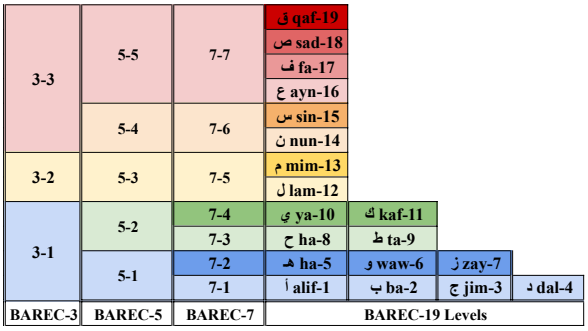


Figure 1: The BAREC *Pyramid* illustrates the BAREC levels and three collapsed variants.

3 BAREC Resources & Tools

The BAREC project provides resources to help educational writers tailor texts to their audience through readability assessment. The current demo highlights two key components: a search interface for exploring the large annotated corpus, and an AI module that predicts sentence-level readability.

3.1 BAREC Corpus

The BAREC corpus (Habash et al., 2025; Elmadani et al., 2025) is a large-scale Arabic readability dataset containing over 69 thousand sentences (comprising 1 million words) with sentence-level annotations across 19 readability levels. The BAREC pyramid (Figure 1) illustrates the scaffolding of these levels, and three collapsed versions of level size 7, 5, and 3. The corpus spans a wide educational range and diverse topics, compiled from 1,922 documents manually categorized into three domains: **Arts & Humanities**, **Social Sciences**, and **STEM**, and three readership groups: **Foundational**, **Advanced**, and **Specialized**. Texts were sourced from 30 different resources, all selected to comply with copyright restrictions. Approximately 25% of the sentences were manually transcribed from previously unavailable sources.

3.2 BAREC AI Component

The AI component of the BAREC demo website is designed to predict the readability level of a given sentence. For this, we fine-tune AraBERTv02 (Antoun et al., 2020) on the BAREC corpus using cross-entropy loss (CE), following the AraBERTv02+Word+CE configuration from Elmadani et al. (2025), and implement it with the `Transformers` library (Wolf et al., 2019). This configuration performs the best among the setups that do not require any data pre-processing. Model

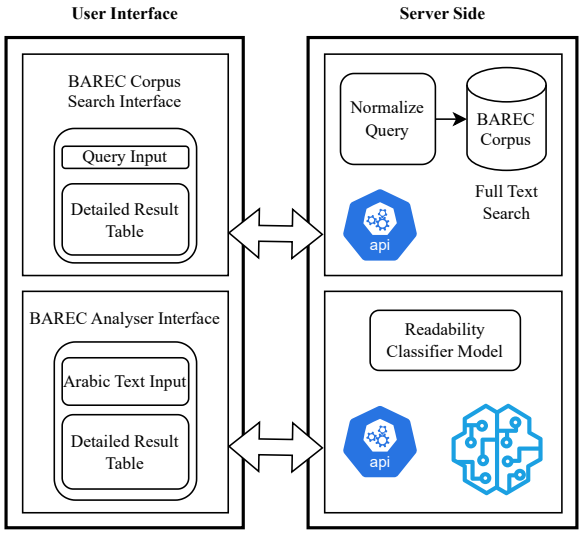


Figure 2: Overview of the BAREC System Architecture. This diagram illustrates the interaction between the user interface components (BAREC Corpus Search Interface and BAREC Analyze Interface) and the server-side services via APIs, including the BAREC Corpus for full-text search and the Readability Classifier Model.

training is performed on an NVIDIA V100 GPU for six epochs, with a learning rate of 5×10^{-5}, a batch size of 64, and a cross-entropy loss function for multi-class classification over 19 readability levels. We report of its performance in §6.

4 System Design and Implementation

At a high level, the BAREC demo follows a classic web-based client-server architecture, with additional linguistic processing and quality control components. Figure 2 provides a system design overview. The core components include: (1) a database and back-end services for managing the annotated corpus; (2) a linguistic processing module that analyzes sentences and fetches annotated dataset information; (3) a front-end web interface for users; and (4) BAREC Analyzer model, an automatic readability assessment system. We discuss these components in some detail next.

4.1 Design Considerations

The BAREC demo design prioritizes efficient search, data accessibility, and a user-friendly interface. Following are the key design choices we made to address the unique challenges of Arabic text processing, large dataset management, and responsive web delivery.

Efficient Arabic Full-Text Search A primary design goal was to enable highly effective and accurate search across a large corpus of Arabic text. This necessitates specialized handling of Arabic script complexities, including diacritics, letter elongations (Kashidas), and character variations (e.g., different forms of Alif Hamza). In addition to these orthographic challenges, the system also addresses some of Arabic's rich morphological structure. For example, a search for the word تفسير *tfsyr*[6] 'explanation' should return morphologically related forms like تفسيرها *tfsyrhA* 'its explanation', التفسير *Altfsyr* 'the explanation', and تفسيرات *tfsyrAt* 'explanations'. We apply orthographic normalization (removing Hamzas, collapsing Alif Maqsura and Yeh, etc.) on both the indexing and query sides. We also use partial matching to address Arabic's rich concatenative morphology.[7] Without such processing, Arabic's orthographic ambiguity and morphological variability would severely limit search performance.

Modular and Maintainable Architecture The system is designed with a modular architecture that cleanly separates the back-end (API and data logic) from the front-end (user interface). Built using the Express.js framework, this structure improves maintainability, simplifies debugging, and enables independent development and scaling of each layer.

Responsive Web Interface The front-end is designed to be accessible via a web browser, providing an intuitive interface for users to interact with the search and analysis functionalities. The use of EJS (Embedded JavaScript) templating ensures dynamic content rendering and language internationalization (Arabic/English) support, catering to a diverse user base.

API-Driven Communication The interaction between the front-end and back-end is entirely driven by RESTful APIs. This design allows for a decoupled system, enabling future expansion to other client types (e.g., mobile applications) without requiring significant changes to the back-end.

4.2 Main Interfaces

The system provides two main Web user interfaces.

Search Interface This interface enables users to explore the BAREC pre-annotated. Users can query the corpus by specific criteria such as *book*,

[6] Arabic HSB Romanization (Habash et al., 2007).

[7] We leave matching on morphological roots or across templatic variants to future work.

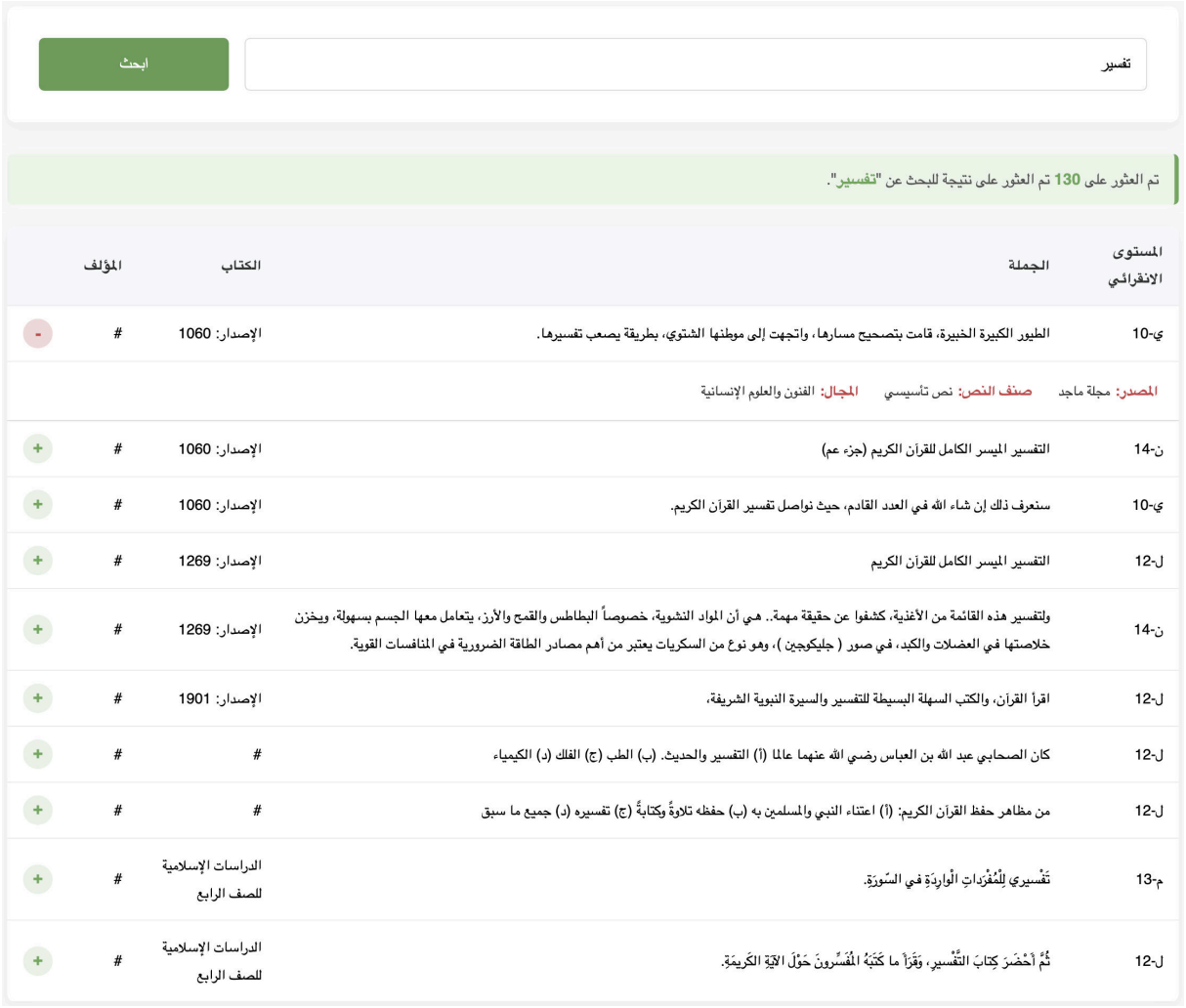


Figure 3: BAREC Demo search interface. The main search UI allows querying by sentence, author, or book. Users can navigate annotated dataset entries and expand metadata by clicking the (+) icon. The query تفسير *tfsyr* 'explanation' returns morphologically related forms. **The English version is in Appendix A Figure 5.**

author, or *sentence words* (Figure 3). The back-end performs a full-text search in the PostgreSQL database, processes the results, and sends a response back to the UI. The search results are dynamically displayed in a paginated table, showing the retrieved text segments along with their associated readability levels, as well as the book title, author, domain, and other metadata.

Analyze Interface This component allows users to input arbitrary Arabic text into a dedicated text area. Upon submission, the system processes the text in real time and provides an immediate readability assessment for each sentence. The results include a visual readability bar and a detailed tabular breakdown (Figure 4). The table presents three key elements: (a) an estimate of the overall readability level, defined as the highest predicted level among all sentences; (b) sentence-level labels across 19 fine-grained readability levels, reflecting individual sentence complexity; (c) and a visual summary of the distribution of readability levels within the text, helping users assess variation across the passage.

4.3 Implementation

The BAREC demo is a web application with a modular front-end/back-end architecture for efficient processing and retrieval of Arabic text.

Back-end Built with Express.js,[8] the back-end provides RESTful APIs for accessing the annotated dataset and for analyzing user-submitted text using the BAREC readability model. All data is stored in a PostgreSQL[9] database, which includes metadata such as source, author, readability levels, and

[8] https://expressjs.com
[9] https://www.postgresql.org

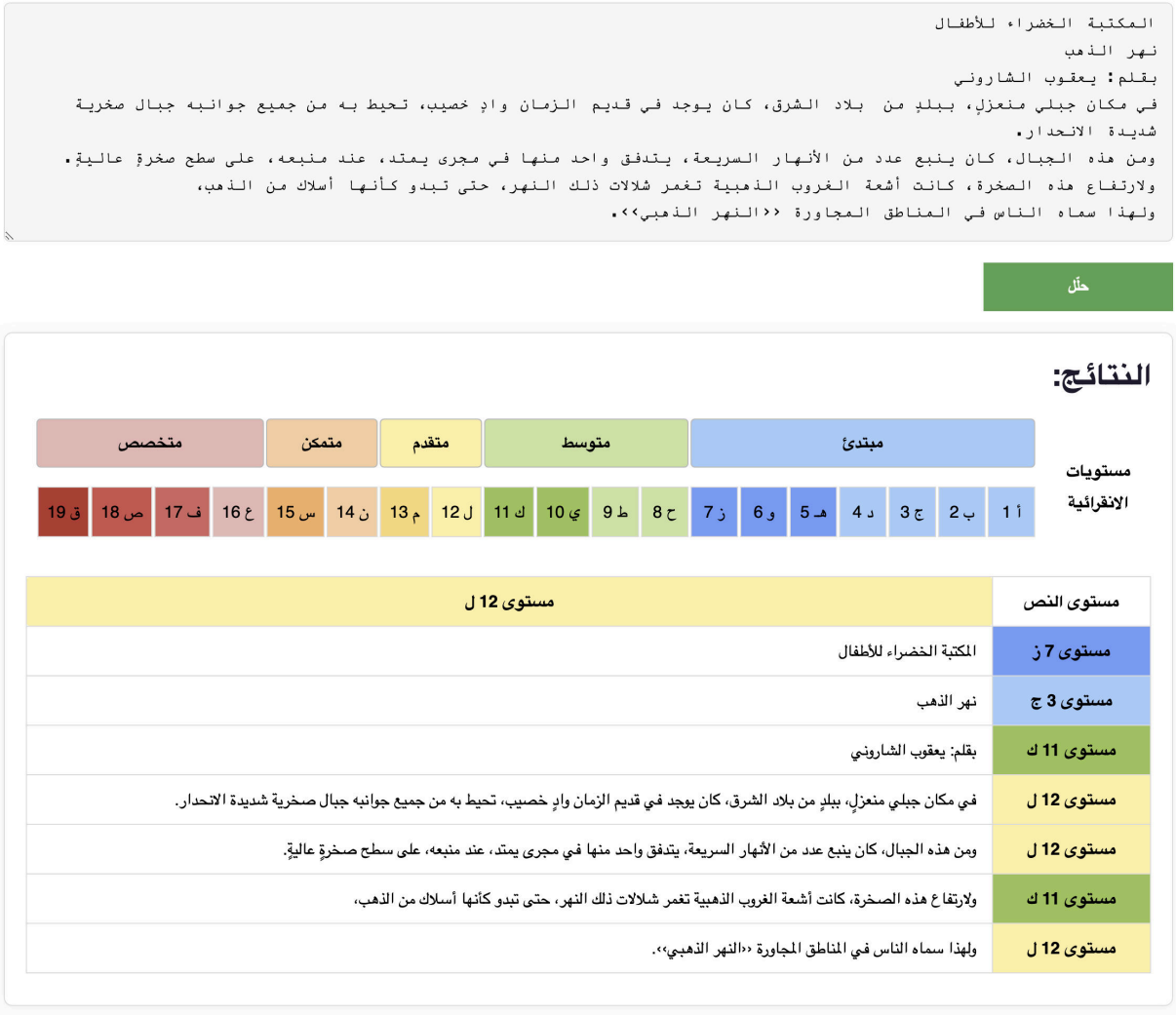


Figure 4: The BAREC Demo Analyze Interface, including the text input area, the Readability Bar (colored guide bar for readability level), the overall Text Readability Level, and the detailed, sentence-by-sentence readability levels. **The English version of the interface is in Appendix B Figure 6**.

domain. The AI model performs sentence-level analysis and returns results to the front-end.

Front-end The interface uses EJS templating for dynamic server-side rendering and interaction with the back-end services.

5 Use Case and Example Scenario

This section demonstrates the BAREC Demo's Search and Analyze utilities through typical user interactions and output interpretation.

5.1 Search Interface Walkthrough

The Search interface empowers users to explore the BAREC annotated corpus (Figures 3 and 5).

- **Query Input** Users navigate to the Search page, where a prominent search bar is available for keyword input.
- **Search Parameters** Users can explore the corpus by searching across various dimensions such as مؤلف أو كتاب أو جملة *Author, Book, or Sentence*. For instance, a user might search for all sentences by a particular author to analyze their writing style across different readability levels.
- **Dynamic Table Rendering** Search results are dynamically displayed in a responsive, paginated table, allowing users to browse through the retrieved segments efficiently.
- **Result Columns** The search results table presents key columns (from right to left): *Readability Level*, *Sentence*, *Book*, and *Author*. Additional metadata, such as Source, Text Class, and Domain are also be available through an expansion by clicking on (+) at the

leftmost part of the record.

In Figure 3, the search term is تفسير *tfsyr* 'explanation'. The first record with the expanded metadata view has the word تفسيرها *tfsyrhA* 'its explanation' in a sentence from a children's magazine.

5.2 Analyze Interface Walkthrough

The Analyze interface provides real-time readability assessment of user-provided Arabic text (Figures 4 and 6).

Text Input Users navigate to the Analyze Interface, which presents a clear text area for input. They can enter any Arabic text, ranging from a single sentence to a longer passage.

Analysis Process Upon clicking the *Analyze* button, the system processes the input text using its fine-grained readability models. The back-end sends the text to the analysis server, which returns detailed readability levels.

Readability Bar The system then displays a *Readability Row* listing levels from level 1أ *1 Alif* to 19ق *19 Qaf*. This colored bar provides an immediate, intuitive graphical guide of the detected readability levels for the overall text or for individual sentences within the input. The colors and labels correspond directly to the 19 BAREC levels, offering a quick visual summary of the text's complexity. For example, predominantly green labels might indicate an easier text, while a mix of yellow and red could signal varying levels of difficulty.

Detailed Result Table Below the readability bar, a detailed tabular output is shown. The header of this table indicates the Text Level (defined as the maximum of the text sentence levels). Next, the table provides a sentence-by-sentence breakdown of the analysis, listing each sentence from the input text alongside its predicted readability level with matching color for easy visualization. This allows users to quickly pinpoint specific sentences that contribute to the overall text complexity.

In Figure 4, the text is the opening to a children's book called نهر الذهب 'The Golden River'. The text snippet includes seven sentences/fragments, three of which are at level 12ل 12 Lam (around fifth grade reading), two are at level 11ك 11 Kaf (around fourth grade reading), and the rest (titles of series and book) are further below. The overall text level is 12ل 12 Lam.

6 Evaluation

We evaluate the Analyzer using Elmadani et al.'s (2025) AraBERTv02+Word+CE setup, and report on the following metrics:

- **Acc:** Exact match accuracy for 19-, 7-, 5-, and 3-level groupings (Figure 1).
- **±1 Acc**[19]**:** Accuracy within one level of the gold label (19-level).
- **Dist:** Average absolute error between prediction and gold.
- **Quadratic Weighted Kappa (QWK):** Weighted agreement score penalizing larger errors (Cohen, 1968; Doewes et al., 2023).

On the BAREC test set, the AraBERTv02+Word+CE model (Antoun et al., 2020) (see Section 3.2) achieves an exact accuracy of 55.8% on the 19-level scheme (**Acc**[19]). When collapsing the levels, the accuracy jumps to 64.9%, 69.1%, and 74.7% for the 7-, 5-, and 3-level schemes, respectively. The adjacent accuracy (**±1 Acc**[19]) reaches 69.4%, indicating that most predictions are close to the correct level. The average **distance** between predictions and gold labels is 1.09 (out of 19), and the **QWK** score is **81.0%**, reflecting strong overall agreement.

7 Conclusions and Future Work

The BAREC demo fills a key gap in Arabic NLP by offering an intuitive interface for sentence-level readability assessment across 19 levels. It supports educators and content creators through two core tools: a search interface for exploring a large, manually annotated corpus, and an analyzer for real-time text evaluation.

Future work includes expanding the corpus to 10 million words across a wider range of genres and dialects, refining annotation guidelines based on ongoing insights and user feedback, enhancing model accuracy through advanced linguistic features and machine learning techniques, and conducting in-depth user studies to guide usability improvements and feature development.

Acknowledgments

The BAREC project is supported by the Abu Dhabi Arabic Language Centre (ALC) / Department of Culture and Tourism, UAE. We acknowledge the support of the High Performance Computing Center at New York University Abu Dhabi.

Limitations

Readability assessment, despite rigorous guidelines and extensive training of annotators, inherently involves a degree of subjectivity. This can lead to some variability in annotation decisions and, consequently, in model predictions. While the BAREC corpus is extensive and diverse, it may not fully capture the entire linguistic landscape of the Arab world, nor all possible genres, styles, or regional nuances. This limitation could affect the generalizability of the models, particularly for less represented text types. Furthermore, potential biases or gaps might exist in the corpus due to the selection of source materials or inherent limitations in the manual annotation process.

Ethics Statement

The BAREC project is committed to upholding high ethical standards. All data utilized in the corpus curation process has been sourced responsibly and legally, respecting intellectual property rights. The manual annotation process is conducted with transparency and fairness, involving multiple annotators to mitigate individual biases and ensure reliability. All annotators are compensated fairly for their contributions. It is explicitly recognized that readability measures, like any AI assessment system, could potentially be misused, for instance, for biased profiling of individuals based on their writing style. However, this is not the intended purpose of BAREC, and such malicious use is strongly discouraged. BAREC is designed to function as a supportive tool for educators, learners, and content creators, rather than as a definitive or judgmental evaluator. The commitment to making the BAREC corpus and its associated guidelines openly accessible promotes transparency, reproducibility of research, and fosters collaborative advancements within the Arabic language research community.

References

Muhamed Al Khalil, Nizar Habash, and Hind Saddiki. 2017. Simplification of Arabic masterpieces for extensive reading: A project overview. *Procedia Computer Science*, 117:192–198.

Muhamed Al Khalil, Hind Saddiki, Nizar Habash, and Latifa Alfalasi. 2018. A Leveled Reading Corpus of Modern Standard Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

Abdel Karim Al Tamimi, Manar Jaradat, Nuha Al-Jarrah, and Sahar Ghanem. 2014. AARI: automatic Arabic readability index. *International Arab Journal of Information Technology*, 11(4):370–378.

Bashar Alhafni, Reem Hazim, Juan David Pineros Liberato, Muhamed Al Khalil, and Nizar Habash. 2024. The SAMER Arabic text simplification corpus. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16079–16093, Torino, Italia. ELRA and ICCL.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Afrizal Doewes, Nughthoh Arfawi Kurdhi, and Akrati Saxena. 2023. Evaluating quadratic weighted kappa as the standard performance metric for automated essay scoring. In *Proceedings of the 16th International Conference on Educational Data Mining*, pages 103–113, Bengaluru, India. International Educational Data Mining Society.

William H DuBay. 2004. The principles of readability. *Online Submission*.

Mahmoud El-Haj and Paul Rayson. 2016. OSMAN: A novel Arabic readability metric. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Portorož, Slovenia.

Khalid N. Elmadani, Nizar Habash, and Hanada Taha-Thomure. 2025. A large and balanced corpus for fine-grained Arabic readability assessment. In *Findings of the Association for Computational Linguistics: ACL 2025*, Vienna, Austria. Association for Computational Linguistics.

Charles F Ferguson. 1959. Diglossia. *Word*, 15(2):325–340.

Jonathan Forsyth. 2014. Automatic readability prediction for modern standard Arabic. In *Proceedings of the Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT)*.

Nizar Habash and David Palfreyman. 2022. ZAEBUC: An annotated Arabic-English bilingual writer corpus. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Marseille, France.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer, Netherlands.

Nizar Habash, Hanada Taha-Thomure, Khalid N. Elmadani, Zeina Zeino, and Abdallah Abushmaes. 2025. Guidelines for fine-grained sentence-level Arabic readability annotation. In *Proceedings of the 19th Linguistic Annotation Workshop (LAW-XIX-2025)*,

Vienna, Austria. Association for Computational Linguistics.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Reem Hazim, Hind Saddiki, Bashar Alhafni, Muhamed Al Khalil, and Nizar Habash. 2022. Arabic word-level readability visualization for assisted text simplification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 242–249, Abu Dhabi, UAE. Association for Computational Linguistics.

Zhengyang Jiang, Nizar Habash, and Muhamed Al Khalil. 2020. An online readability leveled Arabic thesaurus. In *Proceedings of the 28th International Conference on Computational Linguistics: System Demonstrations*, pages 59–63, Barcelona, Spain (Online). International Committee on Computational Linguistics (ICCL).

Adam Kilgarriff, Frieda Charalabopoulou, Maria Gavrilidou, Janne Bondi Johannessen, Saussan Khalil, Sofie Johansson Kokkinakis, Robert Lew, Serge Sharoff, Ravikiran Vadlapudi, and Elena Volodina. 2014. Corpus-based vocabulary lists for language learners for nine languages. *Language Resources and Evaluation*, 48(1):121–163.

Juan Liberato, Bashar Alhafni, Muhamed Khalil, and Nizar Habash. 2024. Strategies for Arabic readability modeling. In *Proceedings of The Second Arabic Natural Language Processing Conference*, pages 55–66, Bangkok, Thailand.

Tarek Naous, Michael J. Ryan, Anton Lavrouk, Mohit Chandra, and Wei Xu. 2023. Readme++: Benchmarking multilingual language models for multi-domain readability assessment. *Preprint*, arXiv:2305.14463.

Nobal Niraula, Vasile Rus, Rajendra Banjade, Dan Stefanescu, William Baggett, and Brent Morgan. 2014. The DARE corpus: A resource for anaphora resolution in dialogue based intelligent tutoring systems. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3199–3203, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. CAMeL tools: An open source python toolkit for Arabic natural language processing. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France.

Hind Saddiki, Nizar Habash, Violetta Cavalli-Sforza, and Muhamed Al Khalil. 2018. Feature optimization for predicting readability of arabic l1 and l2. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 20–29.

Rasha Soliman and Laila Familiar. 2024. Creating a cefr arabic vocabulary profile: A frequency-based multi-dialectal approach. *Critical Multilingualism Studies*, 11(1):266–286.

Hanada Taha-Thomure. 2017. *Arabic Language Text Leveling* (معايير هنادا طه لتصنيف مستويات النصوص العربية). Educational Book House (دار الكتاب التربوي للنشر والتوزيع).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

A BAREC Demo Search Interface (English Version)

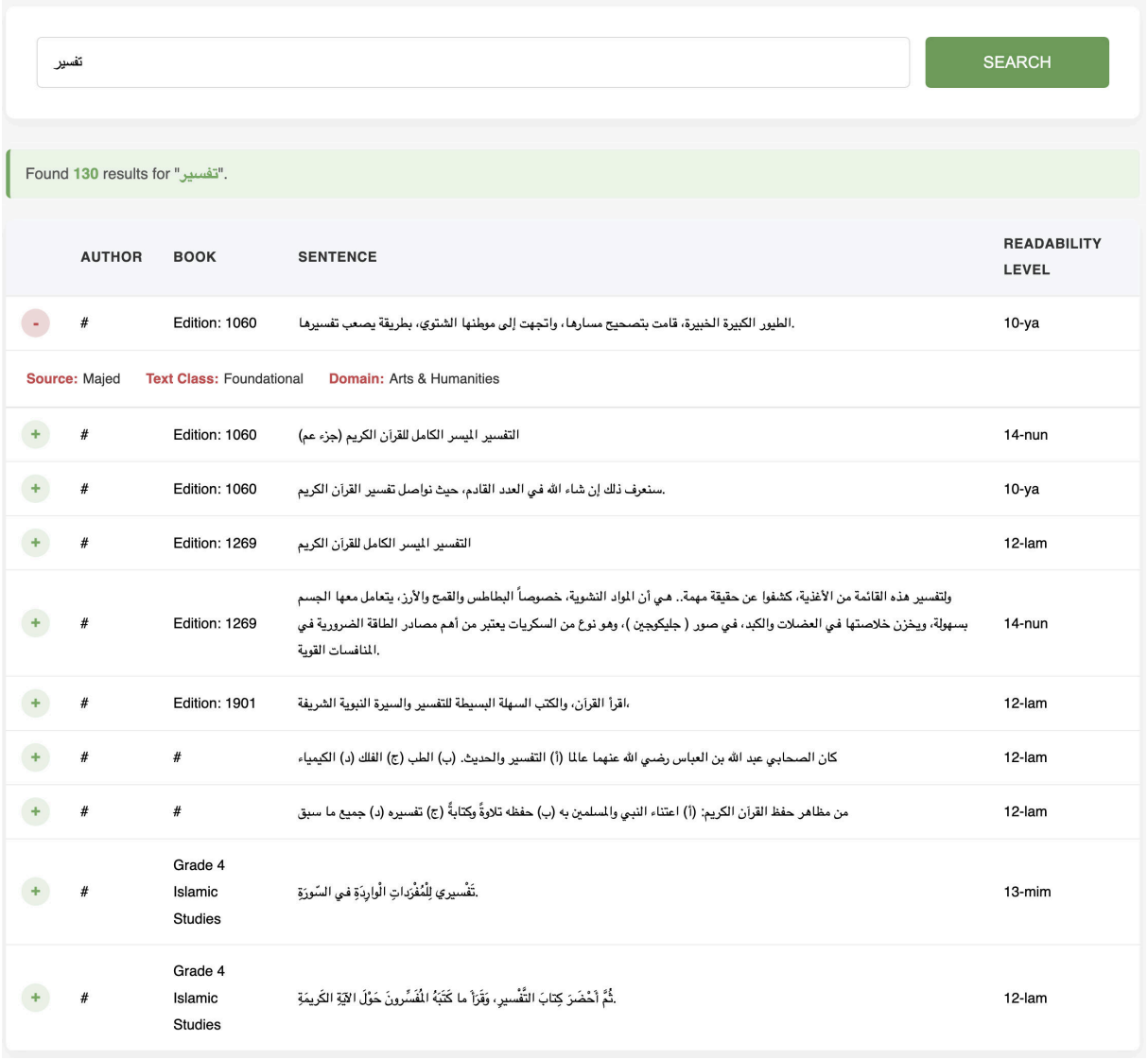


Figure 5: BAREC Demo search interface. The main search UI allows querying by sentence, author, or book. Users can navigate annotated dataset entries and expand metadata by clicking the (+) icon. The query تفسير *tfsyr* 'explanation' returns morphologically related forms.

B BAREC Demo Analyze Interface (English Version)

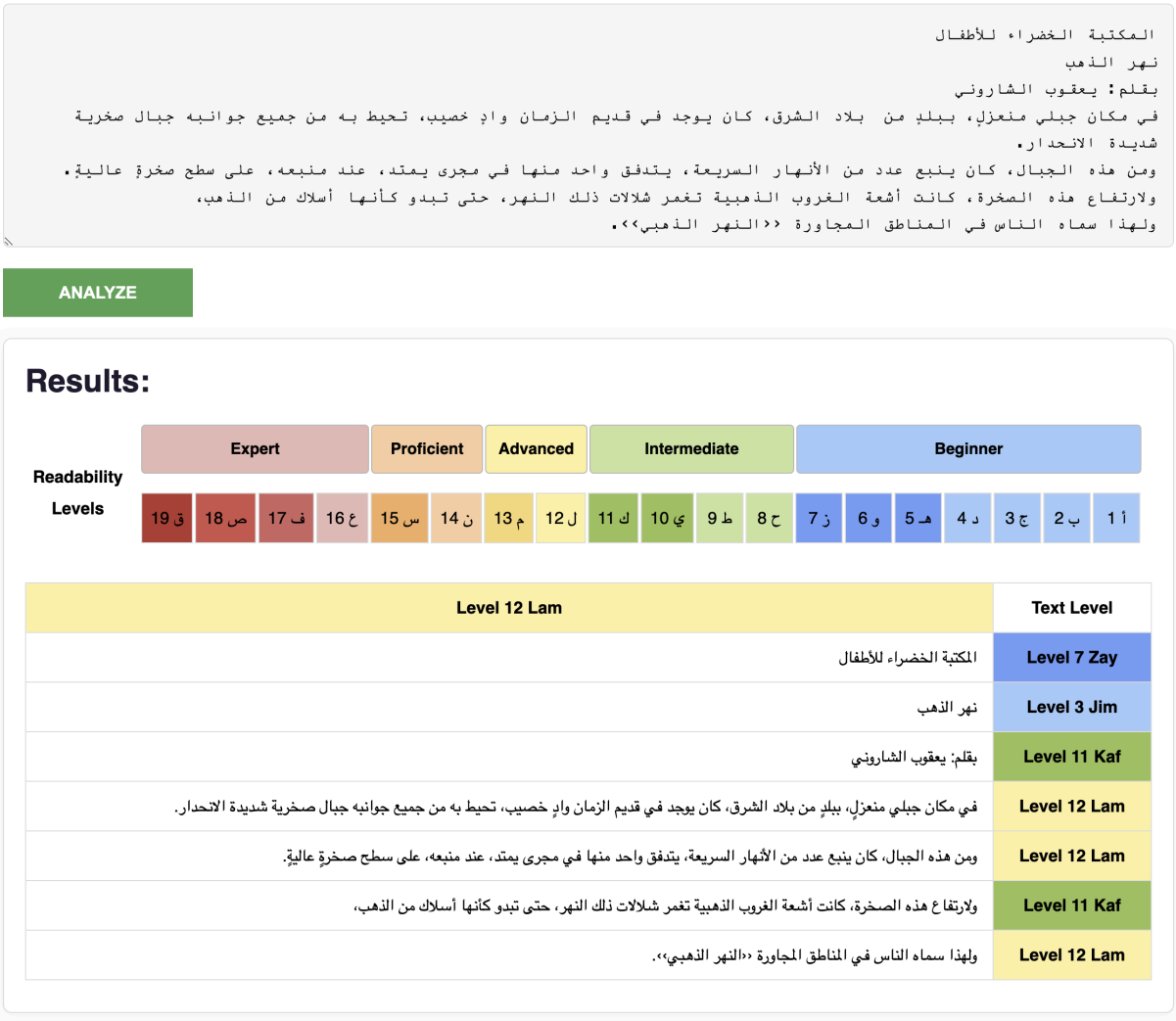


Figure 6: The BAREC Demo Analyze Interface, including the text input area, the Readability Bar (colored guide bar for readability level), the overall Text Readability Level, and the detailed, sentence-by-sentence readability levels.

Easy Dataset: A Unified and Extensible Framework for Synthesizing LLM Fine-Tuning Data from Unstructured Documents

Ziyang Miao[1], Qiyu Sun[1], Jingyuan Wang[1], Yuchen Gong[1], Yaowei Zheng[1], Shiqi Li[2]*, Richong Zhang[1]†
[1]School of Computer Science and Engineering, Beihang University, China
[2]Independent Researcher
Open-source repository: **https://github.com/ConardLi/easy-dataset**
Demonstration video: **https://youtu.be/HlyvdE1ASRk**

Abstract

Large language models (LLMs) have shown impressive performance on general-purpose tasks, yet adapting them to specific domains remains challenging due to the scarcity of high-quality domain data. Existing data synthesis tools often struggle to extract reliable fine-tuning data from heterogeneous documents effectively. To address this limitation, we propose Easy Dataset, a unified framework for synthesizing fine-tuning data from unstructured documents via an intuitive graphical user interface (GUI). Specifically, Easy Dataset allows users to easily configure text extraction models and chunking strategies to transform raw documents into coherent text chunks. It then leverages a persona-driven prompting approach to generate diverse question-answer pairs using public-available LLMs. Throughout the pipeline, a human-in-the-loop visual interface facilitates the review and refinement of intermediate outputs to ensure data quality. Experiments on a financial question-answering task show that fine-tuning LLMs on the synthesized dataset significantly improves domain-specific performance while preserving general knowledge. The source code and installable package are available at https://github.com/ConardLi/easy-dataset and have garnered over 10,000 GitHub stars.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across general-purpose applications (Achiam et al., 2023; Yang et al., 2025), leading to their widespread deployment in both academia and industry. Nevertheless, as real-world applications increasingly demand domain-specific knowledge, effectively adapting LLMs to specialized scenarios remains a persistent challenge. Among existing approaches, supervised fine-tuning (Ouyang et al., 2022; Taori et al., 2023) has emerged as a practical and effective method for domain adaptation, enabling models to internalize domain-specific knowledge through learning from curated labeled data. Yet the effectiveness of fine-tuning depends on the availability of large-scale, high-quality datasets tailored to the target domain. In practice, constructing such datasets manually is often costly and time-consuming, particularly in domains that require expert annotations. As a result, automated data synthesis (Wang et al., 2024b; Tan et al., 2024) has gained increasing attention as an attractive alternative to reduce human effort while preserving data diversity and domain relevance.

Despite the promise of automated data synthesis, implementing it in practice remains non-trivial due to two primary challenges: effectively parsing heterogeneous and noisy source documents and generating large-scale, high-quality supervised data. The first challenge stems from the structural complexity of real-world documents, which often contain a combination of unstructured and semi-structured elements, including free text, tables, and figures. As a result, standard parsing methods often struggle to handle this variability robustly and consistently (Zhang et al., 2024), resulting in incomplete or inaccurate information extraction. The second challenge involves generating diverse and faithful question-answer (QA) pairs that accurately capture the underlying domain knowledge. Simply reusing or repeating synthesized QA pairs can lead to overfitting and degrade downstream performance after fine-tuning. Therefore, an effective data augmentation strategy is crucial to generate varied examples while preserving semantic correctness and consistency with domain-specific content.

Recent studies have investigated automated data synthesis methods, such as constructing fine-tuning datasets based on predefined topics (Chesterfield Laboratories Inc., 2025) or problem sets (Canto et al., 2024; Marten et al., 2025). However, these tools generally lack reliable capabilities for parsing

*Corresponding author: 1009903985@qq.com
†Corresponding author: zhangrc@act.buaa.edu.cn

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 960–968

Table 1: Comparison of existing synthetic data generation tools.

Feature	Distilabel	Kiln	Curator	Data-Juicer	GraphGen	Easy Dataset
Adaptive Parsing	✓		✓	✓	✓	✓
Hybrid Chunking			✓	✓	✓	✓
Persona-Driven Data Synthesis						✓
Human-In-The-Loop		✓				✓
End-to-End GUI		✓		✓	✓	✓

unstructured documents to extract domain-specific data. Chen et al. (2024a,b) introduce various document processing techniques, but they fall short of yielding high-quality QA pairs that are directly suitable for fine-tuning. Consequently, there remains a clear need for an end-to-end solution for synthesizing fine-tuning data from raw documents to effectively improve the performance of LLMs in domain-specific applications.

We introduce Easy Dataset, a unified framework for synthesizing fine-tuning data from unstructured documents through an intuitive graphical user interface (GUI). It comprises two main components: **adaptive document processing** and **persona-driven data synthesis**. To transform raw documents into coherent text chunks, we integrate a suite of text extraction models, enabling robust parsing across various formats. The extracted textual content is subsequently segmented into semantically coherent chunks using a hybrid chunking strategy. To construct high-quality datasets from these chunks, we adopt a persona-driven prompting approach that generates diverse and large-scale question-answer (QA) pairs by leveraging publicly available LLMs. Importantly, the GUI facilitates human-in-the-loop refinement, allowing users to review and improve the quality of the synthesized data. It further allows users to execute the entire data synthesis workflow without writing any code, greatly enhancing usability for non-technical users.

Our contributions are summarized as follows:

- We introduce Easy Dataset, an extensible end-to-end framework that unifies adaptive document processing and persona-driven data synthesis to automate the curation of high-quality fine-tuning data from unstructured documents, while minimizing manual effort.
- We develop an intuitive visual interface that enhances accessibility for non-technical users and facilitates efficient human-in-the-loop refinement to ensure data quality.
- Empirical results on a financial QA task show that fine-tuning LLMs on the synthesized data significantly improves domain-specific performance while retaining general knowledge.

2 Related Work

In recent years, numerous systems have been proposed to facilitate automated synthetic data generation. Distilabel (Canto et al., 2024) provides a modular framework with practical components for generating various types of synthetic data, including instruction-tuning and preference datasets. Kiln (Chesterfield Laboratories Inc., 2025) features a GUI-based platform that supports zero-shot data generation, topic tree construction, and reasoning data creation, along with an interactive interface for human review and rating. Curator (Marten et al., 2025) facilitates the synthesis of diverse data types, including reasoning, code execution, chart generation, and function-calling data, to support fine-tuning across a broad spectrum of downstream tasks. Data-Juicer (Chen et al., 2024a,b) supports multiple user interfaces and provides a rich set of data processing operators, making it highly customizable. GraphGen (Chen et al., 2025) leverages knowledge graphs to generate question-answer pairs from source documents through a visual interface. Despite their contributions, these tools still exhibit several notable limitations (Table 1). Many lack support for human-in-the-loop interaction, hindering opportunities for iterative quality refinement and alignment with user preferences. Others do not support synthesizing fine-tuning data based on source documents, making it difficult to efficiently construct domain-specific datasets. Moreover, few provide a fully end-to-end graphical user interface (GUI) that integrates the entire data synthesis pipeline, resulting in steep learning curves and limited usability for users with diverse technical backgrounds. These gaps underscore the need for a unified, accessible solution.

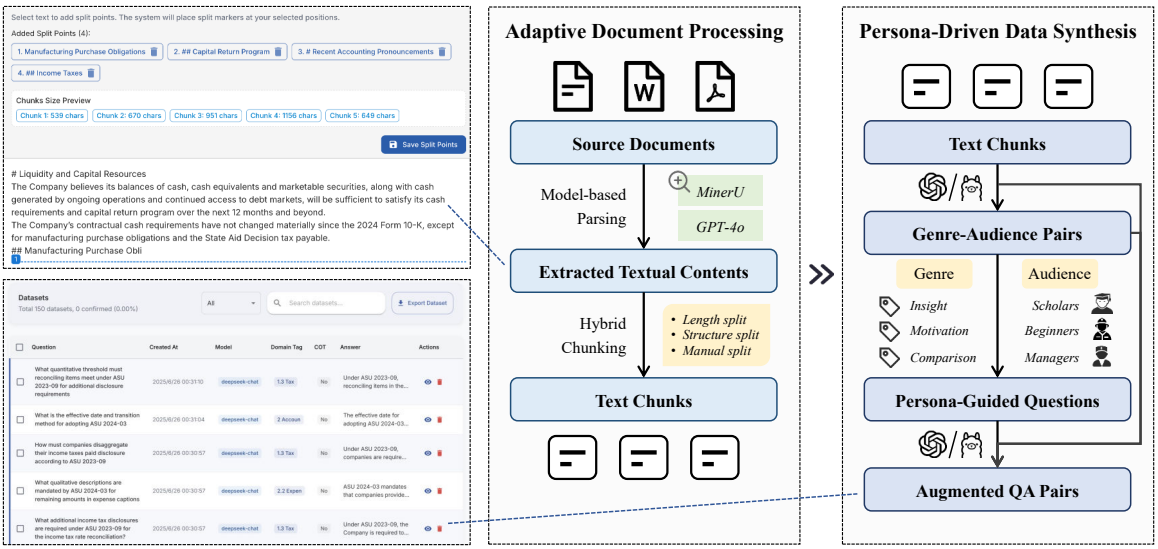


Figure 1: Overview of the Easy Dataset framework. The framework consists of two primary components: adaptive document processing and persona-driven data synthesis. In the first component, documents of different formats are processed via model-based parsing, followed by hybrid chunking to produce text chunks. In the second component, *Genre-Audience* pairs are created for each document to guide the construction of diverse question-answer pairs. Persona-guided questions are then generated to further increase diversity, and final augmented QA pairs are synthesized via knowledge-enhanced prompting to ensure factual consistency. The entire framework provides interactive human-in-the-loop refinement to maintain data quality.

3 Framework

We present Easy Dataset, a unified and extensible framework designed to synthesize high-quality fine-tuning datasets from unstructured documents. As illustrated in Figure 1, the framework implements a visual human-in-the-loop pipeline that enables interactive refinement at all stages of dataset construction. It begins with adaptive document processing, leveraging vision-language models (VLMs) to extract textual content from raw documents. The extracted text is then segmented using a hybrid chunking strategy that combines length-based, structure-based, and manual splitting. In the QA generation stage, Easy Dataset supports both naive and persona-driven data synthesis to generate stylistically diverse and semantically grounded QA pairs. A modular model configuration interface enables flexible access to both API-based and locally deployed LLMs and offers fine-grained control over generation parameters. Finally, the framework supports export of datasets in multiple formats and is natively compatible with fine-tuning frameworks such as LlamaFactory (Zheng et al., 2024).

3.1 Adaptive Document Processing

Document Parsing Our framework supports extracting textual content from unstructured documents of various formats, including plain text, Markdown, DOCX, and PDF, which is essential for constructing high-quality datasets for downstream tasks. For plain text and Markdown documents, the original content is retained with minimal processing to preserve inherent semantics. In contrast, DOCX documents often lack explicit structure, necessitating preprocessing to extract meaningful text. To this end, we employ the lightweight Mammoth library (Williamson, 2015) to convert DOCX files into Markdown, which helps preserve the original semantics while avoiding unnecessary formatting noise. For PDF documents with relatively simple layouts, we use the pdf2md tool (Zillmann, 2017) to extract textual content efficiently. However, for PDFs with complex or mixed text-image compositions, rule-based methods often result in significant information loss. To address this limitation, we first apply layout analysis to automatically detect content regions within the document. Once the layout structure is established, textual regions are extracted directly, while visual regions are parsed using a vision-language model (VLM), which can accurately capture the content. This strategy ensures accurate text extraction and preserves key information, providing a reliable foundation for subsequent downstream data processing. In addition, our framework integrates with state-of-the-art PDF processing tools, including MinerU (Wang

et al., 2024a; He et al., 2024), providing users with greater flexibility to handle documents of varying complexity and scale.

Text Chunking To enable precise control over data sampling and ensure compatibility with existing LLM context windows, we propose *Hybrid-Chunking*, a structure-aware and adaptive preprocessing strategy that flexibly segments source documents into coherent text chunks. It allows users to configure chunk sizes and customize text delimiters, enabling adaptation to various content types, such as plain text, code snippets, or tabular data. The process begins with initial coarse-grained segmentation based on line breaks, followed by a hybrid splitting-and-merging routine: lengthy chunks are recursively split using user-defined delimiters, while neighboring short segments are merged if necessary to meet length constraints without breaking semantic units. Furthermore, for edge cases where automated rules may fail to work optimally, we provide a visual text-chunking interface that enables users to perform fine-grained manual adjustments, ensuring precise text segmentation. Users can flexibly tailor chunking policies and thresholds according to specific document content. This hybrid design achieves a balance between automation and user control, significantly improving the consistency and reliability of the resulting text chunks.

3.2 Persona-Driven Data Synthesis

Easy Dataset provides a flexible pipeline for generating question-answer (QA) pairs from text chunks. The process is highly configurable, supporting two generation modes: naive QA-pair generation and persona-driven QA-pair generation, the latter enabling the synthesis of diverse and stylistically controlled fine-tuning data.

Question Generation Based on the text chunks, questions are automatically constructed by prompting the LLMs. Specifically, each text chunk is combined with a customizable system prompt before being passed to the LLM for question generation. This system prompt enables fine-grained control over question style, target audience, and tone. For example, it can specify whether questions should be concise, elaborative, or directive in nature. To further improve robustness in real-world scenarios, we introduce a stochastic punctuation dropout mechanism. In particular, question marks are randomly removed to prevent the model from overrelying on punctuation cues, thereby enhancing the generalization performance of models fine-tuned on the synthesized QA dataset.

Answer Generation Based on the generated questions, we derive high-quality answers that remain faithful to the source content. We employ a knowledge-enhanced prompting strategy, where the prompt contains the question and its corresponding source content. This ensures that the LLM-generated answers remain semantically aligned with the source content, factually consistent, and relevant to the overall domain context. The generation style is configurable to meet task-specific requirements, such as producing concise, detailed, or explanatory answers, allowing users to tailor the output style and format as needed. When using reasoning LLMs with chain-of-thought (CoT) capabilities, such as DeepSeek-R1 (Guo et al., 2025), the intermediate reasoning steps are also included in the QA pairs to provide transparency and support more interpretable fine-tuning. This substantially improves interpretability and facilitates downstream error analysis. To ensure the quality and reliability of the generated QA pairs, we provide a post-generation refinement interface that allows users to manually review, edit, and verify answers. In addition, LLMs are used to automatically refine both final answers and their corresponding CoT reasoning traces, serving as an additional step to improve overall robustness and factual accuracy.

Persona-Driven QA-Pair Generation Ge et al. (2024) first introduced the persona-driven data synthesis approach, demonstrating that prompting LLMs with different communicative roles (personas) enables the generation of stylistically diverse outputs, which can significantly increase dataset size. Building on this insight, we integrate persona-driven data synthesis into our Easy Dataset framework via a two-stage pipeline inspired by the Massive Genre-Audience (MGA) method (Hao et al., 2025), which provides a systematic way to diversify output styles. Specifically, the pipeline first automatically generates diverse Genre-Audience (GA) pairs to guide stylistic conditioning and then produces multiple QA pairs conditioned on these personas, enriching the dataset with varied perspectives and tones. While MGA was originally developed to expand pretraining corpora by reformulating documents and reducing data redundancy, we adapt its core principles for QA pair synthesis in Easy Dataset to boost both the diversity and scale of the generated data for domain-specific fine-tuning.

Table 2: Persona-driven QA pairs synthesized by Easy Dataset. By varying the GA pair, our framework generates a diverse set of data examples that maintain high fidelity to the source documents.

Example 1: Financial News Summary for Executives and Business Readers
Genre: *Financial News Summary.* This genre condenses complex financial information into concise, accessible updates. It highlights key points like liquidity status, capital returns, and regulatory changes, making it suitable for readers who need quick, high-level insights without deep technical details.
Audience: *Busy Executives and General Business Readers.* Professionals who need to stay informed about financial developments but lack the time for in-depth analysis. They prefer succinct summaries that provide the essential takeaways without overwhelming detail.
Generated Question: *When is the Company expected to adopt ASU 2023-09, and what transition method will be used?*
Generated Answer: The Company will adopt ASU 2023-09 in its fourth quarter of 2026 using a prospective transition method. This update will enhance income tax disclosures by requiring additional details in the tax rate reconciliation and disaggregating taxes paid by federal, state, and foreign jurisdictions.
Example 2: In-depth Policy Discussion for Professionals and Policy Makers
Genre: *Tax and Accounting Policy Discussion.* This genre focuses on the implications of new tax and accounting policies, such as FASB updates. It explores how these changes affect corporate reporting, tax strategies, and financial transparency, often inviting debate and expert opinions.
Audience: *Tax Professionals and Policy Makers.* Experts in tax law and accounting standards who need to understand the practical and strategic impacts of regulatory changes. They are interested in how policies like ASU 2023-09 influence corporate behavior and financial disclosures.
Generated Question: *How does ASU 2023-09 require companies to disaggregate income taxes paid by jurisdiction?*
Generated Answer: ASU 2023-09 mandates that companies disaggregate their income taxes paid disclosure by federal, state, and foreign taxes. Additionally, for foreign taxes, further disaggregation is required for significant individual jurisdictions. This enhanced disclosure framework aims to provide greater transparency into the geographic distribution of tax payments, enabling stakeholders to better assess the company's tax profile and jurisdictional exposures. *(truncated)*

Our adapted persona-driven pipeline consists of two stages for synthesizing diverse, persona-guided QA pairs from unstructured documents. In the **Persona Synthesis Stage**, for each source document, we first leverage an LLM to automatically generate a set of unique (*Genre*, *Audience*) pairs based on the content. Each pair jointly defines a persona for the generation task. Specifically, *Genre* characterizes the inquisitive intent and dialogue style, while *Audience* profiles the cognitive state and knowledge background of the questioner. For example, a (Motivation, Beginners) persona guides the model to produce simple, encouragement-oriented questions that help novices build confidence. By introducing diverse Genre-Audience combinations, the synthesis process enables more comprehensive coverage of the original content from multiple perspectives. To enhance flexibility, users can also manually specify or refine GA pairs to better target specific domains or tasks. In the **Persona-Guided QA Generation Stage**, the synthesized personas guide the LLMs to generate diverse questions based on the text chunks from various perspectives. For each generated question, the model subsequently produces an answer conditioned on the question, the corresponding source text chunk, and the associated persona. This yields an augmented QA pair in which both the question and the answer are semantically grounded in the original content and stylistically consistent with the intended persona. This adaptation effectively transforms the original MGA method from a pretraining augmentation strategy into a powerful framework for synthesizing diverse, persona-guided fine-tuning data. As illustrated in Table 2, this persona-driven approach enables the synthesis of diverse and faithful QA pairs from a single source document, thereby enhancing the effective utilization of the original source corpora.

3.3 Model Configuration

To seamlessly integrate LLMs into the data generation pipeline, we propose a flexible model configuration module. This module allows users to configure and manage multiple LLMs for data synthesis through an intuitive visual interface with minimal setup effort. By specifying the model provider, API endpoint, API key, and model name, users can easily utilize their target models. The module also supports locally deployed models via Ollama (Ollama, 2023), ensuring flexibility for different deployment scenarios. Additionally, it provides fine-grained control over generation parameters (e.g., temperature and top-p sampling), enabling users to adapt the generation process to diverse data requirements and domain-specific constraints.

3.4 Dataset Export

Easy Dataset enables export of the generated QA pairs as standard dataset files in formats such as JSON, JSONL, and CSV. It also supports widely adopted data schemas such as Alpaca (Taori et al., 2023) and ShareGPT (Zheng et al., 2023). Users

Table 3: Performance comparison of the Qwen2.5-7B-Instruct model before and after fine-tuning with naive and persona-driven data synthesis on general-purpose and domain-specific benchmarks. **Bold** indicates the best result, and underline indicates the second-best.

Method	MMLU	CMMLU	HellaSwag	MATH	HumanEval	Avg.	Domain Knowledge
Base Model	62.0	77.2	**82.9**	**74.8**	**84.8**	**76.3**	3.2
Naive Data Synthesis	60.2	**78.4**	80.8	72.9	80.5	74.6	57.0
Persona-Driven Data Synthesis	**63.7**	77.7	80.8	74.1	81.1	75.5	**59.6**

can define custom export templates by specifying key fields such as question, answer, reasoning steps, and domain labels, allowing them to flexibly adapt outputs to diverse task-specific fine-tuning pipelines and community standards. Additionally, Easy Dataset offers seamless integration with the LlamaFactory training framework (Zheng et al., 2024) by automatically generating a compatible configuration file. Users can simply specify the path to this configuration file to enable direct use in LlamaFactory, eliminating manual setup and significantly lowering the barriers to fine-tuning.

4 Evaluation

4.1 Experimental Setup

To evaluate the quality of data synthesized by Easy Dataset, we collected five up-to-date (later than the knowledge cutoff) financial reports from public online sources as a representative domain example. We also built a domain-specific evaluation dataset consisting of 100 questions derived from the source documents. Then we used Easy Dataset to generate a training dataset and fine-tuned the Qwen2.5-7B-Instruct (Yang et al., 2024) model using the LlamaFactory framework. We compared the model's performance before and after fine-tuning using either the naive or persona-driven data synthesis pipeline. We evaluated the model on the domain-specific dataset and general-purpose benchmarks, including MMLU (Hendrycks et al., 2021), CMMLU (Li et al., 2024), HellaSwag (Zellers et al., 2019), MATH (Lightman et al., 2024), and HumanEval (Chen et al., 2021). For domain-specific evaluation, we employed the LLM-as-a-judge approach (Zheng et al., 2023) using the DeepSeek-V3 API (Liu et al., 2024) to provide reliable scores, with details provided in Appendix A. Other fine-tuning details are provided in Appendix B.

4.2 Results

Table 3 shows the evaluation results of Qwen2.5-7B-Instruct on several general-purpose benchmarks and the domain-specific evaluation dataset, comparing the base model with its fine-tuned variants trained using either the naive or persona-driven data synthesis methods. On general-purpose benchmarks, we observe that fine-tuning on domain-specific data preserves the model's general language capabilities, demonstrating that the synthetic dataset produced by Easy Dataset enables the model to acquire domain-specific knowledge while maintaining robust generalization. Notably, the persona-driven variant achieves the best results on the MMLU benchmark and competitive performance across most tasks, even outperforming the naive variant on several challenging benchmarks, indicating its potential for improving generalization through stylistic and semantic diversity. On the domain-specific evaluation dataset, both fine-tuned models demonstrate substantial gains over the base model, which performs poorly without exposure to up-to-date financial documents. Specifically, fine-tuning with the dataset generated by Easy Dataset using the naive synthesis pipeline achieves a score of 57.0, while incorporating persona-driven data synthesis further improves the score to 59.6. These results highlight the effectiveness of Easy Dataset in injecting timely, domain-specific knowledge while preserving the model's general capabilities.

5 Conclusion and Future Work

We introduced Easy Dataset, an end-to-end framework that automates the synthesis of domain-specific fine-tuning datasets. With human-in-the-loop control and an intuitive interface, it enables efficient generation of datasets with high quality, diversity, and usability. Empirical results demonstrate that Easy Dataset significantly improves domain adaptation for LLMs while preserving general capabilities and robustness. We plan to extend Easy Dataset in several directions, including supporting broader modalities (e.g., SQL, tables, multi-modal), integrating automatic quality monitoring, and developing advanced enrichment strategies to further enhance data variety and fidelity.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Álvaro Bartolomé Del Canto, Gabriel Martín Blázquez, Agustín Piqueres Lajarín, and Daniel Vila Suero. 2024. Distilabel: An ai feedback (aif) framework for building datasets with and for llms. https://github.com/argilla-io/distilabel.

Daoyuan Chen, Yilun Huang, Zhijian Ma, Hesen Chen, Xuchen Pan, Ce Ge, Dawei Gao, Yuexiang Xie, Zhaoyang Liu, Jinyang Gao, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024a. Data-juicer: A one-stop data processing system for large language models. In *International Conference on Management of Data*.

Daoyuan Chen, Yilun Huang, Xuchen Pan, Nana Jiang, Haibin Wang, Ce Ge, Yushuo Chen, Wenhao Zhang, Zhijian Ma, Yilei Zhang, Jun Huang, Wei Lin, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024b. Data-juicer 2.0: Cloud-scale adaptive data processing for foundation models. *arXiv preprint arXiv:2501.14755*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Zihong Chen, Wanli Jiang, Jinzhe Li, Zhonghang Yuan, Huanjun Kong, Wanli Ouyang, and Nanqing Dong. 2025. Graphgen: Enhancing supervised fine-tuning for llms with knowledge-driven synthetic data generation. *arXiv preprint arXiv:2505.20416*.

Chesterfield Laboratories Inc. 2025. Kiln: Rapid ai prototyping and dataset collaboration tool. https://github.com/Kiln-AI/Kiln.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xintong Hao, Ruijie Zhu, Ge Zhang, Ke Shen, and Chenggang Li. 2025. Reformulation for pretraining data augmentation. *arXiv preprint arXiv:2502.04235*.

Conghui He, Wei Li, Zhenjiang Jin, Chao Xu, Bin Wang, and Dahua Lin. 2024. Opendatalab: Empowering general artificial intelligence with open datasets. *arXiv preprint arXiv:2407.13773*.

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*.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2024. CMMLU: Measuring massive multitask language understanding in Chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11260–11285, Bangkok, Thailand. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Ryan Marten, Trung Vu, Charlie Cheng-Jie Ji, Kartik Sharma, Shreyas Pimpalgaonkar, Alex Dimakis, and Maheswaran Sathiamoorthy. 2025. Curator: A tool for synthetic data creation. https://github.com/bespokelabsai/curator.

Ollama. 2023. Ollama: Run large language models locally. https://github.com/ollama/ollama.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. *arXiv preprint arXiv:2402.13446*.

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.

Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, and 1 others. 2024a. Mineru: An open-source solution for precise document content extraction. *arXiv preprint arXiv:2409.18839*.

Ke Wang, Jiahui Zhu, Minjie Ren, Zeming Liu, Shiwei Li, Zongye Zhang, Chenkai Zhang, Xiaoyu Wu, Qiqi Zhan, Qingjie Liu, and 1 others. 2024b. A survey on data synthesis and augmentation for large language models. *arXiv preprint arXiv:2410.12896*.

Michael Williamson. 2015. Mammoth .docx to html converter. https://github.com/mwilliamson/mammoth.js.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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.

Qintong Zhang, Victor Shea-Jay Huang, Bin Wang, Junyuan Zhang, Zhengren Wang, Hao Liang, Shawn Wang, Matthieu Lin, Conghui He, and Wentao Zhang. 2024. Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction. *arXiv preprint arXiv:2410.21169*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.

Johannes Zillmann. 2017. Pdf-to-markdown converter. https://github.com/jzillmann/pdf-to-markdown.

A LLM-as-a-Judge Prompt

Please act as an impartial evaluator and assess the quality of the AI assistant's response to the user's question. You will be provided with the following information:

1. The original user question (Question)
2. A standard answer containing information directly relevant to the user's question (Ground Truth)
3. The AI assistant's response (Prediction)

Your task is to conduct a thorough evaluation focusing on correctness, scoring from 0 to 5 points.

Evaluation Method:
1. Carefully read the question, the assistant's response, and the ground truth answer.
2. Identify and list all key factual statements from the ground truth.
3. For each fact, determine whether it is correctly reflected in the assistant's response.
4. Assign a final correctness score based on the degree of fact matching. If all facts from the ground truth are correctly reflected in the AI response, assign 5 points. If none are correct, assign 0 points.

Please carefully analyze the correctness of the answer. Finally, provide the score result in JSON format as follows:

```
[
  {
     "correctness": "3"
  }
]
```

Question
`{ Question }`

Prediction
`{ Prediction }`

Ground Truth
`{ Ground Truth }`

B Fine-Tuning Details

All the experiments were conducted on 2 NVIDIA A800 80GB GPUs. For all fine-tuning runs, we used a batch size of 64 and a total of 10,000 training samples. The model was trained for 2 epochs with a learning rate of 10^{-5} using a cosine learning rate schedule. A warmup ratio of 0.1 was applied to stabilize training and mitigate overfitting.

SlackAgents: Scalable Collaboration of AI Agents in Workspaces

Weiran Yao*, **Zhiwei Liu***, **Zuxin Liu**, **Juntao Tan**, **Jianguo Zhang**, **Frank Wang**, **Sukhandeep Nahal**, **Huan Wang**, **Shelby Heinecke**, **Silvio Savarese** and **Caiming Xiong**

Salesforce AI Research

***Equal contributions**

Abstract

The integration of AI agents into organizational workflows remains a significant challenge, limiting their impact on daily business operations despite the rapid progress in agent libraries and large language models. We introduce **SlackAgents**, a scalable **multi-agent** framework that natively embeds AI agents into Slack, the leading enterprise collaboration platform. **SlackAgents** enables seamless agent-to-agent and agent-to-human collaboration, leveraging Slack's flexible messaging infrastructure to orchestrate complex workflows and automate real-world tasks. Our architecture supports modular agent types, distributed orchestration, and intuitive user interfaces, allowing organizations to rapidly deploy, customize, and scale agent teams across diverse use cases. We present several typical real-world deployment cases to demonstrate the effectiveness of **SlackAgents** in bridging the last-mile gap for enterprise AI adoption, unlocking new opportunities for intelligent automation in the workplace.

1 Introduction

AI agents (Chase, 2022; Liu, 2022; Wu et al., 2024; Liu et al., 2024) are autonomous software entities designed to perform specific tasks, make decisions, and interact with both humans and other agents. Leveraging advancements in large language models (LLMs), these agents can now learn, plan and reason, enabling them not only to understand and generate human-like conversations but also to execute actions on our behalf across both real-world and digital environments.

Businesses increasingly adopt AI agents to automate operations, enhance efficiency, and support decision-making. Existing open-source frameworks like LangChain (Chase, 2022) and LlamaIndex (Liu, 2022) offer flexible and integrative capabilities but often fail to transition beyond prototypes due to difficulties integrating them into daily workflows—commonly known as the *last-mile problem* (Wikipedia, 2024). As a result, there is a clear demand for frameworks specifically designed for immediate and practical deployment within workplace settings.

Additionally, managing and coordinating large-scale, multi-agent systems remains challenging, as illustrated by Salesforce's ambitious goal to deploy *one billion agents* by 2025 (Salesforce, 2024). Current multi-agent platforms, including AutoGen (Wu et al., 2024), CAMEL (Li et al., 2023), and CrewAI (Moura, 2024), lack sufficient capabilities for robust communication and collaboration at scale.

To address these limitations, this paper introduces SLACKAGENTS , a scalable multi-agent library integrated directly with Slack, the leading corporate collaboration platform (Slack, 2024). By leveraging Slack's extensive messaging infrastructure, our framework facilitates integrated and scalable collaboration, allowing agents to communicate and orchestrate tasks efficiently through Slack channels and threads. It also enables seamless workspace automation, empowering agents to directly utilize Slack's native features—including canvases, bookmarks, workflows, and notifications—to automate and streamline everyday work tasks. Furthermore, our framework supports immediate deployment and continuous improvement, as agents integrate natively into existing workflows, enabling rapid deployment, iterative enhancements through routine usage, and straightforward distribution across multiple Slack workspaces.

Key features of SLACKAGENTS include:

- **Modular and Scalable Architecture**: Designed for flexibility, each agent operates independently as a Slack application, allowing easy scaling and customization.
- **Efficient Distributed Orchestration**: Utilizing Slack's messaging system, agents dynamically manage communications, enabling seamless col-

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 969–982
November 4-9, 2025

Figure 1: Integrating AI agents with Slack's features and visual components unlocks a variety of applications for workflow automation involving multiple agents: **(1) Collaborative specialist team** for enhancing business workflows and foster decision making from diverse perspectives; **(2) Proactive support specialist** with in-depth knowledge of specific Slack channels (e.g., a scrum team channel) to assist team members proactively, and (3) **Personal assistant** that is deeply familiar with individual users, offering tailored support and solving private tasks efficiently.

laboration and knowledge sharing across agents.

- **User-friendly Interfaces**: SlackAgent offers intuitive interactions through Slack chatting interfaces, interactive dashboards for monitoring and configuring agents, and natural language commands for creating and managing agents without technical expertise.
- **Flexible Agent Types**: Provides customizable agent types—Assistant, Workflow, and Proactive agents—which developers can combine to build tailored solutions from individual productivity assistants to complex multi-agent teams.

2 SLACKAGENTS Architecture

SLACKAGENTS is designed to enable seamless multi-agent collaboration and intelligent automation within Slack workspaces. The architecture introduces a unified agent framework, supporting diverse interaction patterns through modular agent types and extensible tool integration. This section details the fundamental agent classes, communication protocols, and interface components that underpin the system, illustrating how these elements collectively facilitate robust, scalable teamwork in enterprise messaging environments.

2.1 Core Agent Types

The framework defines three primary agent types: the *Assistant Agent*, which performs tasks in multi-turn conversations using tools such as functions, APIs, external libraries, and code interpreters; the *Workflow Agent*, which manages multi-stage workflows by maintaining state and guiding sequential goal completion; and the *Proactive Agent*, which maintains persistent awareness of context and autonomously offers support when needed. Each agent is deployed as a standalone Slack app, equipped with listeners and message-sending capabilities, enabling both user-agent and inter-agent collaboration through Slack channels and threads (see Figure 2). The detailed implementation of agents is in Appendix B.4.

2.2 Scalable Multi-Agent Collaboration

Let us assume that for now we have already built multiple individual AI agents with SLACKAGENTS framework, *agent a, b, and c*, whatever an Assistant or Workflow agent. In this section, we walk you through how users can further enable multi-agent collaboration between them.

SLACKAGENTS defines the collaboration of *agent a,b and c* when initializing each individual agent as follows:

```
from slackagents import SlackAssistant

agent_a = SlackAssistant(
    name=name_agent_a,
    desc=desc_agent_a,
    system_prompt=...,
    tools=[...],
    slack_bot_token=...
    colleagues={
        id_agent_b: {"name": name_agent_b, "description": desc_agent_b},
        id_agent_c: {"name": name_agent_c, "description": desc_agent_c},
    } # define the multi-agent collaboration
)
```

where the *colleagues* key argument defines the possible collaboration to its collaborative agents *a* and

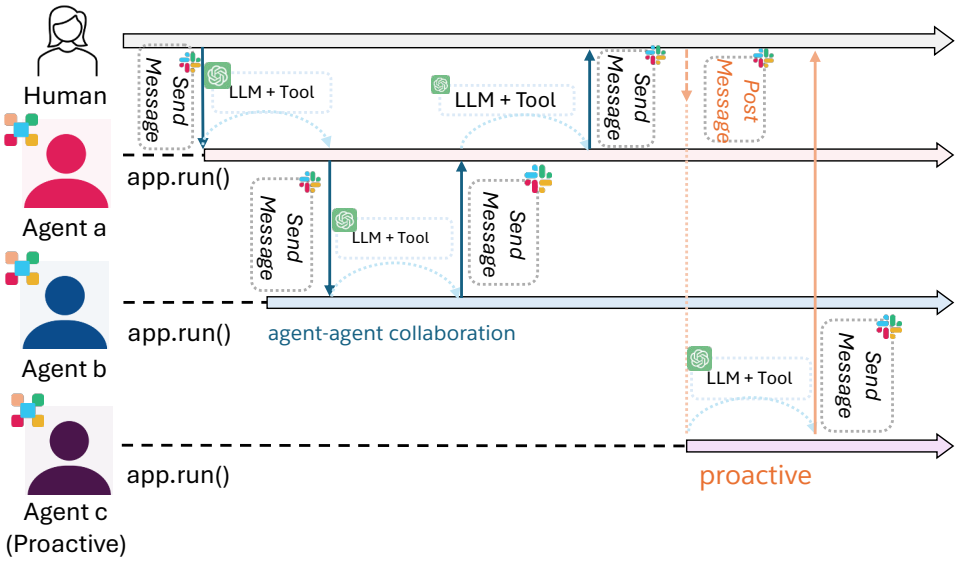


Figure 2: Illustration of how SLACKAGENTS can program a multi-agent conversation in a scalable, decentralized way. The traces of the operation of a multi-agent team, step by step. When *human* sends the first message directly to *agent a*, *agent a* decides to ask the help from *agent b* and respond back to human, which shows the agent-agent collaboration via SLACKAGENTS . When human posts a message under the thread without mentioning a specific agent, a proactive agent 3 listens this message and chimes in to respond back, which illustrates the proactive mode. Proactive agent 3 may also respond towards the message sending from another agent.

b. It is worth noting that we use the Slack User ID as the id of each agent, such as *id_agent_b*.

After we define all agents, SLACKAGENTS runs each agent as a individual Slack App[1]. Built upon the scalable message handling backend of Slack, SLACKAGENTS provides three types of Slack message handlers, which are Direct Message Handler, Channel Message Handler and Proactive Message Handler to process different chatting modes.

Direct Message Handler supports sending direct message to an agent through Slack app direct message interface. One could initialize a agent from SLACKAGENTS and register this agent to listening direct messages. *Channel Message Handler* in SLACKAGENTS supports @ mention in a slack channel. In this way, all agents and humans are able to send message to any specific agent, which enables team collaboration. It also enables the agent to reach out to others for help. Note that we could include either another agent or another human for help. *Proactive Message Handler* provides a proactive way to monitor all messages and step in to provide help only when needed. We show this process as in Figure 2. Once the proactive agent is activated in the backend, it monitors all messages within that thread. The agent determines whether to participate in the discussion based on its capabilities from system prompts and available tools. When the agent identifies a need or request for support, it automatically employs its tools and engages in the conversation. Detailed implementation fo those message hanlders are in Appendix B.1.

2.3 Collaboration Protocol

The SLACKASSISTANT class represents a conversational agent designed for multi-agent collaboration within Slack channel environments. This agent leverages Slack-specific functionalities, making it particularly suitable for team-based interactions.

The core of the multi-agent collaboration in SLACKAGENTS is for the current agent to **(1) produce** a message that contains a request for assistance with @ mention of the chosen agents or human from a pre-defined colleague list, **(2) send** the message to the colleague(s) in a dedicated session, and **(3) listen** for colleagues' responses in the session. Compared with the "handoff" strategy in OpenAI swarm (OpenAI, 2024), which hands off all messages to another agent by swapping system prompt and tools, our collaboration strategy is decentralized, asynchronous, and scalable by leveraging Slack-specific functionalities, and importantly, same as how human workers collaborate in Slack channels by looping in colleagues for help.

We use function calling as the standard protocol

[1]https://api.slack.com/docs/apps

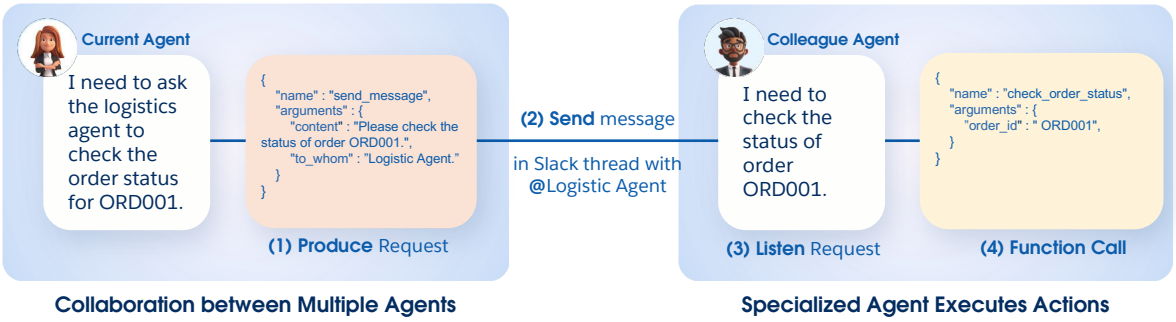


Figure 3: Multi-agent communication protocol. This example illustrates how two agents can collaborate in a dedicated collaboration session in Slack thread.

for producing collaboration requests. Three Slack conversation tools, SEND_MESSAGE, WAIT, and GET_THREAD_HISTORY are added to each SLACKASSISTANT to facilitate multi-agent collaboration in a Slack session. As illustrated in Figure 9, when the current agent in the session decides that the current conversation is out of its capabilities but falls into its team members' roles and capabilities, it will execute SEND_MESSAGE functions, which sends a `"<@to_whom>" + content` message in the thread, where the colleague agent consumes this request through listener and chimes in to help. After sending a request, the agent executes the WAIT function, ending its tool request loop and awaiting responses from colleague agents. The proactive agent behavior is mainly achieved through this function. More details can be found in Sec. B.4.3. All agents are equipped with GET_THREAD_HISTORY by default to obtain the past messages in the thread, in case the request message which has been sent is not informative enough. More detailed implementation are in Appendix B.2.

2.4 User Interface

2.4.1 Command Line Interface (CLI)

The SlackAgents Command-Line Interface (CLI) provides a comprehensive set of commands for managing AI agents within Slack workspaces. This specification details the command structure, available operations, and implementation guidelines.

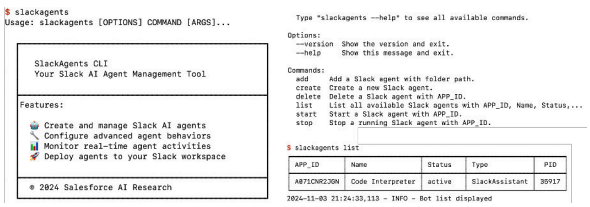


Figure 4: SLACKAGENTS Command Line Interface.

Command Structure The CLI follows a standardized command pattern:

```
slackagents [COMMAND] [OPTIONS]
```

Core Commands

`create` Interactive wizard for new agent creation

`add [FOLDER_PATH]` Register existing agent from specified directory

`list` Display agents with APP_ID, name, status, and type (`-verbose` for details)

`start [APP_ID]` Launch specified agent

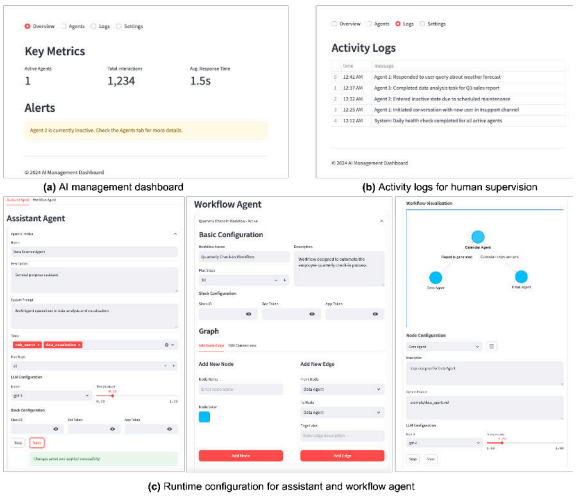
`stop [APP_ID]` Terminate specified agent

`delete [APP_ID]` Remove agent and associated resources

To effectively use the CLI, begin operations with `slackagents list` to view available agents. Access detailed documentation for any command using `-help`. Keep track of APP_ID records for agent management operations, noting that all commands are case-sensitive and perform input validation before execution.

2.4.2 Admin Dashboard

The admin dashboard serves as a central hub for managing AI agents, tools, and workflows. It provides real-time visibility into agent performance, task execution, and system status. Key features include user management, tool configuration, and detailed analytics on agent interactions. Admins can monitor agent activity, adjust tool settings, and troubleshoot issues directly from the dashboard.



(a) AI management dashboard (b) Activity logs for human supervision

(c) Runtime configuration for assistant and workflow agent

Figure 5: AI Mangement Dashboard.

Configuration of AI agents at runtime lets the admin to modify a Slack agent's behaviors from the Admin Dashboard UI, without needing to restart the application.

3 Applications of SLACKAGENTS

In this section, we present a comprehensive exploration of native applications enabled by SLACKAGENTS to automate our daily work and improve decision-making processes, which aligns the video demoing **proactive agent**, **code interpreter agent**, and **multi-agent customer service team**.

3.1 Proactive Assistant - AgentPro

AgentPro is an intelligent and context-aware Slack assistant designed to enhance workplace collaboration. Unlike traditional chatbots, AgentPro seamlessly integrates into team discussions, participating only when necessary to provide valuable insights or assistance. It continuously monitors conversations, identifying key topics or moments where its expertise can contribute, and steps in either when directly mentioned or when it recognizes an opportunity to add meaningful value. An illustration of how proactive agent monitors the message in slack channel is in Figure 6.

A key strength of AgentPro is its ability to engage proactively without being disruptive. It avoids unnecessary interruptions during casual exchanges, choosing instead to focus on moments where its input is most valuable. This thoughtful engagement helps maintain the flow of natural conversations while ensuring critical questions are addressed.

By transforming the traditional chatbot paradigm into a more natural and dynamic collaborator, AgentPro enhances workplace interactions. Its ability to intelligently balance when to engage and when to stay silent makes it an invaluable tool for streamlining workflows, fostering effective communication, and improving team productivity.

3.2 Code Interpreter Agent

The *Code Interpreter Agent* is deployed as a Slack direct-message (DM) application that couples LLM dialogue with a Python sandbox runtime. The runtime exposes the essential scientific-Python stack—`numpy`, `pandas`, `scikit-learn`, `matplotlib`, and `seaborn` behind a restricted I/O boundary. Whenever a user submits a request—e.g., "*Please explore the IRIS dataset and visualise the class separability*"—the agent enters a five-stage loop: *plan*, *generate code*, *confirm*, *execute*, and *recap*. It first decomposes the goal into minimal steps and presents this plan for approval. Only after the user signs off does the agent emit Markdown-formatted Python code, run it inside a containerized kernel, and surface the results—figures, tables, or metrics—before advancing to the next step. This human-in-the-loop protocol yields an auditable trail of intentions, code, and outputs while preventing unintended or malicious execution.

3.3 Autonomous Customer Service Team

To optimize customer support operations for an e-commerce company, we developed a multi-agent system with SLACKAGENTS comprising three specialized agents: the Customer Service Agent, the Sales Agent, and the Logistics Agent. Each agent plays a distinct role within the system, designed to address specific aspects of customer requests while seamlessly collaborate to provide service.

The Customer Service Agent acts as the primary interface with customers, handling inquiries related to order status, cancellations, modifications, and product recommendations, as shown in Figure 7. This agent ensures that every customer message is acknowledged and processed promptly. For order-related tasks, the agent collaborates with either the Logistics Agent or the Sales Agent, depending on the nature of the request. For example, when a customer inquires about an order status, the *Customer Service Agent* collects the order ID and queries the Logistics Agent to check the order status. Similarly, product requests are relayed to the Sales Agent.

The Logistics Agent oversees order status and shipping logistics. It provides detailed updates on

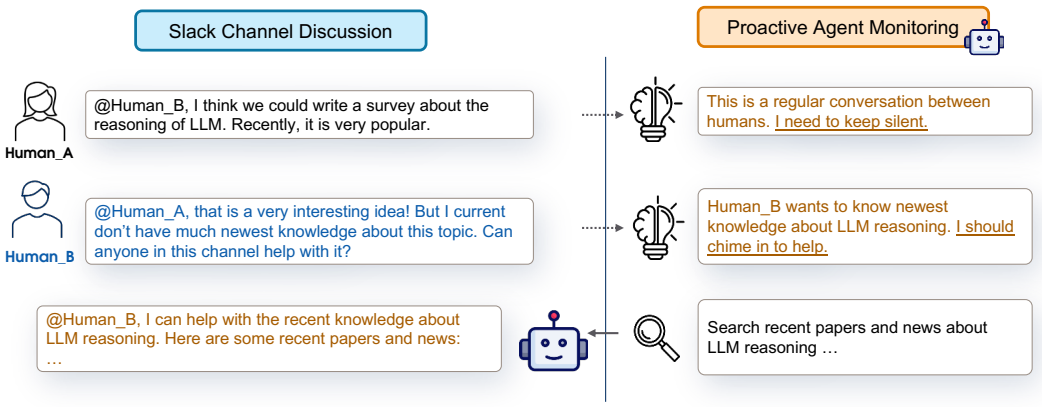


Figure 6: Proactive channel assistant monitors the messages in slack channel and decide to keep silent or chime in to help. Its decision making is conditioned on the compound of tools, prompts and messages.

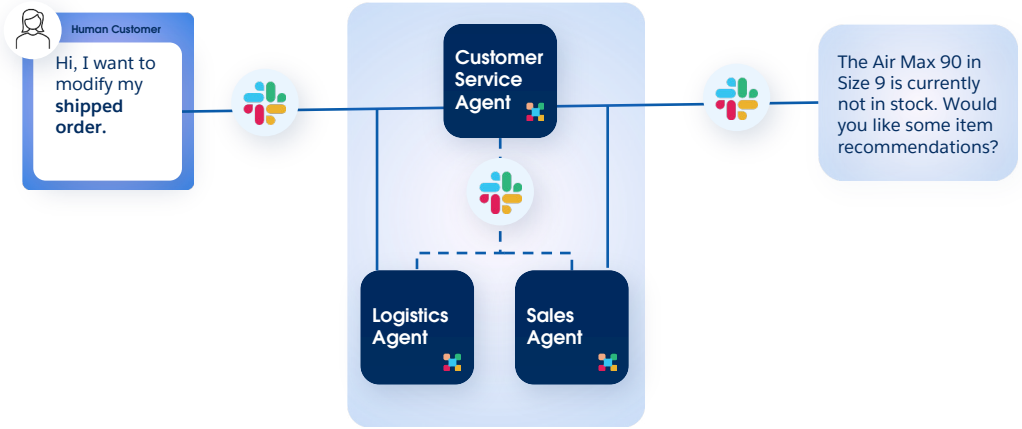


Figure 7: A customer service team for E-commerce. Customer Service Agent coordinates logistics agent and sales agent. Human customer can also directly send message to each agent.

shipping statuses, estimated delivery times, and any potential delays. The Sales Agent specializes in managing product-related interactions, particularly for recommendation. This agent is responsible for inventory checks, product recommendations, and order creation or modification. The strength lies in its collaboration. SLACKAGENTS advances multi-agent application development in its flexible collaboration work styles. Human customer could ask customer service agent to relay the information to other agents. Or a human customer could ask directly to any specific agent for help.

Additional applications are also avaiable in our source code and appendix C.

4 Conclusion

In this paper, we introduced SlackAgents, a robust framework for scalable deployment and collaboration of AI agents within enterprise workspaces. By tightly integrating with Slack, SlackAgents enables seamless orchestration, communication, and automation, which significantly advances in realizing the vision of intelligent, collaborative, and adaptive workplace automation. Our modular architecture supports a variety of agent types and interaction modes, facilitating flexible deployment and efficient teamwork across diverse business scenarios. Through real-world applications such as proactive assistance, code interpretation, and multi-agent customer service, we demonstrate the practical value and transformative potential of AI-powered collaboration in modern organizations.

Limitations

While SLACKAGENTS provides significant advantages for workplace AI agent deployment, several limitations should be acknowledged:

- Platform dependency on Slack infrastructure
- Scalability constraints inherent to the underlying platform
- Learning curve for organizations new to multi-agent systems
- Integration complexity with highly specialized enterprise systems

These limitations represent areas for future improvement and development of the framework.

Acknowledgments

We thank the Salesforce AI Research team for their support and collaboration in developing this framework. We also acknowledge the valuable feedback from early adopters and beta testers who helped refine the platform's capabilities and user experience.

References

Harrison Chase. 2022. LangChain.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Jerry Liu. 2022. LlamaIndex.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Liangwei Yang, Zuxin Liu, Juntao Tan, Prafulla K Choubey, Tian Lan, Jason Wu, Huan Wang, and 1 others. 2024. Agentlite: A lightweight library for building and advancing task-oriented llm agent system. *arXiv preprint arXiv:2402.15538*.

Joao Moura. 2024. CrewAI.

OpenAI. 2024. Swarm: An educational framework exploring ergonomic, lightweight multi-agent orchestration.

Salesforce. 2024. Salesforce unveils agentforce–what ai was meant to be. Accessed on Oct 10, 2024.

Slack. 2024. Announcing agents and ai innovations in slack.

Wikipedia. 2024. Last mile (transportation).

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Eric Zhu, Li Jiang, Shaokun Zhang, Xiaoyun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework.

A Recommended Guidelines for Using SlackAgents Library

Here we introduce the basic on-boarding steps of **SlackAgents**. Firstly, you should create a Slack APP and configure the agent with tokens from an APP. After that, you run an agent with **SlackAgents**and interact with it through Slack direct message. The SlackAgents framework streamlines the process of integrating AI agents into Slack. This section details the steps required to create and configure a Slack app, enabling seamless interaction between the app and your agent.

A.1 Step 1: Create Your Slack App

You have two options to create your Slack app: using the Slack Manifest and `slack_sdk` API (recommended) or manually creating the app.

Option 1: Using the Slack Manifest and `slack_sdk` API We recommend setting up your Slack app via the Slack Manifest for its simplicity and automation. An example manifest JSON file is provided in the `app/your_first_slack_assistant` folder. Start by obtaining your App Configuration Access Token and exporting it as an environment variable:

```
export SLACK_APP_CONFIG_TOKEN=xoxe.xoxp...
```

Next, navigate to the `app/your_first_slack_assistant` folder or copy it and rename it. Update the `agent_config.json` file with your agent's name and description. These values will populate fields in the `example_manifest.json` file. Use the `create_slack_app.py` script to create the app by running:

```
cd app/your_first_slack_assistant
python create_slack_app.py \
--manifest example_manifest.json
```

The created app will appear in the Slack App Dashboard. The script also saves the App ID in the `slack_bolt_id.json` file, which will need to be updated with corresponding tokens.

To configure tokens, refer to Steps 1-6 of *Tokens and Installing Apps*[2]. Set up App-level tokens with `connections:write` scope and Bot tokens in the OAuth & Permissions section. Ensure the *Allow users to send Slash commands and messages from the messages tab* option is enabled in the App Home section. Reinstall the app to your workspace. No additional scope modifications are needed when using the example manifest.

After completing these steps, you should have the following credentials:

- **App ID**: Found in the Basic Information section.
- **SLACK_BOT_TOKEN**: Found in the OAuth & Permissions section (e.g., `xoxb-...`).
- **SLACK_APP_TOKEN**: Found in the OAuth & Permissions section (e.g., `xapp-...`).

Option 2: Manually Create the App Follow the Slack Bolt Python Getting Started Guide to manually set up your app. Complete the steps for *Create an App*, *Tokens and Installing Apps*, and *Setting Up Events*. Ensure the following settings:

- *Allow users to send Slash commands and messages from the messages tab*
- *Always Show My Bot as Online* (in the App Home section).

After setup, obtain the App ID, SLACK_BOT_TOKEN, and SLACK_APP_TOKEN from the respective sections of the app's dashboard. Ensure your app has scopes such as `groups:read`, `channels:history`, `channels:read`, `im:history`, `chat:write`, `users:read`, `im:read`, and `app_mentions:read`.

A.2 Step 2: Configure Your Slack Bolt ID and Enable Direct Messaging

Update the `slack_bolt_id.json` file with your App ID, SLACK_BOT_TOKEN, and SLACK_APP_TOKEN. To enable direct messaging, navigate to the Slack App Dashboard, select the *App Home* tab, and enable the *Allow users to send Slash commands and messages from the messages tab* option. Reinstall the app via the OAuth & Permissions section and refresh or restart your Slack client.

Note: If the message window does not appear immediately, try restarting Slack after a few minutes.

[2] https://tools.slack.dev/bolt-python/getting-started/#tokens-and-installing-apps

A.3 Step 3: Start Your Agent

To launch your direct message agent, execute the following command:

```
python my_agent.py -type dm
-agent-config agent_config.json
-bolt-config slack_bolt_id.json
```

You can choose from `type dm` for direct messaging or `assistant` for channel-based interaction where the agent responds to @ mentions. Refer to the tutorial for additional details.

A.4 Step 4: Interact with Your Agent in Slack

Once your agent is running, search for it under *Apps* or the General Search Window Bar in Slack. Send messages to your app, and your agent will respond accordingly.

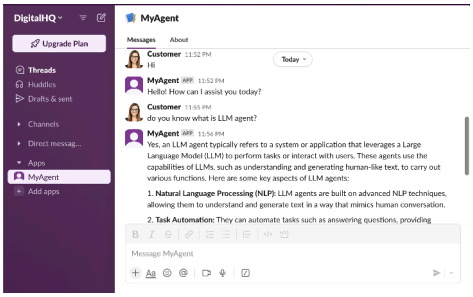


Figure 8: The direct message interface when chatting with **SlackAgents**in Slack.

B Implementation Details

B.1 Message Handlers: Consume messages from subscribed sessions

After we define all agents, SLACKAGENTS runs each agent as a individual Slack App[3]. Built upon the scalable message handling backend of Slack, SLACKAGENTS provides three types of Slack message listeners for humans and all agents to work in a workspace.

Direct Message Handler. SLACKAGENTS supports sending direct message to an agent through Slack app direct message interface. A sample serving code for handling direct message is as follows:

```python
from slack_bolt_id import BOLT_CONFIG
from slackagents import SlackDMHandler
from slackagents import SlackDMAgent

if __name__ == "__main__":
    agent = SlackDMAgent(name=name,desc=desc)
    handler = SlackDMHandler(BOLT_CONFIG, agent)
    handler.run()
```

[3] https://api.slack.com/docs/apps

One could initialize a *SlackDMAgent* from SLACKAGENTS and register this agent to *SlackDMHandler* to listening direct messages.

Channel Message Handler. Besides sending direct messages to those agents, SLACKAGENTS also supports @ mention in a slack channel. In this way, all agents and humans are able to send message to any specific agent, which enables team collaboration. A sample serving code is as follows:

```
from slack_bolt_id import BOLT_CONFIG
from slackagents import SlackChannelHandler
from slackagents import SlackAssistant

if __name__ == "__main__":
    agent = SlackAssistant(name=name, desc=desc, colleagues=colleagues)
    handler = SlackChannelHandler(BOLT_CONFIG, agent)
    handler.run()
```

where one could define an agent with *SlackAssistant* API from SLACKAGENTS and register it into *SlackChannelHandler*. The colleagues are the key argument for this agent to collarabote with. It enables the agent to reach out to others for help. The colleagues could include either another agent or another human for help.

Proactive Message Handler Instead of assigning a message to a agent, SLACKAGENTS also provides a proactive agent to monitor all messages and step in to provide help only when needed. We show this process as in Figure 2 and a sample code is provided as follows:

```
from slack_bolt_id import BOLT_CONFIG
from slackagents import SlackProactiveHandler
from slackagents import SlackAssistant

if __name__ == "__main__":
    agent = SlackAssistant(name=name, desc=desc)
    handler = SlackProactiveHandler(BOLT_CONFIG, agent)
    handler.run()
```

Once the proactive agent is activated in the backend, users can @ mention it in any Slack thread, enabling it to monitor all messages within that thread. The agent determines whether to participate in the discussion based on its capabilities, including system prompts and available tools. When the agent identifies a need for support, it automatically employs its tools and engages in the conversation.

B.1.1 Conversation Tools: Produce Collaboration Request

B.2 Conversation Tools: Produce Collaboration Request

The SLACKASSISTANT class represents a conversational agent designed for multi-agent collaboration within Slack channel environments. This agent leverages Slack-specific functionalities, making it particularly suitable for team-based interactions. The agent maintains awareness of human and agent team members through a colleague system. This feature allows the agent to understand team composition and maintain appropriate context about different team members' roles and capabilities.

Collaboration Strategy The core of the multi-agent collaboration in SLACKAGENTS is for the current agent to **(1) produce** a message that contains a request for assistance with @ mention of the chosen agents or human from a pre-defined colleague list, **(2) send** the message to the colleague(s) in a dedicated session, and **(3) listen** for colleagues' responses in the session. Compared with the "handoff" strategy in OpenAI swarm (OpenAI, 2024), which hands off all messages to another agent by swapping system prompt and tools, our collaboration strategy is decentralized, asynchronous, and scalable by leveraging Slack-specific functionalities, and importantly, same as how human workers collaborate in Slack channels by looping in colleagues for help in threads.

Collaboration Protocol We use function calling as the standard protocol for producing collaboration requests. Three Slack conversation tools, SEND_MESSAGE, WAIT, and GET_THREAD_HISTORY are added to each SLACKASSISTANT to facilitate multi-agent collaboration in a Slack session.

SEND_MESSAGE. As illustrated in Figure 9, when the current agent in the session decides that the current conversation is out of its capabilities but falls into its team members' roles and capabilities, it will execute SEND_MESSAGE functions, which sends a `"<@to_whom>" + content` message in the thread, where the colleague agent consumes this request through listener and chimes in to help.

```
{
    "type": "function",
    "function": {
        "name": "send_message",
        "description": "Send a message
    to one of your colleagues or to the
    message sender.",
        "parameters": {
            "type": "object",
            "properties": {
                "content": {
                    "type": "string",
                    "description": "The
    content of the message to be sent."
                },
                "to_whom": {
                    "type": "string",
```

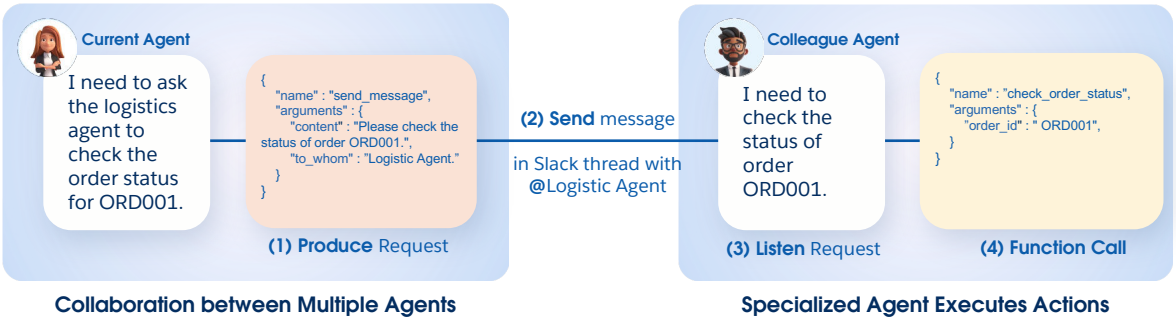


Figure 9: Multi-agent communication protocol. This example illustrates how two agents can collaborate in a dedicated collaboration session in Slack thread.

```
                "description": "The
    name of the recipient."
                }
            },
            "required": ["content", "
    to_whom"],
            "additionalProperties":
    false
        }
    }
}
```

WAIT. After sending a request, the agent executes the WAIT function, ending its tool request loop and awaiting responses from colleague agents. The proactive agent behavior is mainly achieved through this function. More details can be found in Sec. B.4.3.

```
{
    "type": "function",
    "function": {
        "name": "wait",
        "description": "Wait for the
    next message.",
        "parameters": {
            "type": "object",
            "properties": {
                "reason": {
                    "type": "string",
                    "description": "The
    reason for waiting."
                }
            },
            "required": ["reason"],
            "additionalProperties":
    false
        }
    }
}
```

All agents are equipped with GET_THREAD_HISTORY by default to obtain the past messages in the thread, in case the request message which has been sent is not informative enough.

B.3 Collaboration Protocol

We use function calling as the standard protocol for producing collaboration requests. Three Slack conversation tools, SEND_MESSAGE, WAIT, and GET_THREAD_HISTORY are added to each SLACK-ASSISTANT to facilitate multi-agent collaboration in a Slack session. As illustrated in Figure 9, when the current agent in the session decides that the current conversation is out of its capabilities but falls into its team members' roles and capabilities, it will execute SEND_MESSAGE functions, which sends a `"<@to_whom>" + content` message in the thread, where the colleague agent consumes this request through listener and chimes in to help.

```
{
    "type": "function",
    "function": {
        "name": "send_message",
        "description": "Send a message
    to one of your colleagues or to the
    message sender.",
        "parameters": {
            "type": "object",
            "properties": {
                "content": {
                    "type": "string",
                    "description": "The
    content of the message to be sent."
                },
                "to_whom": {
                    "type": "string",
                    "description": "The
    name of the recipient."
                }
            },
            "required": ["content", "
    to_whom"],
            "additionalProperties":
    false
        }
    }
}
```

WAIT. After sending a request, the agent executes the WAIT function, ending its tool request loop and awaiting responses from colleague agents.

The proactive agent behavior is mainly achieved through this function. More details can be found in Sec. B.4.3.

```
{
    "type": "function",
    "function": {
        "name": "wait",
        "description": "Wait for the
    next message.",
        "parameters": {
            "type": "object",
            "properties": {
                "reason": {
                    "type": "string",
                    "description": "The
    reason for waiting."
                }
            },
            "required": ["reason"],
            "additionalProperties":
    false
        }
    }
}
```

All agents are equipped with GET_THREAD_HISTORY by default to obtain the past messages in the thread, in case the request message which has been sent is not informative enough.

B.4 Agent Classes

With communication and collaboration among multiple agents enabled by SLACKAGENTS , the library further provides two key classes for creating custom individual AI agents: **Agent** and **Workflow**. These two classes help define not just the basic behaviors of individual agents but also how an agent act in a cohesive manner to accomplish sequential goals in a principled way.

For tools, SLACKAGENTS supports 4 types of methods to create agent tools, for corresponding use cases. Furthermore, users can also directly load tools from other popular libraries, such as LANGCHAIN, LLAMAINDEX, CREWAI, etc. to develop Slack agents.

B.4.1 Assistant

The SLACKAGENTS library provides a standard way of designing individual agents that can integrate with Slack to perform specific tasks, using various tools and language models. You will need to configure the agent's core attributes, including its name, description, language model, tools, and system prompt. Each of these plays a critical role in defining the agent's behavior.

- NAME: A human-readable name for the agent that will be displayed in Slack.
- DESC: A short description explaining the agent's purpose and functionality.
- LLM: The language model the agent will use for processing natural language input.
- TOOLS: A list of tools that the agent can use to complete tasks with details in Appendix B.4.4.
- SYSTEM PROMPT: The system instruction that sets the context and domain policy for the agent's behavior and interactions with the user.

Here is an example for how an AI agent for brainstorming research ideas and writing paper abstracts is instantiated on Slack. The agent can search on ArXiv for generating new research ideas on a given topic, and writing paper abstract for that idea.

```
from slackagents import AssistantAgent

paper_abstract_agent = AssistantAgent(
    name="Paper Guru",
    desc="Brainstorm abstracts for a given topic",
    llm=OpenAILLM(BaseLLMConfig(model="gpt-4o")),
    tools=[arxiv_tool, abstract_writer_tool],
    system_prompt="You are an AI assistant that can help brainstorm\
    an abstract for a given topic."
)
```

B.4.2 Workflow

A **workflow** is created by organizing individual agents into a structured directed graph, where each agent plays a role in moving toward a common goal. Despite involving multiple agents, the workflow operates as **a single, unified Slack agent**. This design allows users to interact directly with the workflow, which manages the coordination of all agents behind the scenes. From the user's perspective, the workflow appears as a single agent, or in other words, a **workflow agent** that follows a clear, step-by-step process, offering a smooth and reliable experience.

Graph A graph defines the sequence of execution, detailing the interactions and transitions between different agents. Each node in the graph represents an agent, and the edges represent the transitions that dictate when and how agents pass information or trigger each other's actions.

Here's an example where several agents work together for a quarterly check-in process on Slack:

Step 1. Define individual agents.

```
from slackagents import AssistantAgent

data_agent = AssistantAgent(
    name="Data Agent",
    desc="""AI agent designed to generate a report for the quarterly
    check-in meeting with Jira record.""",
    tools = [
        FunctionTool.from_function(load_jira_record_tool),
        FunctionTool.from_function(write_tool),
    ],
```

```
    system_prompt=system_prompt,
    verbose=True
)

calendar_agent = AssistantAgent(
    name="Calendar Agent",
    desc="AI agent designed to load an employee's calendar and\
    send the calendar invites",
    tools=[
        FunctionTool.from_function(load_employee_calendar_tool),
        FunctionTool.from_function(send_calendar_invite_tool)
    ],
    system_prompt=system_prompt,
    verbose=True
)

email_agent = AssistantAgent(
    name="Email Agent",
    desc="AI agent designed to send emails to employees",
    tools=[FunctionTool.from_function(send_email_tool)],
    system_prompt=system_prompt,
    verbose=True
)
```

Step 2. Build the execution graph that defines the workflow.

```
from slackagents import ExecutionGraph, ExecutionTransition

graph = ExecutionGraph()
graph.add_agent(data_agent)
graph.add_agent(calendar_agent)
graph.add_agent(email_agent)
```

Step 3. Define transitions between agents.

```
graph.add_transition(
    ExecutionTransition(
        source_module=graph.get_module("Data Agent"),
        target_module=graph.get_module("Calendar Agent"),
        desc="After the report is written to the employee's local directory"
    )
)

graph.add_transition(
    ExecutionTransition(
        source_module=graph.get_module("Data Agent"),
        target_module=graph.get_module("Email Agent"),
        desc="After the meeting is scheduled."
    )
)
```

Step 4. Set the initial agent in the workflow.

```
graph.set_initial_module(graph.get_module("Data Agent"))
```

Workflow The workflow is a higher-level construct that ties everything together. It encapsulates the agents and the execution graph into a single, reusable, and scalable process. Once the workflow is defined, it can be triggered in Slack to automate a series of tasks.

```
from slackagents import WorkflowAgent
quaterlycheckin_agent = WorkflowAgent(
    name="Quarterly Check-in Workflow",
    desc="Workflow to automate quarterly check-in process",
    graph=graph
)
```

This workflow agent ensures that the check-in process is automated on Slack, starting with generating the report from Jira data dump, scheduling the meeting, and finally sending email notifications, all without human intervention.

B.4.3 Proactive Behavior

Proactive behavior refers to the agent's ability to monitor conversations and provide timely assistance without being intrusive. This behavior is primarily achieved through the WAIT function, as described in Sec. B.2. It enables the proactive message handler to continuously monitor all incoming messages in threads where the agent is active. Based on the context and its capabilities, the agent determines whether to engage in the discussion or continue waiting for additional input.

This process is guided by a structured system prompt, which defines interaction rules, response guidelines, and criteria for silence or engagement. These directives ensure the agent maintains a balance between helpfulness and unobtrusiveness, leveraging the WAIT function to respond appropriately and effectively.

B.4.4 Tools

Tools represent the lowest-level actions that AI agents can perform. Coupled with function calling, tools enable AI models to automatically interface with and control external systems and applications. SLACKAGENTS supports 4 types of methods to define agent tools, for corresponding use cases. Furthermore, users can also directly use tools in other popular libraries, such as LANGCHAIN, LLAMAINDEX, CREWAI, and COMPOSIO, etc. to develop Slack agents.

Function Tool Users can define an arbitrary function and then use FUNCTIONTOOL.FROM_FUNCTION to generate a tool that agents can use automatically. Note that users are highly recommended to use type hints, together with standard Python docstrings, to provide information about the function's input and output. We currently support parsing of ReST, Google, Numpydoc-style and Epydoc docstrings. Automatic error handling is added for all functions wrapped inside FUNCTIONTOOL.

```
from slackagents.tools.function_tool import FunctionTool

def calculate_area(length: float, width: float) -> float:
    """
    Calculate the area of a rectangle.

    :param length: The length of the rectangle.
    :param width: The width of the rectangle.
    :return: The area of the rectangle.
    """
    return length * width

tool = FunctionTool.from_function(calculate_area)
```

Model Tool Besides putting everything inside the function docstrings, one can also explicitly define

an agent tool from a Pydantic data model. Pydantic is a data validation library that uses Python type annotations for data validation and settings management. We can a Pydantic model for our function, specifying its input and output types. This approach offers several benefits:

- Ensures strictly that the tool JSON file sent to LLMs matches our expectations
- Provides better visualization of the tool definition
- Enables automatic input validation

```python
from slackagents.tools.function_tool import FunctionTool
from pydantic import BaseModel, Field

class CalculateArea(BaseModel):
    length: float = Field(..., description="Length of the rectangle")
    width: float = Field(..., description="Width of the rectangle")

    @classmethod
    def execute(cls, length: float, width: float):
        return length * width

tool = FunctionTool.from_pydantic(
    model=CalculateArea,
    name="calculate_area",
    description="Calculate the area of a rectangle"
```

By using a Pydantic model, we can create a strict and type-safe function tool compared to the previous approach of wrapping a Python function directly.

RESTful API Tool AI agents can also use OpenAPI JSON files to define RESTful API tools. Most digital tools today are available as RESTful APIs with standard request formats like GET or POST. While functions or Pydantic classes can be written for these requests, leveraging OpenAPI JSON files is more efficient for defining tools directly. This allows an agent to access a batch of tools simply by referencing a folder of JSON files. The tools also support various types of API authorization, including API keys (in headers or parameters), bearer tokens, and basic authentication with username and password, ensuring secure credentials handling.

```python
from slackagents.tools.function_tool import FunctionTool
tool_schema = ... # load a openapi json file

tool = OpenAPITool(
    name="api_name",
    openapi_spec=tool_schema,
    auth_type=AuthType.NO_AUTH
)
```

External Tool The community has developed an extensive number of AI agent tools. For instance, Composio.ai[4] has aggregated thousands of tools from over 150 popular systems, including GitHub, Twitter, etc. Likewise, LlamaIndex's [5] curates tools from open-source communities. As a result, the quickest way to jumpstart agent development is by using these ready-made tools. SLACKAGENTS supports this with wrappers that directly enable usage of tools from external libraries like, LLamaHub, LangChain[6], CrewAI[7] and Composio.

For rapid development, users can simply import the appropriate external tools from the pool of 1,000+ public, open-source tools. Detailed examples are given in the examples notebooks.

C Additional Applications

C.1 Workflow Agent on Slack

Figure 10: Quarterly check-in workflow. Employee initiates the check-in workflow with data agent, then schedule the check-in time slots with calendar agent, and finally send out the notification through email agent.

Autonomous Quarterly Check-In. The Quarterly Check-In Workflow demonstrates the effectiveness of SLACKAGENTS in automating structured organizational tasks, specifically the quarterly check-in process of employees. Designed as a Slack-integrated system, the workflow leverages a collaborative network of agents to perform tasks such as generating performance reports, scheduling meetings, and sending notifications. By modularizing the workflow into specialized agents—namely, the *Data Agent*, *Calendar Agent*, and *Email Agent*, this approach showcases the benefits of task delegation and automated coordination in multi-agent systems.

At the core of the workflow is the Data Agent, which initiates the process by synthesizing data from Jira records to create a detailed report on employee progress and challenges. This report is further enriched through direct interactions with employees, ensuring the inclusion of qualitative insights alongside quantitative metrics. The agent generates a markdown file summarizing key performance indicators, areas for improvement, and

[4] https://composio.dev/

[5] https://llamahub.ai/

[6] https://python.langchain.com/docs/integrations/tools/

[7] https://github.com/crewAIInc/crewAI-tools

future goals, enabling employees and their managers to prepare effectively for the check-in discussion. Once the report is finalized, the workflow seamlessly transitions to the Calendar Agent.

The Calendar Agent exemplifies the integration of intelligent scheduling with decision-making heuristics. By analyzing multiple employee calendars, it identifies optimal meeting slots based on criteria such as business hours, workload distribution, and personal preferences. This agent minimizes conflicts while prioritizing convenience, offering flexible time slots for employees to choose from when necessary. Once the scheduling task is complete, the workflow moves to the Email Agent, which ensures timely communication of the check-in details.

The Email Agent serves as the final component of the workflow, responsible for sending personalized notifications to employees. These emails summarize the scheduled meeting information and provide access to the prepared performance reports. The agent's role highlights how multi-agent systems can enhance the human-centric aspects of organizational processes by ensuring clarity, timeliness, and personalization.

C.2 Multi-Agent Reasoning on Slack

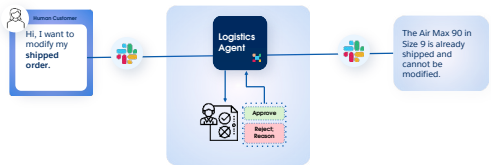


Figure 11: Trustworthy verification of Slack Agent. Logistics Agent asks an verifier for approval before actually executing a tool call.

Trustworthy Verifier. We design a human-in-the-loop slack agent to enhance the trustworthy when agent is going to use a tool, which is illustrated in Figure 11. In this application, we design a logistics agent for checking or modifying the status of an order. Besides tool call generation and execution, the agent execution involves another inner loop for tool call verification. A verifier, which can either be a *human* or another *agent*, approves or rejects with a reason a tool call request from the logistics agent. After receiving the approval or rejection response, logistics agent executes the tool-call or generates a new message, respectively. This trustworthy verification application demonstrates the extendability of SLACKAGENTS framework. Developers could easily integrate another agent or human into the agent execution loop such that the actions of an agent are monitored and verified.

D External Dependencies

The library is extremely lightweight, which only requires the following Python packages:

- `docstring-parser`
- `openai`
- `networkx`
- `matplotlib`
- `slack-sdk`
- `slack-bolt`

Open Political Corpora: Structuring, Searching, and Analyzing Political Text Collections with PoliCorp

Nina Smirnova and **Muhammad Ahsan Shahid** and **Philipp Mayr**
GESIS – Leibniz Institute for the Social Sciences
Unter Sachsenhausen 6-8, 50667 Cologne
{nina.smirnova, ahsan.shahid, philipp.mayr}@gesis.org

Abstract

In this work, we present PoliCorp, a web portal designed to facilitate the search and analysis of political text corpora. PoliCorp provides researchers with access to rich textual data, enabling in-depth analysis of parliamentary discourse over time. The platform currently features a collection of transcripts from debates in the German parliament, spanning 76 years of proceedings. With the advanced search functionality, researchers can apply logical operations to combine or exclude search criteria, making it easier to filter through vast amounts of parliamentary debate data. The search can be customised by combining multiple fields and applying logical operators to uncover complex patterns and insights within the data. Additional data processing steps were performed to enable web-based search and incorporate extra features. A key feature that differentiates PoliCorp is its intuitive web-based interface that enables users to query processed political texts without requiring programming skills. The user-friendly platform allows for the creation of custom subcorpora via search parameters, which can be freely downloaded in JSON format for further analysis.

1 Introduction

Parliamentary debates offer a broad variety of topics for academic exploration, serving as an effective instrument for agenda-setting and influencing political power. By examining parliamentary speeches, researchers can uncover the implicit programmatic and ideological positions political parties hold. Recent studies include investigations into gender dynamics and examinations of negativity levels inferred from interjections (Ash et al., 2025) and sentiment and negativity analysis (Jenny et al., 2021). Additional efforts focus on developing automated topic modeling (Watanabe and Zhou, 2022) and discursive framing approaches within legislative settings (Reinig et al., 2024).

Parliamentary data from many countries is openly available online through governmental initiatives and open data platforms. Examples include the Open Data portal of the German parliament (Bundestag)[1], the Austrian parliament transcripts[2], and Hansard, the official record of the UK Parliament[3]. In addition to raw data, some initiatives provide preprocessed and linguistically annotated corpora, e.g., the Polish Parliamentary Corpus[4], GermaParl, a linguistically annotated corpus of German Bundestag plenary debates (Blaette, 2021, 2017), and DutchParl, which contains documents from the parliaments of the Netherlands, Flanders, and Belgium in Dutch (Marx and Schuth, 2010). Several platforms offer web-based search interfaces for parliamentary records, such as the polit-X portal[5], which includes transcripts from the German Bundestag and state parliaments; the StateParl portal[6], which covers debates from the 16 German state parliaments; and the Italian parliamentary transcripts portal[7].

Although these resources provide comprehensive collections, they often present certain limitations. Raw data typically requires significant preprocessing before it can be analyzed, and many annotated datasets are distributed as complete corpora, necessitating advanced analytical skills to extract specific information (Marx and Schuth, 2010; Blaette, 2017). Furthermore, some data sets are embedded within specialized software environments, which require a knowledge of particular programming languages for effective use (Blaette, 2021, 2020). Although certain platforms, such as polit-X, offer user-friendly online interfaces for data explo-

[1] https://www.bundestag.de/services/opendata
[2] https://www.data.gv.at/en/
[3] https://hansard.parliament.uk/
[4] https://clip.ipipan.waw.pl/PPC
[5] https://polit-x.de/
[6] https://stateparl.de/
[7] https://accademiadellacrusca.it/it/contenuti/discorsi-parlamentari/23591

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 983–992

ration, they often operate on a subscription basis, limiting open access. Overall, the lack of standardized, machine-readable annotations for parliamentary speeches limits the potential for quantitative research in this area (Wissik, 2021).

To overcome these challenges, we introduce the Pollux Political Corpora (PoliCorp) platform[8], an advanced resource that offers researchers open, structured, and searchable access to processed political corpora. The platform currently contains a collection of transcripts of Bundestag debates, spanning 76 years of parliamentary debates – from September 1949 to July 2025. The platform is designed for political scientists, interdisciplinary researchers, and others engaged in the analysis of parliamentary discourse.

Additional data processing steps were performed to enable web-based search and incorporate supplementary features. For instance, PoliCorp allows users to perform targeted searches not only within speeches but also across "interjections" and "calls to order". Calls to order can serve as indicators for analyzing incivility in parliamentary discourse, and offer a unique perspective on political polarization (Jenny et al., 2021), and are therefore of particular interest for political research. Moreover, analysis of calls to order as markers of disruptive language is a novel approach to studies of parliamentary corpora, going beyond traditional sentiment or stance analysis. Interjections constitute a key resource for understanding democratic processes, uncovering hidden power dynamics, and examining conflicts within parliament (Ilie, 2015; Truan, 2017).

A key feature that differentiates PoliCorp is its intuitive web-based interface that enables users to query processed political texts without requiring programming skills[9]. The user-friendly platform allows the creation of custom subcorpora via search parameters, which can be freely downloaded in JSON format for further analysis. The portal's content is distributed under the CLARIN PUB+BY+NC+SA license.

Background on Parliamentary Discourse

The legislative period is a specific period of time for which a parliament is elected. In Germany, it spans four years. Within each legislative period, the work of the German Bundestag is organized through plenary sessions, which follow a set procedure. The session is presided over by a member of the Bundestag Presidium, comprising the president and two secretaries. The president moderates the session, ensuring procedural compliance and managing the allocation of speaking time. Speaking time is distributed proportionally among parliamentary groups, reflecting their relative representation in the parliament. Each session usually comprises several agendas, dedicated to a specific topic.

If the speaker or an attending parliamentary member violates parliamentary order, i.e., interruptions, the president is authorized to issue a formal call to order[10]. Calls to order were analysed by (Jenny et al., 2021) from a perspective of negativity analysis. Figure 3 demonstrates an example of a call to order during a parliamentary session from the PoliCorp interface.

Interruptions or interjections during a speech may include spontaneous remarks or attempts to insert commentary by other members of parliament[11]. Interjections are widely examined in political science research. Research on interjections has largely addressed their pragmatic, rhetorical, and disruptive functions (Truan, 2017; Shenhav, 2008). Many scholarly works have further investigated interjections in relation to gendered dynamics of interjections, including gendered patterns of attacks, interruptions, and participation in legislative debates across different contexts (van Dijk and Poljak, 2025; Ash et al., 2025; Vallejo Vera and Gómez Vidal, 2022; Och, 2020; Poljak, 2022; Vallejo Vera and Gómez Vidal, 2022; Miller and Sutherland, 2023). Additional lines of inquiry have explored interjections in relation to expertise, seniority, and affiliation dynamics (Diener, 2025), cross-cultural variation in speech styles (Isosävi et al., 2025), and broader patterns of interruption behavior (Poljak, 2023).

2 Backend Implementation

Raw parliamentary speeches up to September 7, 2021, were sourced from the GermaParl corpus (Blaette, 2017), a comprehensive linguistic dataset curated by the PolMine project[12]. GermaParl covers transcripts of parliamentary debates from September 7, 1949, to September 7, 2021, and comprises of 958,100 speech contributions. Raw par-

[8] https://demo-pollux.gesis.org/

[9] A demonstration of the search functionality is available at the following link: https://youtu.be/KplgIZRVVwQ

[10] https://www.bundestag.de/services/glossar/glossar/O/ordnungsruf-869614

[11] https://de.wiktionary.org/wiki/Zwischenruf

[12] https://polmine.github.io/

liamentary speeches published after September 7, 2021, were sourced from the Bundestag Open Data project[13]. The raw GermaParl data is available for download in a GitHub repository[14]. The records from the Bundestag Open Data project were retrieved using an API.

The raw data underwent a series of processing stages, as illustrated in Figure 1. The full processed dataset, currently hosted by PoliCorp, comprises 1,035,744 speech contributions[15]. In the final stage, the processed data is indexed using Elasticsearch (version 8.12.1), facilitating efficient retrieval of user-specific information via web-interface.

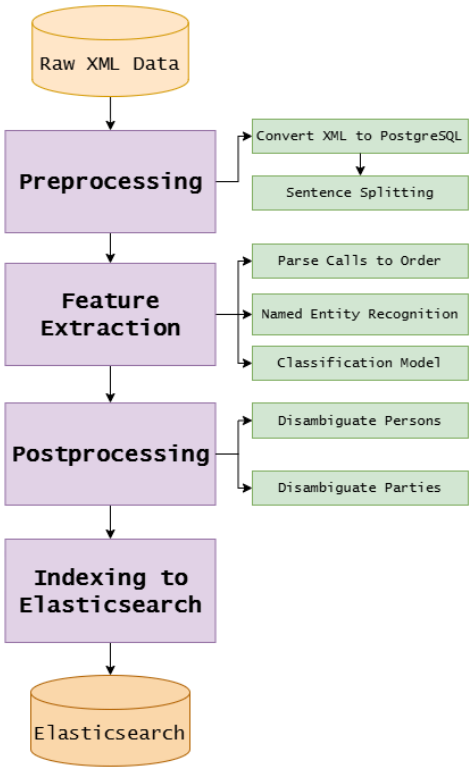


Figure 1: Data processing workflow for PoliCorp

2.1 Preprocessing

The Bundestag and GermaParl data were originally available in XML format, although with different structural schemas (see Appendix A). To address this issue, we developed two distinct conversion pipelines to map each XML structure into a unified database schema. The unified data schema comprises information about the agenda, speaker, and a corresponding speech contribution, as Figure 8 demonstrates. After the conversion process, the textual content of individual speech contributions was segmented into sentences.

2.2 Feature Extraction

Bundestag specifically marks interjections in their raw data (as Figures 11 and 12 demonstrate). Conversely, calls to order are not explicitly indicated in the data. Our objective was also to identify sentences that include calls to order. This procedure is regulated and involves the use of specific phrases. In the first step, we manually reviewed a part of the dataset containing only the speeches of the session's president. Based on this review, we developed a set of rules to identify calls to order, as illustrated in Figure 2. Following, we applied these rules to analyze only the speeches given by the session's president to detect instances of calls to order. Figure 3 shows an example of a marked call to order in the PoliCorp interface[16].

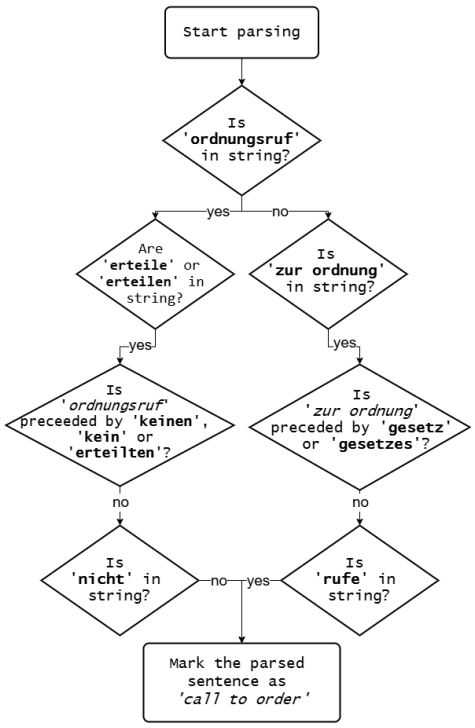


Figure 2: Rules to match calls to order

The output of the Named Entity Recognition (NER) models is integrated into PoliCorp as an experimental feature. Currently, users can see the output of two NER models, Legal German NER[17]

[13] https://www.bundestag.de/services/opendata

[14] https://github.com/PolMine/GermaParlTEI/tree/main

[15] As of September 2025.

[16] Translation of the text in Figure 3 is provided in Appendix B

[17] https://huggingface.co/flair/ner-german-legal

(Leitner et al., 2019; Akbik et al., 2018) and German NER[18] (Akbik et al., 2018). We conducted additional processing of the models' output and retrieved mentions of the German parties using pattern matching. Figure 3 shows the example of the output of the German NER model in the PoliCorp interface.

In addition, each speech contribution is annotated with a topic classification, generated using a BERT-based model[19] (Klamm et al., 2022). The classifier was applied to the full text of each speech contribution, excluding those delivered by the session president, which merely consist of procedural moderation. The underlying model was trained to differentiate between 21 distinct topics relevant to German parliamentary discourse. Contributions from the session president were instead assigned a separate category, labeled Presidency Action, thereby introducing an additional topic class. Figure 4 shows the distribution of topics in the PoliCorp collection over 21 legislative periods. The diagram illustrates the longitudinal development of the discussed topics within the Bundestag. Across all legislative periods, presidency actions and governmental affairs emerge as the most consistently debated and stable topics. In contrast, the environment topic, while being of comparatively low relevance during the early legislative periods, progressively gained relevance, becoming one of the major topics in the late periods.

2.3 Postprocessing

As the final processing step, the names of speakers and their associated political parties were disambiguated. This step was necessary due to inconsistencies in the raw data, which included spelling errors in speaker names, variations of the first names, as well as multiple variations of political party names, i.e., both abbreviated and full forms.

For the disambiguation of speaker names, a rule-based approach was employed. This approach utilized a database containing the names of all members of the German Parliament throughout its history[20]. Individuals with unique surname or surname–first name combinations were directly assigned the corresponding identifier. In cases where identical surname or surname–first name combinations appeared multiple times in the database, additional disambiguation was necessary. This was addressed by aligning the individuals with the legislative periods during which the corresponding speech contribution was delivered. If the date of a speech fell within an individual's documented parliamentary tenure, that person was considered a match. When multiple potential matches remained or no corresponding surname–first name combination could be identified in the database, the entry was kept as ambiguous, and no identifier was assigned.

Subsequently, political party names were disambiguated using pattern matching in conjunction with a comprehensive list of all German political parties and their known abbreviations across the history of the Bundestag. This process allowed for the normalization of party references into a unified format. For instance, both Freie Demokratische Partei and FDP were identified as referring to the same political entity.

3 Frontend Implementation

The frontend is developed using Express.js with Pug templates for server-side rendering. It provides an interactive web-based interface for querying, visualizing, and downloading results from parliamentary corpora. Users can perform both simple and advanced queries, with input parameters dynamically translated into Elasticsearch queries targeting the backend index. The interface includes additional informational pages (e.g., About, GitHub, FAQs, tutorial videos) to support usability. The design emphasizes responsiveness, modularity, and seamless backend integration to enable efficient and effective information retrieval for political science research.

3.1 User Interface

Figure 5 represents a basic search bar. By default, the query is executed against the full text of a speech contribution and all indexed metadata fields. Upon submission, the system returns a results list, displaying the total number of matches and an expandable metadata panel, as Figure 6 shows. Results may be reordered by relevance or chronological order.

For a more specific search, the user can proceed by selecting an advanced search option, as Figure 7 shows. The advanced interface has a set of structured input rows, each row comprising three elements: a logical operator, a field selector, and a

[18] https://huggingface.co/flair/ner-german

[19] https://huggingface.co/chkla/parlbert-topic-german

[20] https://www.bundestag.de/services/opendata

CALL TO ORDER: Herr Kollege Baumann[PER] für den Zwischenruf „Sie Hetzer!“ rufe ich Sie zur Ordnung.

INTERJECTION: (Zuruf von der AfD[PARTY] Also, hier können Leute „Nazi[ORG] sagen!)

SPEECH: – Einen Moment. Vorsicht! Die Ordnungsrufe des Präsidenten werden nicht kommentiert.

INTERJECTION: (Britta Haßelmann[PER] [BÜNDNIS 90/DIE GRÜNEN[PARTY], an die AfD[PARTY] gewandt: Herr Hilse[PER] das gilt auch für Sie! – Gegenruf des Abg. Tino Chrupalla[PER] [AfD[PARTY]: Aber auch für Sie! – Weitere Zurufe von der AfD[PARTY]

CALL TO ORDER: Dafür rufe ich Sie zur Ordnung, Herr Kollege.

INTERJECTION: (Stephan Brandner[PER] [AfD[PARTY]: Unglaublich, was hier abgeht! Unglaublich! – Zuruf: Ja, in der Tat!)

Figure 3: Example of marked call to order and NER output from the PoliCorp interface in the speech of the session's president, Wolfgang Schäuble, on October 2, 2020.

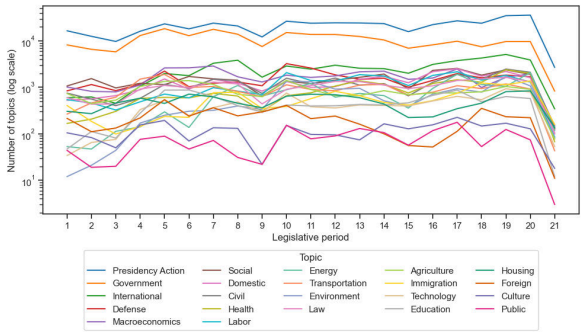


Figure 4: Distribution of topics over legislative periods.

Figure 5: Basic search

value field. Logical operator fields may be configured individually as AND, OR, or NOT, permitting execution of complex Boolean expressions. The drop-down list with the field selectors contains all searchable attributes, including but not limited to: full text, speaker, party, legislative period, topic, or date.

If a user is interested in a specific speech contribution, an expandable metadata panel can be accessed. Figure 8 displays the complete set of metadata associated with the selected speech, including general session-related information such as the legislative period, agenda and session sequence numbers, the date of the session, as well as the agenda type and a brief description. Additional metadata covers speaker-related details, including the speaker's name, party affiliation, and role or position within the Bundestag. Furthermore, the topic of the speech and a link to the corresponding source file are also provided. The web portal incorporates

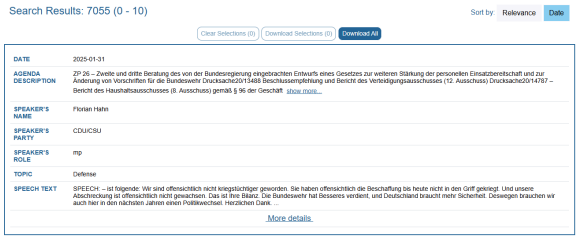


Figure 6: Search results

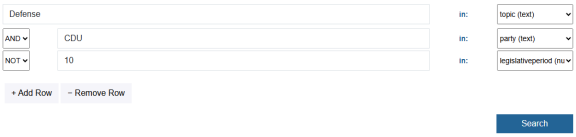


Figure 7: Advanced search

experimental features, such as the outputs of two NER models, as detailed in Section 2.2. These features are accessible via a drop-down menu labeled Speech Text. Selecting a specific NER model from the list dynamically updates the display of the speech contribution, as illustrated in Figure 3, by highlighting the recognized named entities within the text.

LEGISLATIVE PERIOD	20
SESSION NUMBER	211
AGENDA NUMBER	5
DATE	2025-01-31
AGENDA DESCRIPTION	Beratung der Unterrichtung durch die Enquete-Kommission L Drucksache20/14500
SPEAKER'S NAME	Peter Beyer
SPEAKER'S PARTY	CDU/CSU
SPEAKER'S ROLE	mp
URL TO THE SOURCE FILE	https://dserver.bundestag.de/btp/20/20211.xml
TOPIC	Defense
SPEECH TEXT	SPEECH: Frau Präsidentin! Meine sehr geehrten Damen unc Jahren lautete, 20 Jahre deutsches Engagement in Afghanis unserer Arbeit als Enquete-Kommission lag auch auf den Let

Figure 8: Available metadata

A key feature of PoliCorp is that it enables users to combine data in various ways, i.e., generating custom subcorpora and exporting them in JSON

format (Figure 9).

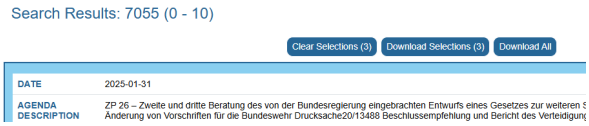


Figure 9: Download options

The data contained in the downloaded JSON file, as shown in Figure 10, fully corresponds to the information presented in the web interface. It includes all associated metadata, sentence-level segmentation of the speech contribution, and available named entity annotations. Thus, metadata comprises details about the speaker, including their identity, affiliation, and functional role. Furthermore, it encompasses information specific to the speech itself, such as the legislative period, session, and agenda numbers, date of delivery, and the speech topic.


```
"date": "2025-05-14",
"description": "(Fortsetzung der Aussprache)",
"speech_id": "4_21003_3_35",
"speechno": 35,
"sessionno": 3,
"agenda_id": "4_21003_3",
"source_file": "https://dserver.bundestag.de/btp/21/21003.xml",
"agenda_type": null,
"agenda_nr": 4,
"party_id": 27,
"speaker_id": 11004709,
"name": "Thomas Erndl",
"position": null,
"party": "CDU/CSU",
"role": "mp",
"topic": "Defense",
"text_raw": [
  {
    "seq_no": 0,
    "legal_ner": [
      {
        "end_pos": 81,
        "text": "Deutschland",
        "label": "LD",
        "start_pos": 70
      }
    ],
    "ner": [
      {
        "end_pos": 81,
        "text": "Deutschland",
        "label": "LOC",
        "start_pos": 70
      }
    ],
    "text": "Meine Damen und Herren, heute, 70 Jahre nach dem NATO-Beitritt, trägt Deuts
    "type": "speech"
  },
  {
    "seq_no": 1,
    "legal_ner": null,
    "ner": null,
    "text": "Lassen Sie uns dem mit einer starken Verteidigungspolitik gerecht werden!",
    "type": "speech"
  },
```


Figure 10: Example of JSON format

4 Portal Usage and Evaluation

Between January 5, 2025, and September 14, 2025, a total of 2,573 user queries were executed within the PoliCorp system, comprising 699 unique queries. The statistical analysis excludes queries exhibiting patterns indicative of potential malicious activity, such as those containing keywords like system, sleep, exec, bash, and similar commands. Table 1 displays the top 10 most frequently searched terms.

Statistics indicate that users typically search for data from a specific legislative period, e.g, 19 and 20. The second most popular search requests include political parties (CDU, SPD, DIE LINKE) and names of politicians (Merkel, Angela Merkel). The other common search requests relate to keywords such as Atomausstieg (nuclear phase-out), Klimawandel (climate change), or Migration (migration).

search term	frequency
19	118
20	93
cdu	89
merkel	77
spd	76
angela merkel	66
atomausstieg	55
klimawandel	52
migration	39
die linke	35

Table 1: Top 10 Most Frequently Searched Terms

The evaluation of the web portal focused on user-centered design principles. Functionality of PoliCorp was developed and refined in collaboration with a small user group with a background in political science or computer science-related disciplines, ensuring that key features align with user needs and expectations. Based on the evaluation results, certain interface and analysis features were added or removed from the portal, and the final interface layout was developed.

5 Conclusion and Future Work

In this work, we present PoliCorp, a web portal designed to facilitate the search and analysis of political text corpora. PoliCorp provides researchers with access to rich textual data, enabling in-depth analysis of parliamentary discourse over time. The platform currently contains a collection of transcripts of Bundestag debates, spanning 76 years of parliamentary debates. With the advanced search functionality, researchers can apply logical operations to combine or exclude search criteria, making it easier to filter through vast amounts of parliamentary debate data. The search can be customised by combining multiple fields and applying logical operators to uncover complex patterns and insights within the data. Selected datasets can be downloaded freely in JSON format, providing a convenient option for further analysis using com-

putational tools.

PoliCorp is a demonstration version that is currently under development. As of the current period, the platform comprises speech contributions from the German Bundestag covering the period from September 1949 to July 2025. Its content will be continuously updated with new speeches as they become available through the Bundestag Open Data portal. Ongoing work includes the implementation of a toxicity detection module, which will assign predefined toxicity labels to individual speech segments. Following, we are planning to link available transcripts with corresponding video recordings.

Future iterations will incorporate additional corpora, such as StateParl, and present them in a unified format to facilitate consistent processing and comparative analysis.

Furthermore, we are working on the integration of the analysis tools, i.e., descriptive statistics and visualization, and cross-corpus content comparison capabilities. Additionally, the available download formats will be expanded. The tool's source code will be made publicly available in a future release.

Limitations

PoliCorp represents a prototype implementation, with many features still under active development. Due to the early stage of the project, the web portal has been evaluated from a user-centered design perspective using a small group of participants. Broader evaluations, including surveys and workshops, are planned for future phases.

The platform incorporates experimental functionalities such as topic classification and NER, both of which are generated automatically and may contain inaccuracies. Users are therefore advised to interpret these outputs with caution and independently verify any critical information.

Furthermore, PoliCorp is a domain-specific resource primarily intended for users engaged in political science research.

Acknowledgements

Nina Smirnova received funding from the Deutsche Forschungsgemeinschaft (DFG) under grant number: MA 3964/7-3 (POLLUX Project).

Nina Smirnova and Philipp Mayr received additional funding from the European Union under the Horizon Europe grant OMINO (Hołyst et al., 2024) – Overcoming Multilevel Information Overload (http://ominoproject.eu) under grant number 101086321.

Muhammad Ahsan Shahid received funding from the Deutsche Forschungsgemeinschaft (DFG) under grant number: MA 3964/20-1 (OFFZIB Project).

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Elliott Ash, Johann Krümmel, and Jonathan B. Slapin. 2025. Gender and reactions to speeches in german parliamentary debates. *American Journal of Political Science*, 69(3):866–880.

Andreas Blaette. 2017. GermaParl. corpus of plenary protocols of the german bundestag.

Andreas Blaette. 2020. polmineR: Verbs and nouns for corpus analysis.

Andreas Blaette. 2021. Germaparl. download and augment the corpus of plenary protocols of the german bundestag. R package version 1.5.3.

Julius Diener. 2025. Explaining interruption behavior in parliament: The role of topic expertise, career status, and government-opposition dynamics. *Politische Vierteljahresschrift*, 66(2):303–324.

Janusz A. Hołyst, Philipp Mayr, Michael Thelwall, Ingo Frommholz, Shlomo Havlin, Alon Sela, Yoed N. Kenett, Denis Helic, Aljoša Rehar, Sebastijan R. Maček, Przemysław Kazienko, Tomasz Kajdanowicz, Przemysław Biecek, Boleslaw K. Szymanski, and Julian Sienkiewicz. 2024. Protect our environment from information overload. *Nature Human Behaviour*, 8:402–403.

Cornelia Ilie. 2015. Parliamentary discourse. In Karen Tracy, Cornelia Ilie, and Todd Sandel, editors, *The International Encyclopedia of Language and Social Interaction*. John Wiley & Sons, Inc., Malden, MA.

Johanna Isosävi, Heike Baldauf-Quilliatre, Christophe Gagne, and Eero Voutilainen. 2025. Reactions to interruptions in finnish, french and german parliamentary debates. *Journal of Language and Politics*, 24(2):301–327.

Marcelo Jenny, Martin Haselmayer, and Daniel Kapla. 2021. *Measuring incivility in parliamentary debates: validating a sentiment analysis procedure with calls to order in the Austrian Parliament*, pages 56–66.

Christopher Klamm, Ines Rehbein, and Simone Ponzetto. 2022. Frameast: A framework for second-level agenda setting in parliamentary debates through the lense of comparative agenda topics. *ParlaCLARIN III at LREC2022*.

Elena Leitner, Georg Rehm, and Julian Moreno-Schneider. 2019. Fine-grained Named Entity Recognition in Legal Documents. In *Semantic Systems. The Power of AI and Knowledge Graphs. Proceedings of the 15th International Conference (SEMANTiCS 2019)*, pages 272–287.

Maarten Marx and Anne Schuth. 2010. DutchParl. the parliamentary documents in Dutch. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Michael G. Miller and Joseph L. Sutherland. 2023. The effect of gender on interruptions at congressional hearings. *American Political Science Review*, 117(1):103–121.

Malliga Och. 2020. Manterrupting in the german bundestag: Gendered opposition to female members of parliament? *Politics & Gender*, 16(2):388–408.

Željko Poljak. 2022. The role of gender in parliamentary attacks and incivility. *Politics and Governance*, 10(4).

Željko Poljak. 2023. Parties' attack behaviour in parliaments: Who attacks whom and when. *European Journal of Political Research*, 62(3):903–923.

Ines Reinig, Ines Rehbein, and Simone Paolo Ponzetto. 2024. How to do politics with words: Investigating speech acts in parliamentary debates. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8287–8300, Torino, Italia. ELRA and ICCL.

Shaul R. Shenhav. 2008. Showing and telling in parliamentary discourse: the case of repeated interjections to rabin's speeches in the israeli parliament. *Discourse & Society*, 19(2):223–255.

Naomi Truan. 2017. On the pragmatics of interjections in parliamentary interruptions. *Revue de sémantique et pragmatique*, 40(40):125–144.

Sebastián Vallejo Vera and Analía Gómez Vidal. 2022. The politics of interruptions: Gendered disruptions of legislative speeches. *The Journal of Politics*, 84(3):1384–1402.

Rozemarijn E van Dijk and Željko Poljak. 2025. Interrupting the interruptions. how women transform the parliamentary debate. *Parliamentary Affairs*, page gsaf040.

Kohei Watanabe and Yuan Zhou. 2022. Theory-driven analysis of large corpora: Semisupervised topic classification of the un speeches. *Social Science Computer Review*, 40(2):346–366.

Tanja Wissik. 2021. Encoding interruptions in parliamentary data: From applause to interjections and laughter. *Journal of the Text Encoding Initiative*.

A Raw XML Formats

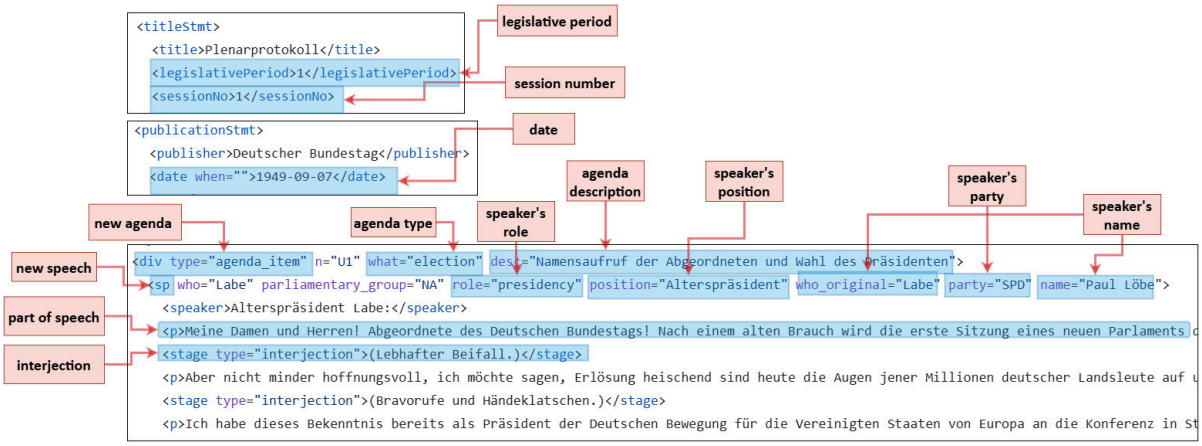


Figure 11: Example of the raw GermaParl data format

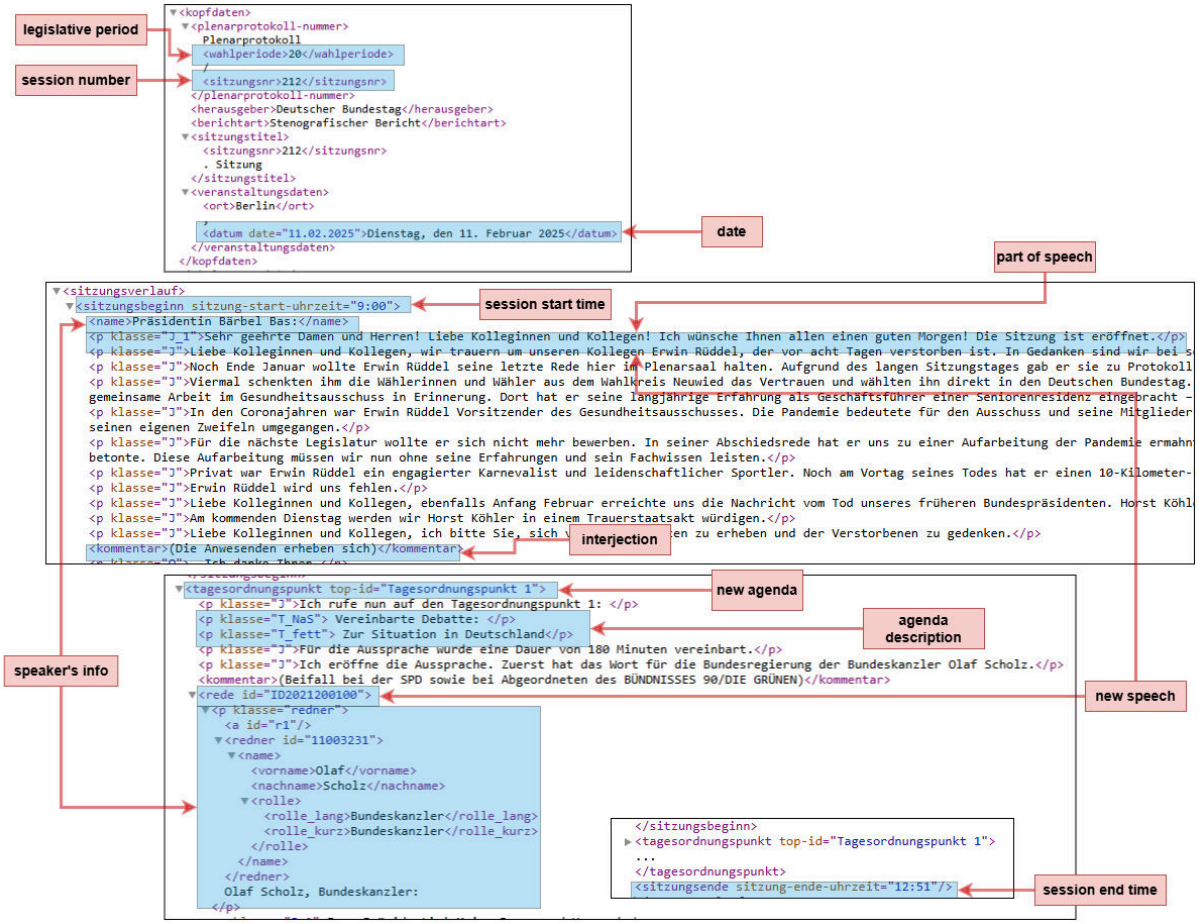


Figure 12: Example of the raw Bundestag Open Data data format

B Translation of the German text in Figure 3

- **CALL TO ORDER:** Mr. Baumann, I call you to order for the interjection "You agitator!".
- **INTERJECTION:** (Shout from the AfD: So, people can say "Nazi" here!) SPEECH: - One moment. Watch out! The President's calls to order will not be commented on.

- **INTERJECTION:** (Britta Haßelmann [Alliance 90/The Greens], addressing the AfD: Mr. Hilse, that goes for you too! - Counter-cry from Tino Chrupalla [AfD]: But also for you! - More shouts from the AfD)
- **CALL TO ORDER:** I call you to order for that, Mr. Hilse.
- **INTERJECTION:** (Stephan Brandner [AfD]: Unbelievable what's going on here! Unbelievable! - Shout-out: Yes, indeed!)

Author Index