arXivBot: A Large Language Model Chatbot That Has High Factuality and Coverage by Few-Shot Grounding on arXiv

Stanford CS224N Custom Project

Xiaofeng Tang
Department of Computer Science
Stanford University
xiao419@stanford.edu

Abstract

ArXivBot is a chatbot grounded on arXiv's corpus. The goal of this project is to develop a large language model (LLM) chatbot that has both high factuality and extensive coverage of knowledge intensive task.

ArXivBot retrieves and re-ranks relevant information from a vector database that breaks each arXiv article into passages and embeds them for semantic search. The retrieved information is then digested and reorganized by the LLM to form individual claims, filtering out non-relevant content to ensure accuracy. Comprehensive coverage is achieved through a two-hop query process, where the LLM refines the initial search results and generates a second query by chain-of-thought (CoT) to address any missing information. This allows it to exhaustively retrieve related information from the corpus to answer the question thoroughly.

We evaluate arXivBot and baselines by crafting questions that cover common topics users typically explore in the arXiv computer science category. These topics include author information, datasets, summaries, limitations and discussions, and comparisons between different methods. To ensure accuracy and recall, we thoroughly read the articles before preparing answers. The results show that arXivBot exhibits better accuracy and coverage compared to GPT-4 and GPT-4o, especially on the latest articles. Additionally, arXivBot outperforms ChatGPT, which has the capability of searching the internet.

1 Key Information

- Internal mentor: Sina J. Semnani (Stanford PhD student)
- TA mentor: Shikhar Murty

2 Introduction

Rapid advancements in reasoning, generalizability, and versatility have made Large Language Models (LLMs) incredibly helpful tools, quickly adopted by millions of people into their workflows. However, a well-known issue with LLMs is their tendency to hallucinate, fabricating misleading information with confident language. This occurs because LLMs find it difficult to admit their ignorance on a topic, undermining trust in their outputs and necessitating additional cross-validation costs. Moreover, LLMs face challenges in providing comprehensive coverage on knowledge-intensive tasks, often requiring information retrieval from multiple queries. This is especially true for datasets like arXiv, where much of the content is long-tailed and difficult for LLMs to generate directly [1].

To handle queries on a large database, one approach is to rely on a long context length window. For instance, Google Gemini 1.5 Pro boasts a context window of 1 million tokens, setting a record for

the longest context window to date. However, this is still insufficient for encompassing all the data in arXiv, which alone exceeds 2TB of text. There is study to expand the context window to 200 million tokens by only retaining the beginning and ending portions of the context [2]. This method, however, causes LLMs to overlook important details in the middle. Additionally, the cost and time delay associated with processing such a long context window hinder its practical application for large databases.

Another approach is to fine-tune the LLM on the target dataset, allowing it to remember the resources during inference. However, this method is computationally expensive and struggles to keep up with the continuous addition of new data to large corpora like arXiv.

Both previously mentioned approaches still suffer from the hallucination issue. ArXivBot employs a third approach, "retrieve-then-generate" which generates responses based on retrieved information to improve accuracy. ArXivBot enhances accuracy by filtering out non-relevant retrievals. The key to this approach is generating the right query based on the user's input, as the quality of the retrieval results significantly depends on the precision of the query. ArXivBot further uses a two-hop retrieval process that the LLM examines the evidence retrieved in the first round and generates a second query using chain-of-thought reasoning if any information is missing, ensuring a comprehensive answer to the question.

We evaluate our results using carefully curated questions and answers provided by humans, as this remains the gold standard for assessing a chatbot's performance. The results show that arXivBot exhibits significantly better accuracy and coverage compared to GPT-4 and GPT-4o, which tend to hallucinate on almost every question on arXiv, especially regarding the latest articles. Additionally, arXivBot outperforms ChatGPT, which relies on its own information retrieval method by searching the internet.

3 Related Work

There are severual knowledge-grounded chatbots developed based on information retrieval. BlenderBot2 [3] incorporates Internet search and was later surpassed by SeeKer [4] which uses a three-step strategy: query generation, knowledge extraction from retrieval and final response generation. BlenderBot3 [5] further finetuned a large 175B OPT [3] based on 20 dialogue datasets. And Atlas [6] is a state-of-the-art model on the KILT benchmark [7] with 11 knowledge-oriented tasks.

One of our baselines, WikiChat [8], achieves a factual accuracy of 97.9% on Wikipedia's recent topics, significantly outperforming GPT-4's 55.0%. WikiChat employs a strategy that combines retrieved information with fact-checked LLM generation. We built upon this architecture to better suit the content of arXiv by removing the claim generating step by LLM, as it tends to hallucinate frequently for long-tailed tasks on arXiv. Instead, we added a two-hop retrieval process to thoroughly search the database for comprehensive answers.

4 Approach

Baseline Architecture Our objective with arXivBot is to ensure high factuality and comprehensive coverage. To achieve this, we further developed the architecture based on our baseline WikiChat, as it has demonstrated a 97.9% factual accuracy rate with Wikipedia content [8]. The baseline is outlined in a 10-step pipeline (Fig 1), which operates as follows:

Upon receiving user input, the system initiates two parallel tasks. The first task, encompassing steps 2 through 4, involves generating a query based on the user's question, select the top three most relevant passages ranked by LLM from multiple retrieved passages, and a prompted LLM was then instructed to break these passages down into individual claims. These claims are prepared for inclusion in the initial draft response.

Simultaneously, the second task (step 5 to 7) involves the LLM engine providing a ungrounded, direct answer to the user's question. This answer is also split into claims, each of which is fact-checked by prompted LLM based on the information retrieved by the same methods in the first task. Only the claims that are verified as correct are selected for inclusion in the draft response.

The draft response, composed of both directly retrieved claims and validated claims from the LLM's direct answer, undergoes further refinement in step 9. This refinement process enhances the response's relevancy, temporal correctness, naturalness, and non-repetitiveness through few-shot prompted LLM techniques. The outcome of this step is Wikichat's final, polished response.

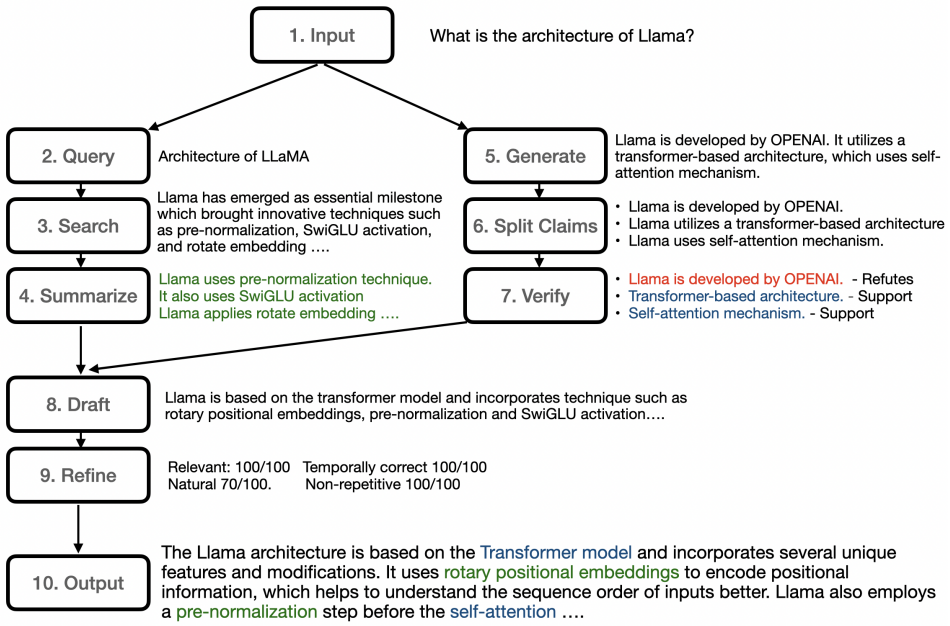

Figure 1: Baseline WikiChat workflow. And an example of conversation about Llama architecture, edited for brevity. The steps to generate outputs are 1). Input from user 2) Generate query to Qdrant database. 3). Retrieve top 3 relevant passages from corpus. 4). Summarize and split results into individual claims. 5). Directly generate using LLM, this step can be parallel to 2) to boost efficiency. 6). Split generated info into claims. 7). Query and retrieve info from corpus based on each claim and fact check each claim. 8). Combine the claims from both directly query and LLM generation to form a draft response. 9). Refine the draft to based on metrics of relevant, natural, temporally correct and non-repetitive. 10). Output the refined final answer.

ArxivBot Architecture While the existing architecture works well on WikiPedia, it is not designed to handle long-tailed corpora, such as arXiv. This limitation rendered steps 5-7, which directly generate claims from the LLM, ineffective, as the LLM often hallucinates on such topics. Consequently, we removed these three steps to enhance efficiency. Additionally, we removed step 9, Refinement, to further boost performance, as our primary focus is on accuracy and coverage and the LLM provides sufficiently good conversationality by default.

Additionally, the current architecture achieves high factuality but does not focus on providing comprehensive coverage of answers. To address this, we incorporated second-hop information retrieval. Specifically, we utilize the Chain of Thought (COT) approach to guide the LLM in assessing the sufficiency of the initially retrieved information. If the initial information is adequate, the LLM composes the answer using that data. If not, the LLM conducts a second round of information retrieval to gather the missing information. (Fig 2)

This method enables a more thorough search of the database and exhaustively explores potential answers. These enhancements aim to improve the depth and breadth of the information retrieved, ensuring that the answers provided cover all relevant aspects of the user's inquiry.

Other Baselines For our comparison, besides of WikiChat, we selected four more baselines: GPT-4, GPT-4o, ChatGPT Website (GPT-4) and ChatGPT Website (GPT-4o). We opted for GPT-4 and GPT-4o based on their proven high-quality response capabilities. Additionally, we included the ChatGPT Website versions because they offer the capability to search the internet, which is particularly beneficial for addressing queries related to arXiv's extensive and specialized content.

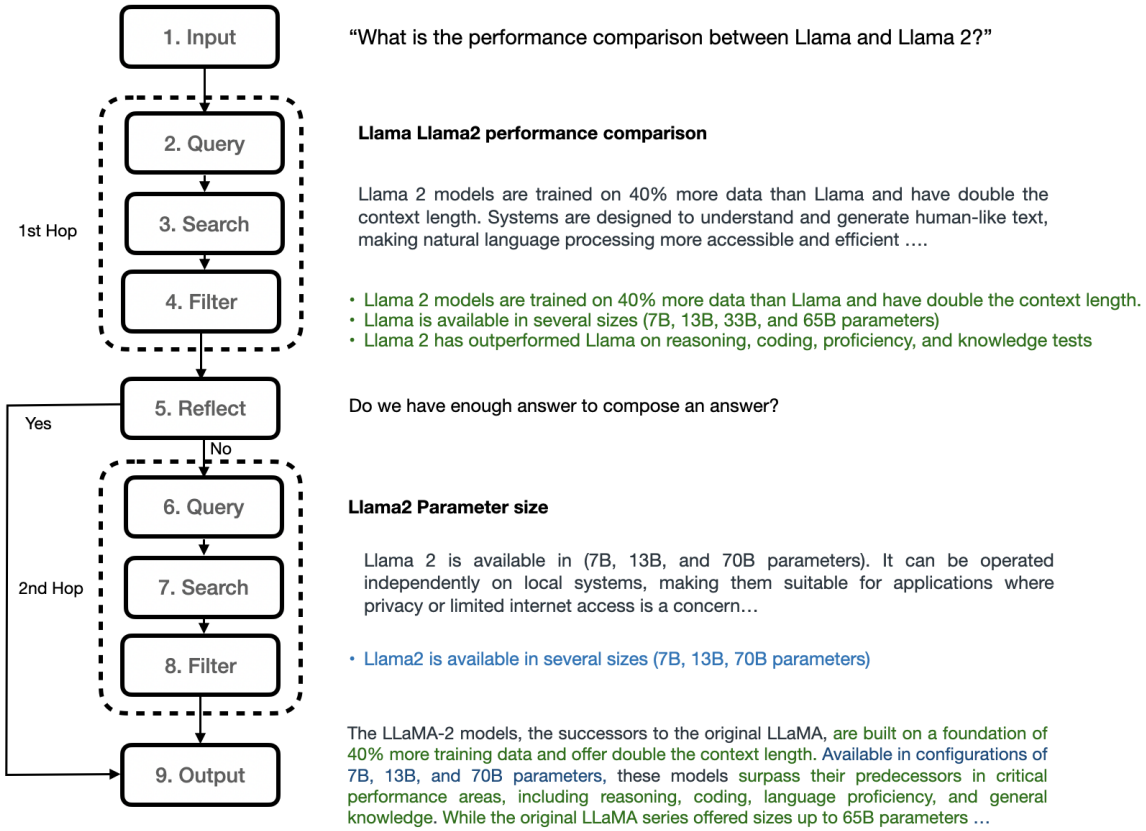

Figure 2: ArXivBot workflow. And an example of conversation about Llama and Llama2 comparison, edited for brevity. The steps to generate outputs are 1). Input from user 2) Generate query to Qdrant database. 3). Retrieve top 3 relevant passages from corpus. 4). Filter and split results into individual claims. 5). Use the LLM to reflect on whether sufficient information has been gathered. 6). If not, generate a second query to retrieve the missing information. 7). Retrieve top 3 relevant passages from corpus based on latest query. 8). Combine the newly fetched claims with those from step 4. Total six claims. 9). Output the final answer.

This internet-search functionality is advantageous over the standard API versions, which rely solely on pre-trained knowledge. To ensure consistency, we specifically prompt the ChatGPT Website to search arXiv before providing responses to relevant questions.

5 Experiments

Data. Our dataset is sourced from the arXiv HTML dump [1], which encompasses articles from 1991 to 2024. It is divided into three subsets based on the severity of LaTeXML conversion issues: no_problem, warning, and error, with respective sizes of 155GB, 2TB, and 753GB. Due to resource and timing constraints, we selected only the no_problem subset and focused exclusively on computer science papers.

The HTML dump does not categorize articles by subject nor include category information in the metadata, necessitating a filtering process through the API, which is rate-limited to approximately 4 seconds per query by the website. This limitation significantly slowed our ability to filter the papers. By the project deadline, we successfully narrowed down our selection to 27,447 papers published between 2020 and 2024.

[1] arXiv HTML dump `https://sigmathling.kwarc.info/resources/ar5iv-dataset-2024/`

Following the selection of papers, similar to the approach used in WikiChat, we preprocessed the articles by segmenting each into passages using the *mwparserfromhtml* library[2]. The majority of the passage length ranges from 20 to 300 words (see Appendix for length histogram). This process also involves extracting the title and subtitle for each passage, which aids in later retrieval steps. Subsequently, each passage was embedded into vectors using the BGE-M3 model[3], and these vectors were indexed using vector database Qdrant[4]. This indexing facilitates efficient search and retrieval in subsequent stages of our project.

The preprocessed data was only applicable to WikiChat and arXivBot. In contrast, GPT-4 and ChatGPT were queried directly. GPT-4 relies on its pre-training knowledge, while ChatGPT uses its own information retrieval system based on internet searches.

Evaluation method. To assess the accuracy and coverage of answers generated by the LLM, human evaluation remains the gold standard. However, thoroughly reading academic papers and preparing corresponding standard answers require a significant amount of time and resources, which we are notably constrained by. To expedite the evaluation, we recruited three students with Computer Science background and prepared 50 questions derived from an in-depth review of 15 papers, as of the project deadline. Each answer from the chatbot was then graded by human evaluator.

Experimental details. To evaluate the performance of the baseline LLM, we categorize the papers into "old" and "recent" groups, using December 2023 as the dividing line. This date corresponds to the knowledge cut-off for GPT-4, providing a relevant benchmark for comparing the LLM's performance against more current and older sources. This evaluation mainly focus on both the accuracy of the claims made and the coverage of the answers provided. The questions encompass a wide range of topics, including author information, datasets, summaries of the papers, their limitations, discussions, and comparisons between different methodologies. For a sample of these questions, please refer to the appendix.

Results. We developed 50 questions to assess the factual accuracy and coverage of each baseline, dividing them into two sets: 25 questions focused on older papers and 25 targeting newer publications. Each answer may contain multiple claims. An answer will receive points for factual accuracy based on the proportion of claims that are correct. Conversely, the chatbot's coverage is graded based on how many of its claims match those in the standard answer. Partial points can be earned. The final results are then converted into percentages for better presentation.

"Old" papers	Factual %	Coverage %
GPT-4	25	22
GPT-4o	28	25
ChatGPT Website (G4)	81	71
ChatGPT Website (G4o)	84	72
WikiChat (G4o)	95	87
ArXivBot (G4o)	94	93
"Recent" papers	Factual %	Coverage %
GPT-4	0	0
GPT-4o	0	0
ChatGPT Website (G4)	82	72
ChatGPT Website (G4o)	83	71
WikiChat (G4)	94	88
ArXivBot (G4o)	94	92

Table 1. Performance Comparison in Factual Accuracy and Coverage

We observed that the ChatGPT website, with its online searching capability, outperformed the GPT API version greatly in terms of both factual accuracy and coverage, particularly for newer papers published after the knowledge cut-off date. The GPT API version tends to hallucinate more frequently, likely due to these topics being absent during its pre-training phase and that is the reason we removed the direct LLM generation from our pipeline. Conversely, WikiChat demonstrated strong performance in factual accuracy, while arXivBot excelled in both factual accuracy and coverage.

[2]mwparserfromhtml library: `https://pypi.org/project/mwparserfromhtml/`

[3]BGE-M3: `https://huggingface.co/BAAI/bge-m3`

[4]Qdrant: `https://python-client.qdrant.tech/`

6 Analysis

Few-shot Prompt For information retrieval, the words extracted from a question are critical, as they often determine the quality of the answer. A few-shot prompt can greatly enhance this process. One example we observed involved the question, "How does RTVLM evaluate fairness?" where RTVLM is a Redteaming dataset. With zero-shot prompting, the LLM extracted the query words "RTVLM evaluate fairness," resulting in retrievals that only included results and findings from RTVLM instead of its methods. We then instructed the LLM in the prompt to focus on the "method" needed for questions related "how," which led it to extract "RTVLM evaluate fairness method" as the query. This adjustment returned a much better answer.

Two-hop Retrieval Similar to how humans refine their search queries on Google based on initial results, we apply the same concept to inspire our two-hop retrieval method. For example, when asked, "What is the largest pretraining dataset for LLM?" the initial search results included datasets only in Chinese due to the original query has a high cosine similarity with the top 3 most relevant passages which are about largest datasets in Chinese. In the second-hop retrieval, the LLM can be instructed to recognize this bias and refine the query to "Largest pretraining dataset for LLM in all languages" This adjustment helps in obtaining more relevant and comprehensive answers.

Query typo, synonym and hypernum. A common method to improve query precision is to correct any typos so the retrieval can match the exact keywords. Another widely used technique is to include synonyms or hypernyms of the words. However, these are not necessary applicable here because the LLM can effectively auto-correct typos. Additionally, vector-based search matches information with similar semantics, eliminating the need to generate alternative synonyms or hypernyms.

Limitation As arXiv contains more than 2TB of data, our preprocessed dataset, which is only 12GB, represents a very small proportion of the entire corpus. Sometimes, the lack of coverage in answers is not due to the architecture of the method but rather the insufficient database caused by the rate limit of arXiv's API. Though we restrict our questions to only articles available from the dataset, We anticipate that as we incorporate more data, the coverage score will improve, particularly for open questions that can be referenced from multiple sources.

7 Conclusion

This paper demonstrates how to create a chatbot using LLMs that achieves both high factual accuracy and coverage on knowledge-intensive tasks. Inspired by WikiChat's information grounding pipeline, we developed a second-hop retrieval method, which further boosted the answer's coverage, achieving 94% accuracy and 93% coverage overall. This same architecture can also be applied to other knowledge-intensive applications, such as chatbots for finance news or email assistance..

8 Ethical Consideration

One ethical consideration of this paper is the environmental impact due to hardware and energy usage. The computations for this research involved several GPU-hours on an NVIDIA A100 GPU for data preprocessing and hundreds of API calls to OpenAI for inference using GPT-4 and GPT-4o. However, it is difficult to estimate the precise environmental cost of using these models due to the lack of public information on their hardware and architecture. If this project were to scale up significantly, we could mitigate the energy cost by distilling the model into a smaller open-source model, such as Llama 3.

Another social consideration is data collection. This work focuses on helping users quickly understand academic papers available on the public arXiv platform. Most of the evaluations are conducted by three students without involving crowd workers, so we do not anticipate any harm from the proposed approaches. If this project scales up, fair compensation will need to be provided for each crowd worker, and they will need to be fully informed about the nature of the study. Additionally, we plan to release the code, few-shot prompts to encourage further research and development of more helpful chatbots.

References

[1] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and nonparametric memories. 2022.

[2] Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. 2023.

[3] Moya Chen, Douwe Kiela, Mojtaba Komeili, Spencer Poff, Stephen Roller, Kurt Shuster, Arthur Szlam, Jason Weston, and Jing Xu. Blenderbot 2.0: An open source chatbot that builds long-term memory and searches the internet. In *https://parl.ai/projects/blenderbot2/*, 2021.

[4] Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, , and Jason Weston. Language models that seek for knowledge: Modular search generation for dialogue and prompt completion. In *In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 373–393, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.*, 2022.

[5] Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal, Arthur Szlam, Y-Lan Boureau, Melanie Kambadur, and Jason Weston. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. 2023.

[6] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane DwivediYu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. 2022.

[7] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, , and Sebastian Riedel. Kilt: a benchmark for knowledge intensive language tasks. In *In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 2523–2544, Online. Association for Computational Linguistics*, 2021.

[8] Sina J. Semnani, Violet Z. Yao, Heidi C. Zhang, and Monica S. Lam. Wikichat: Stopping the hallucination of large language model chatbots by few-shot grounding on wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023, 2387-2413*, 2023.

Appendix: Histogram of Data Length

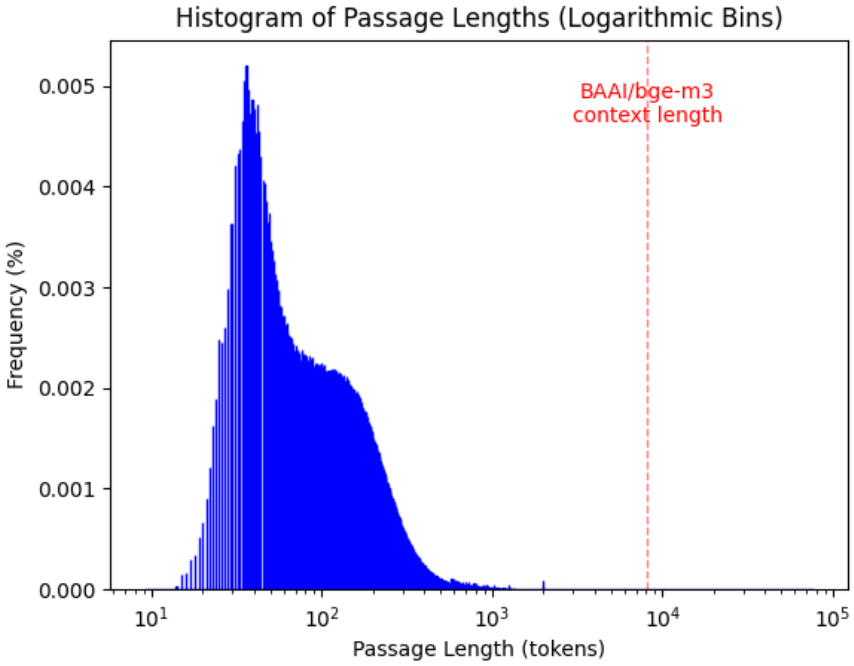

Figure 3: Histogram of passage length in corpus

Appendix: Sample of Questions and Answers

Article: Link Me Baby One More Time: Social Music Discovery on Spotify
Summary: With millions of track sharing events, this study aims to understand patterns and behaviors that enhance engagement with new music, offering insights into the effectiveness of social recommendations in digital platforms.
Questions:

- Who are the authors of the paper: Link Me Baby One More Time: Social Music Discovery on Spotify, and what is their affiliation?
  - Shazia'Ayn Babul, Desislava Hristova, Antonio Lima, Renaud Lambiotte and Mariano Beguerisse-D´ıaz
  - This work was joint effort between University of Oxford, The Alan Turing Institute and Spotify

  What datasets were used in this study, and how were they utilized to analyze music discovery on Spotify?
  - The researchers used data from Spotify, focusing on track share events within the Spotify social network.
  - They analyzed over 2.72 million events across 3,102 distinct artists to investigate how social and contextual factors affect users' likelihood to engage with new artists after receiving a recommendation.
  - They split the social media application type into two categories, direct or broadcast, depending on whether they represent one-to-one or one-to-many modes of communication.
  - They use vector representations of Spotify users and content in latent space. These vector embeddings are computed by word2vec algorithm on user generated playlists, with the playlists as the "documents" and the tracks as the "words".
- Can you summarize the key findings and limitations of this study as discussed in the paper?
  - The study found that the likelihood of a user engaging with a new artist increases if they share similar music tastes with the recommender, have strong social ties, and if the artist is popular within their social network.
  - There may be other important factors they are unable to capture with the data available such as the type of relationship between the sender and receiver, the context in which the link was shared, and so on.
  - The study's limitations include its reliance on Spotify's data, which may not generalize to other music platforms or non-digital music discovery methods.

Appendix: Sample of Few-shot Prompt

instruction
The following is a conversation between a friendly chatbot named chatbot_name , and a user. When responding to the user, chatbot_name carefully examine each result and then decide if there are sufficient information to answer user's question or need to further search arXiv to answer.

Respond only using the provided information from arXiv.

input
User: What ethical challenges does LLM pose?

Search results from arXiv:
- LLM poses several ethical challenges related to privacy, transparency, reproducibility, explainability, and potential biases in the data used for training these models.
- Biases in training data can lead to skewed outcomes, affecting decision-making processes and scientific research.
- Additionally, there are concerns about data privacy and consent when LLMs are integrated into applications, particularly in high-performance computing (HPC).
- It's also crucial to ensure that users and stakeholders understand how LLMs impact computations and results, and to establish accountability mechanisms to maintain trust and integrity in LLM-generated content.

Given the search results, do you need to search arXiv again to give an answer with better coverage? If so, what keywords you will use to search? If you think there are already sufficient info, then answer No.

output
Yes. You search "example of ethical challenges of LLM".

input
User: what is resumeAI?
Search results:
- ResumAI is an AI tool designed to assist users in enhancing their resumes and career trajectories, particularly in the finance sector.
- It provides tailored advice based on the user's existing resume and career goals.
- ResumAI is notable for its ability to support low-resource job applicants and college students by offering quick, personalized feedback.
- ResumAI also allows users to receive specific feedback on their resumes and ask for advice on various career-related topics, including career changes.

Given the search results, do you need to search arXiv again to give an answer with better coverage? If so, what keywords you will use to search? If you think there are already sufficient info, then answer No.

output
No.