An Interactive Question-Answering System using Large Language Model and Retrieval-Augmented Generation in An Intelligent Tutoring System on the Programming Domain

Owen Christian Wijaya
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
13520124@std.stei.itb.ac.id

Ayu Purwarianti
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
ayu@informatika.org

2024 11th International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA) | 979-8-3315-2031-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICAICTA63815.2024.10763263

***Abstract*—The insufficient communication between mentors and students has been one of the main disadvantages of modern programming learning platforms. In this paper, we propose the development of a web-based intelligent tutoring system with a question-answering (QA) system to provide live interaction between students and a mentor figure. We propose the implementation of an alternative QA system using a large language model (LLM) and a retrieval-augmented generation (RAG) mechanism. We utilized the LangChain library and integrated the RAG mechanism with the history-aware retriever and direct integration into the web application. We performed internal and external evaluations in the form of qualitative evaluation via subjective scoring towards answers from various quantized LLMs in both single-turn and multi-turn conversation scenarios. We conclude that the Llama 3 model displays consistent and promising results compared to other models and that documents with a higher character count may act as better knowledge bases for the RAG process.**

***Keywords*—LLM, question-answering, retrieval, chatbot**

I. Introduction

There has been a significant increase in the needs of digital talents in Indonesia in the past few years, especially in the programming industry. However, this increase led to a higher gap between the number of required talents and the actual number of digital talents produced by formal education [1]. There have been various attempts to minimize this gap, including the development of education platforms specializing in programming and tutorial websites providing knowledge for software engineering. However, one major weakness these solutions collectively possess was the insufficient interaction between human mentors and students, potentially leading to the loss of motivation in students to inquire further questions. Another arising problem is the contextuality of the inquired questions, where the questions asked may be specific to the student's requirements.

An intelligent tutoring system (ITS), a learning system developed with artificial intelligence-based features that provides instructions and aid without human assistance [2], may provide a solution to these problems. One example is with the integration of a question-answering (QA) system with an interactive interface into the learning process. An interactive QA system enables students to inquire questions and further provide context related to the question in a chat-like manner to ensure a more natural interaction, along with the timely response provided by the system [3].

The rapid uprising of LLMs provides an opportunity to utilize LLMs as an alternative to conventional QA systems. One outstanding method suitable for augmenting LLMs for the QA system's use case is the retrieval-augmented generation (RAG) method, providing external knowledge while maintaining flexibility towards unknown data without having to fine-tune a LLM [4]. The prompt engineering method may be used in tandem with RAG to provide further context for the LLM and align the LLM output with the users' needs [5]. This paper focuses on developing a pipeline combining LLMs with the RAG method, designed as a QA system in the ITS application to respond to inquiry questions.

We introduce a subjective method to evaluate answers from LLMs in this research. We established several qualitative metrics relevant to the answers a LLM may provide in the usage as an interactive QA system, then performed scoring with a Likert scale based on the established metrics. Through this method, it is possible to conduct comparisons between various LLMs based on their answers by checking the fulfillment of several criteria. We also performed a public evaluation to gather external evaluation that may support our internal findings. The introduced solution also utilizes the history-aware retriever mechanism that enables query reformulation based on the user's chat history and current question, ensuring that the retrieved documents are still relevant to those two aspects.

II. Related Works

The significant increase in LLM research and utilization has led to various studies being performed in the specific topic. This section introduces and discusses past research and studies relevant to the development of the QA system in this paper. Studies discussed will cover relevant mechanisms that are related to the final implementation of the QA system. This section will also discuss the existing intelligent tutoring systems for students.

LLMs are a variant of LMs, probabilistic artificial intelligence models based on natural language that produce a series of probabilities of words that may show up to complete a sequence [6]. These models have received major attention and advancements over the past few years in the generative AI field due to its ability to process long inputs

and produce comprehensible, natural answers. LLMs are based on the transformer architecture [7] to enable parallel processing, allowing encoding-decoding processes in a timely manner. Recently, LLMs have been considered to be an alternative to interactive systems due to its ability to provide human-like responses, and are progressing rapidly with both proprietary companies and open-source communities frequently releasing new models. One of the more recent open-source models, Llama 3, has shown competitive performance when compared to both open-source and proprietary models [8].

There have been various existing ITS systems in the programming domain; one notable example being the CPP-Tutor system [9]. This system utilizes a knowledge base of the C++ programming language and an expert intent model to provide corrections and proper suggestions towards the user's code. This system provides a humanlike interaction with the user; however, the interaction is focused only on the use case of code correction. Our system is aimed to provide a natural interaction with various use cases, not limited to code correction.

Previous developments of a QA system with a retrieval system have been carried out before using a retriever-reader architecture [10]. The retriever mechanism is utilized to obtain relevant documents by using the inquired question as a query and comparing it to the document collection. The reader mechanism would then find the relevant information contained in the retrieved document. The TF-IDF vectorizer is utilized to convert the documents and the query into the embedding form. Relevant documents would then be retrieved by computing the cosine similarity between the embedding representation of a document and the query, ranking the documents based on the cosine similarity, and taking the top-ranked documents. Our research further extends this study by utilizing RAG, which is retriever-based and utilizes similarity search between documents using transformer-based embedding models.

Studies regarding the potential of LLMs as an alternative to existing QA systems have shown promising results [11]. On an experiment to compare LLMs with knowledge-based QA (KBQA) systems on various open-ended QA datasets, both the proprietary GPT-4 model by OpenAI and the open-source FLAN-T5 model by Google have proven to be on par with conventional fine-tuned state-of-the-art (SOTA) KBQA, reaching similar or higher accuracy rate on six datasets and better results by GPT-4 on multilingual datasets. Our study extends the findings in this research by integrating the RAG method with LLMs, further enhancing LLM capabilities to answer domain-specific questions with document retrieval and the ability to process long inputs.

There have been various studies related to the usage of retriever mechanisms to overcome limitations of an LLM [12]. A retriever mechanism provides LLMs access to various resources otherwise unreachable, such as external knowledge bases, persistent memory mediums, or tools such as search engines. The RAG mechanism utilizes knowledge retrievers to obtain knowledge outside of a LLM's internal knowledge to generate answers relevant to the knowledge base. RAG eliminated the requirement to fine-tune a model for a specific downstream task, and instead provided access to external knowledge as a non-parametric memory [13]. One such example in a QA system is the chat history between the user and the model, where previous chat history may be relevant to the conversation. We utilized this mechanism to retrieve external knowledge stored in a vector database and chat history stored in a chat history database in order to perform long-context modeling.

Qualitative evaluation regarding LLMs has also been intensively researched for the past few years [14]. One significant study involves utilizing a LLM as an external judge to compare answers between two LLM agents. Despite the LLM limitations, the study showed promising results, with judgments by the LLM aligning with human results. This study also introduced MT-bench, a suite to evaluate LLMs on open-ended multi-turn conversations. We took the inspiration from MT-bench and further developed a small multi-turn conversation dataset comparison to assess the system's performance in multi-turn question-answering.

For evaluating answers for an interactive QA system in a qualitative manner, various metrics must be established to determine whether the answers provided by the system fulfill the desired criteria. One study emphasizes two main criteria; accuracy to evaluate whether the answers provided by the system are factually correct, and usefulness to measure whether the provided answers assist the user's needs and answer the user's questions [15]. These key metrics can reflect the key qualities of an answer provided by a QA system and can be further developed and used for further qualitative assessments. We extended these metrics into several novel metrics that reflected the original metrics and evaluated more crucial qualities in the answers.

III. Design and Implementation

This section discusses and explains the proposed architecture of the developed QA system, alongside various components and considered justifications. This section also discusses various methods performed during the implementation phase of the QA system.

A. Processing External Documents

The learning modules and knowledge were originally written in Indonesian and were contained inside PDF files for reading purposes. To enable the LLM to access the information inside of these documents, we performed several processing steps to convert the information into a retrievable collection of documents. These steps are visualized in Figure 1. Firstly, we extracted and translated the information from each page inside the PDF files. We translated the information in bulk using DeepL Translator due to its capability in producing highly accurate translations. The result was the translation of each PDF file stored inside a text file, where each PDF file may contain specific subtopics.

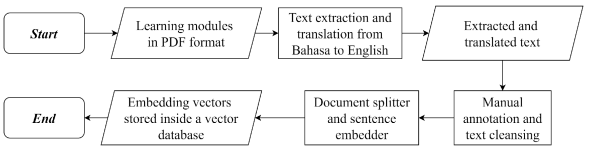


Fig. 1. The document processing steps

The extracted text may contain duplicated or incomplete text chunks. To maintain the data quality, we performed

manual annotation and text cleansing on the extracted and translated texts. The result of this step was a structured document that could be segmented into specific subtopics. Document splitting was performed by splitting each document into smaller documents, where each document contains information about a specific subtopic. This is to maintain the context of each document so that it would ideally contain information about a specific subtopic that may be asked later, so that the provided document includes complete information about the inquired subtopic. We established a specific chunk size and split a lengthy document into smaller documents based on the chunk size.

B. Prompt Engineering

We used the CO-STAR prompting method because of its ability to convey the rules and prompts required by a LLM to produce an expected answer [16]. CO-STAR is an acronym of each property that comprises it, with each property defining the purpose of the agent. Table I shows the pairing of each property in a CO-STAR prompt and the description of its usage in the prompt.

TABLE I. PROPERTIES OF THE CO-STAR PROMPT

Property	Usage
Context	Define that the agent is a chatbot in a QA system inside a learning platform.
Objective	Provide answers based on the previous chat history and retrieved documents. Do not answer malicious questions or questions outside of the programming domain.
Style	Be friendly and polite, provide analogies and examples whenever possible.
Tone	Follow a friendly and helpful tone.
Audience	Beginners in programming, ranging from high schoolers to university freshmen.
Response	Provide at most eight sentences, excluding analogies and examples.

C. Developing the Pipeline

We developed the pipeline using LangChain, a renowned open-source library capable of integrating various tools and databases for various applications utilizing LLMs. The developed pipeline provided the LLM access towards an external knowledge base and a persistent memory medium. Users could interact with this system in a conversational manner due to its ability to comprehend and utilize previous chat history. The system is accessible inside each material page on the website, each page containing a different chat history to maintain the discussion in a specific page. Below are the three main components interacting in the pipeline.

- **Chat history database:** We used Redis to maintain the chat history information due to its efficient caching mechanism, a key-value schema that is sufficient for this use case, and its support for persistent storage. The history of the chat is stored using an unique identifier to differentiate history per material page.
- **Retriever:** In LangChain, a retriever retrieves the relevant documents contained in the vector database based on the user query. A history-aware retriever combines the capabilities of retrievers and those of LLMs to construct a new query based on the previous chat history to retrieve relevant documents based on the current user question and the previous chat history. This component is built-in into LangChain's functionalities and could be integrated into the QA system.
- **LLM:** The LLM would generate the final answer by streaming the answer to the website and generate a new search query in the history-aware retriever. Due to the limited resources available, we utilized quantized LLMs running on a local environment.

The developed pipeline would be evaluated and experimented upon to find the compatible LLM and document size for the retrieval process. The evaluation methods and results are to be discussed in Section 4 and 5 of this study.

IV. EXPERIMENTS

This section discusses the experiment methods conducted on the developed pipeline. The experiment methods would involve comparison between various quantized open-source LLMs and examination of the RAG mechanism. Additionally, we conducted external research to gather external opinions and to further support the internal findings found in the experiment methods.

A. Data

We conducted the experiment on a RAG pipeline with its own knowledge base. In order to ensure a relevant and correct evaluation process with the knowledge base, we developed our own datasets with consideration to the knowledge base. We developed 55 single-turn questions and 10 multi-turn question groups, with each question group containing 3-4 questions. These questions did not have a paired answer, as the focus of the experiment was to compare answers between various models and determine the model with outstanding qualities that is suitable for the QA system. To evaluate the retrieval process, we developed a small document test collection by taking 14 questions from the 55 single-turn questions and selecting the relevant documents retrieved from using those questions as a query.

B. Models

For embedding models, we selected a few models to compare based on the MTEB benchmark. Notable models include:

- **e5-large-v2:** This model was used as a baseline model due to it outperforming models with 40x its parameters in the MTEB leaderboard. This model shows generally favorable results for tasks requiring single vector representations.
- **jina-embeddings-v2-base-en:** This embedding model was specifically developed for long-context embedding up to 8192 tokens, and is suitable for RAG tasks.
- **gte-en-v1.5:** This embedding model had undergone contrastive pre-training with data from various sources. This model managed to outperform other code embedding models due to it treating code chunks as general purpose text during the pretraining and fine-tuning process.

In this experiment, we compared two types of models: LLMs as the main focus of this experiment, and embedding models to measure the model with the best performance on the retrieval process. We picked 17 LLMs from the Ollama model library, with most models having 7B parameters.

Each LLM is quantized to 4-bit weights to ensure comparison fairness and efficient inference speeds. We conducted initial qualitative screenings towards the initial models to determine the best performing model for each family. Table 2 describes the grouping of these models based on their families. In the table, we discussed the reason for the LLM family selection and the selected model, marked by the bolded model name.

TABLE II. CATEGORIZATION OF LLM MODELS BASED ON MODEL FAMILES

Family	Reason of Selection	Models
Llama2	Existed longer than other models with various variants, selected for benchmarking	**CodeLlama-7B Instruct** Llama2-7B Chat Vicuna-7B-16K
Gemma	Competitive results, providing a more structured output and guidance to users	Gemma-7B Base Gemma-7B Instruct CodeGemma-7B Base **CodeGemma-7B Instruct**
DeepSeek	Exceptional performance in coding and mathematics for an open-source model	DeepSeek-Coder-6.7B Base **DeepSeek-Coder-6.7B Instruct**
Mistral	Impressive capability of providing humanlike interaction, suitable for chatbots and assistants	Mistral-7B **Zephyr-7B** **Notus-7B**
Llama 3	The latest open-source model (at the time of writing), improved upon Llama2	**Llama 3-8B**
Qwen	Promising results compared to other models in its parameter class	Qwen-1.5-7B Base Qwen-1.5-7B Chat
Phi	Smaller size compared to other models, however capable of providing similar results in benchmarks	Phi-3B-Base Phi-3-3.8B Base

There are no models chosen from the Qwen and Phi families due to the significant quality degradation caused by the quantization process. Two models, Zephyr and Notus, are elected from the Mistral family due to the quality from both models being on par during the initial screening, and thus further comparison was required. Llama 3 only has one model candidate due to it being a novel model during the experiment, and thus included due to its consistency in quality even when compared to other models.

C. Methods

We conducted internal experiments by performing subjective scoring towards the answers provided by the developed pipeline on single-turn and multi-turn questions. Each elected model would be used in the developed pipeline to answer the provided questions. For single-turn questions, we would evaluate the answer provided by each model for each question. For multi-turn questions, we would evaluate the coherence of the conversation produced between the questions provided by each question group and answers to each question.

The evaluation metrics we used were derived from the related study about chatbot benchmarking [15]. For each metric, we performed the scoring using a Likert scale ranging from 1 to 5. For each metric, the high score resembles a better quality for the metric. Table III describes the metrics we used during the evaluation.

TABLE III. QUALITATIVE METRICS FOR ANSWER EVALUATION

Metric	Description
Friendliness	Evaluates the style of answers provided and whether the answer is understandable. A novel metric introduced in this study.
Conciseness	Evaluates whether the model provides a lengthy or a concise explanation. Resembles the usefulness metric.
Helpfulness	Evaluates whether the model successfully aids the user's requirements or whether the model provides necessary analogies or examples that may further assist the user. Resembles the usefulness metric.
Faithfulness	Evaluates whether the model utilizes the retrieved documents to generate answers. Resembles the accuracy metric.

To evaluate the retrieval process, we used the mean average precision (MAP) metric, commonly used to evaluate document retrieval systems. Additionally, we conducted subjective evaluation with a Likert scale similar to evaluation with the qualitative metrics. The score would represent the contextuality and relevance of the documents retrieved during the retrieval process. We selected different chunk sizes to simulate different document sizes during the retrieval. For each chunk size, we scored the retrieval results and compared the results between embedding models.

For external evaluation, we designed an online questionnaire in order to enable respondents to provide their own Likert scale scoring on single-turn questions. The questionnaire would provide scoring results for the friendliness, conciseness, and helpfulness metrics. The faithfulness metric was not included due to its lengthy comparison process involving comparison with retrieved documents. Respondents would be able to rank the models on each question based on their preferences and the quality of each model's score. The model names were disambiguated to prevent bias towards a preferred model. By the end of the questionnaire, respondents would provide their opinions on which model suits the best as a QA system.

V. RESULTS

This section discusses the results of the conducted evaluation methods discussed in Section 4. For each evaluation method, we provide the results, error analysis, and conclusion from the evaluation method.

A. Evaluation towards Retrieval Results

Figure 2 visualizes the score accumulation for each chunk size. For each chunk size, we compared the embedding results for each candidate model. A lower score represents a worse ranking or cut documents, while a higher score represents a better ranking or documents with better splitting processes. We can see that splitting the documents with the chunk size set to 1000 provided a higher score compared to the chunk size set to 200 or 600. This is due to the resulting document on chunk size 1000 having more subtopic information stored inside of it, thus providing a more holistic view towards a certain subtopic during the scoring process.

In terms of ranking, these models may not seem to significantly vary in hindsight, so MAP calculations were performed on the sample document collection to ensure fair judgment towards the ranking results. Table IV shows the difference between MAP and the memory usage between each model.

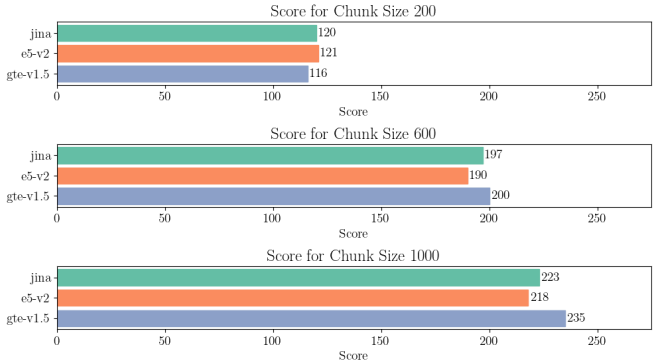


Fig. 2. Subjective scoring results for the retrieval results

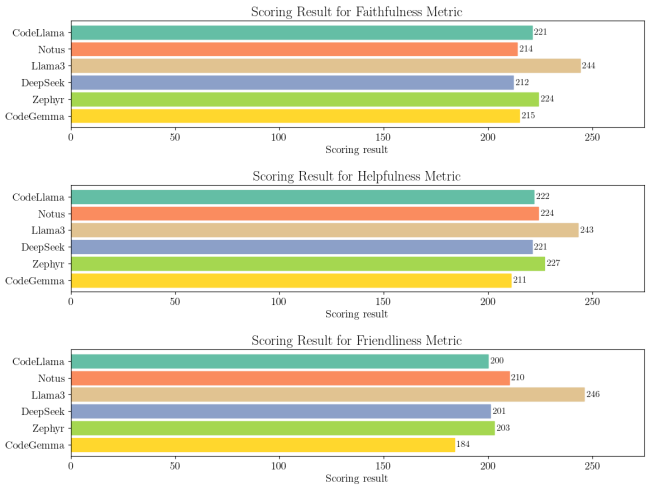
TABLE IV. COMPARISON BETWEEN MAP SCORE AND MEMORY USAGE PER EMBEDDING MODEL

Model	MAP Score			Memory Usage
	200	600	1000	
jina-embeddings-v2	0.7781	0.8191	0.8224	3270 MiB
e5-large-v2	0.7182	0.6856	0.6819	1370 MiB
gte-en-v1.5	0.6516	0.8294	0.7973	648 MiB

Both Jina and GTE models provided similar MAP results on higher chunk sizes. However, the GTE model used noticeably less memory compared to the Jina model, thus more usable on low-resource environments. It could be concluded that both Jina and GTE models provided favorable results, and using a higher chunk size would result in better retrieved documents on RAG tasks.

B. *Subjective Scoring towards Single-Turn Results*

Figure 3 visualizes the score accumulation for subjective scoring for each qualitative metric. A lower score implies a lower model's answer quality on a specific metric, while a higher score implies the metric being reflected better in the model's answer. Based on these results, it can be concluded that Llama 3 consistently reaches the highest score on every metric compared to other models. The preceding CodeLlama model performed well on the conciseness metric, however the model tends to display the retrieved documents as-is without modification. The Notus and Zephyr model performed well on the friendliness and helpfulness metric, considerably because they are both fine-tuned models for chatting purposes.



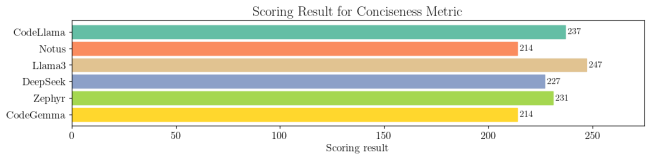


Fig. 3. Subjective scoring results for each qualitative metric

C. *Subjective Scoring towards Multi-Turn Results*

Table V displays the scoring results for multi-turn results. Both Llama 3 and CodeGemma performed best with a similar score, each model showing different strengths. Llama 3 excelled in aiding the user with a friendly and helpful manner, providing a friendlier experience with the user. While being less friendly, CodeGemma tends to provide elaboration and information to further emphasize previous explanations with a concise manner. CodeLlama and Zephyr also demonstrated a coherent multi-turn conversation.

TABLE V. MULTI-TURN SCORING RESULTS

Model	Score
CodeLlama-7B Instruct	44
Notus	37
Llama 3	48
DeepSeek-Coder-6.7B Instruct	39
Zephyr	42
CodeGemma-7B	48

D. *Questionnaire Results*

We distributed the questionnaire to fourteen respondents. The reason for the small number of respondents is due to the high effort required to fill in the questionnaire, and thus we prefer results with a higher consistency and quality over the volume of results. The model names are disambiguated to reduce bias towards selecting a specific model.

Figure 4 visualizes the accumulated ranking points for all models, where each model receives a ranking placement based on its quality on answering a particular question. For each model, the score is calculated by assigning points based on its ranking in different categories. Points are awarded as follows: 6 points for a first-place position, 5 points for second position, 4 points for third position, and so on, down to 1 point for a sixth-place position. The total score for each model is the sum of points earned across all categories.

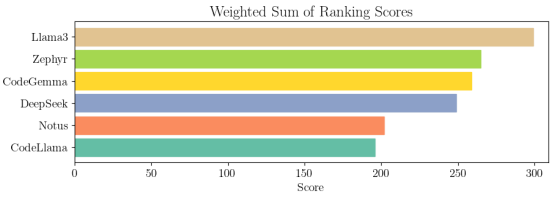


Fig. 4. Weighted sum of ranking scores

Figure 5 visualizes the respondents' voting on the most favorable models. Most respondents favor the usage of Llama 3 in the QA system, with a significant difference of votes received compared to other models' votes. Other favored models from the voting results were DeepSeek,

Zephyr, and CodeGemma, whereas CodeLlama and Notus were not generally favorable with lower voting results.

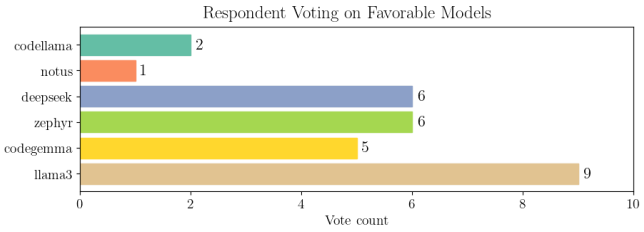


Fig. 5. Respondent voting on favorable models

Based on the questionnaire results, it can be concluded that Llama 3 is suitable for the usage in the QA system, according to the results received from the questionnaire. This supports the internal findings that also suggests that Llama 3 fulfills the qualitative metrics desirable for an interactive QA system. Therefore, we can conclude that Llama 3 shows an overall consistent quality and is preferable for the QA system we developed based on both internal and external findings.

VI. Conclusion

We conclude that LLMs can be used as a promising alternative of conventional KBQA models for an interactive QA system in a learning environment, specifically in an intelligent tutoring system. By utilizing the RAG mechanism for knowledge retrieval and long-context modeling, we successfully implement an interactive QA system for answering user questions based on the learning documents and the previous chat history. Our qualitative result suggests that Llama 3 is an excellent model for a friendly and helpful QA system, and that documents with a higher chunk size and split based on knowledge topics provide a more holistic view of knowledge for the RAG mechanism and thus are more helpful.

We suggest developing more quantitative-based approaches for future research and utilizing non-quantized model weights to ensure a more fair comparison between models. We also suggest exploring the impact and significance of the QA system more with testing towards end users. This is to further accentuate the significance of the conducted research by testing it under the real-world scenario.

References

[1] A. Jatmika and Widiarini, Annisa, "Indonesia Butuh 9 Juta Talenta Digital pada 2030, Apa yang Perlu Dipersiapkan Pelaku Industri?," KOMPAS.com. Accessed: Nov. 06, 2023. [Online]. Available: https://money.kompas.com/read/2023/06/09/150600726/indonesia-butuh-9-juta-talenta-digital-pada-2030-apa-yang-perlu-dipersiapkan

[2] "Design Recommendations for ITS_Volume 4 - Domain Modeling Book_web version_final.pdf." Accessed: Jun. 28, 2024. [Online]. Available: https://gifttutoring.org/attachments/download/1736/Design%20Recommendations%20for%20ITS_Volume%204%20-%20Domain%20Modeling%20Book_web%20version_final.pdf

[3] "893CorbettAnderson1995.pdf." Accessed: Sep. 04, 2024. [Online]. Available: http://act-r.psy.cmu.edu/wordpress/wp-content/uploads/2012/12/893CorbettAnderson1995.pdf

[4] Y. Gao *et al.*, "Retrieval-Augmented Generation for Large Language Models: A Survey," Mar. 27, 2024, *arXiv*: arXiv:2312.10997. Accessed: Sep. 04, 2024. [Online]. Available: http://arxiv.org/abs/2312.10997

[5] J. White *et al.*, "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT," Feb. 21, 2023, *arXiv*: arXiv:2302.11382. doi: 10.48550/arXiv.2302.11382.

[6] D. Jurafsky and J. Martin, "N-gram Language Models," in *Speech and Language Processing*, vol. 3, 2024. Accessed: Oct. 11, 2023. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/3.pdf

[7] A. Vaswani *et al.*, "Attention Is All You Need," Aug. 01, 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[8] "Introducing Meta Llama 3: The most capable openly available LLM to date." Accessed: Jun. 29, 2024. [Online]. Available: https://ai.meta.com/blog/meta-llama-3/

[9] "Developing an Intelligent Tutoring System For Students Learning To Program in C++." Accessed: Sep. 04, 2024. [Online]. Available: https://scialert.net/abstract/?doi=itj.2008.1055.1060

[10] G. N. Ahmad and A. Romadhony, "End-to-end Question Answering System for Indonesian Documents Using TF-IDF and IndoBERT," *Int. Conf. Adv. Inform.*, Oct. 2023.

[11] Y. Tan *et al.*, "Can ChatGPT Replace Traditional KBQA Models? An In-depth Analysis of the Question Answering Performance of the GPT LLM Family," Sep. 20, 2023, *arXiv*: arXiv:2303.07992. doi: 10.48550/arXiv.2303.07992.

[12] P. Zhang, S. Xiao, Z. Liu, Z. Dou, and J.-Y. Nie, "Retrieve Anything To Augment Large Language Models," Oct. 25, 2023, *arXiv*: arXiv:2310.07554. doi: 10.48550/arXiv.2310.07554.

[13] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Apr. 12, 2021, *arXiv*: arXiv:2005.11401. doi: 10.48550/arXiv.2005.11401.

[14] L. Zheng *et al.*, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," Dec. 23, 2023, *arXiv*: arXiv:2306.05685. doi: 10.48550/arXiv.2306.05685.

[15] D. Banerjee, P. Singh, A. Avadhanam, and S. Srivastava, "Benchmarking LLM powered Chatbots: Methods and Metrics," Aug. 08, 2023, *arXiv*: arXiv:2308.04624. doi: 10.48550/arXiv.2308.04624.

[16] Galvao, Sebastian, "Enhancing LLM prompting with CO-STAR." Accessed: Mar. 16, 2024. [Online]. Available: https://www.linkedin.com/pulse/enhancing-llm-prompting-co-star-sebastian-galvao-aaxef/