47054549

A Comparison of Retrieval Augmented Generation System (RAG) Using Different Combinations of Retrieval and Large Language Models (LLM)

Muhammad Yahya Bin Mohd Yunus, *Student, The University of Queensland*

. I. INTRODUCTION

State-of-the-art Large Language Models (LLM) such as GPT-4 reached remarkable accomplishments in tasks such as question answering, manipulating knowledge, and summarizing. LLMs can perform those tasks without any external information sources since they rely solely on pre-trained parameters from extensive amounts of factual knowledge, thus they are also called Parametric Language Models. However, the Parametric Language Models still provide factually incorrect outputs or "hallucinations" as they struggle to use external knowledge that is not within their pre-trained parameters [1].

In the recent past, the Retrieval-Augmented Generation (RAG) system has emerged as a promising solution to this issue, by first retrieving the top-k most relevant documents from external knowledge store to act as non-parametric memory before combining the generative capability of Parametric Language Models to synthesize the output based on the top-k documents. Utilizing the RAG system allows LLMs to produce more contextually accurate outputs based on the retrieved documents [1, 2].

The objective of this study is to compare the performance of Retrieval Augmented Generation Systems (RAG) using different combinations of Retrieval Models and LLMs. To this end, the RAG system was divided into two key stages named the Retrieval Stage and the Generation Stage. In the Retrieval Stage, two retrieval models named Ranker A and Ranker B were used to find the top 10 most relevant documents in a corpus consisting of 609 documents for each query from a set of 2,556 queries. Additionally, two reranker models - *BAAI/bge-reranker-base* and *BAAI/bge-reranker-large*, were applied on top of each retrieval model to further refine the top 10 retrieved documents.

Thus, the retrieval stage provides six different model configurations, 2 rankers and 4 rerankers, namely, rankerA, rankerB, rerankerA, rerankerB, rerankerC, and reranker D which will be explained further in Section II.

For the Generation Stage, the Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and R-precision (Rprec) were used to identify the best-performing ranker and reranker that provides the top 10 documents, which were then used in the question (query) input to four different LLMs called Llama2, Llama3, Mistral, and Zephyr. By evaluating the BERTScore metrics for the generated outputs, this paper aims to study whether the reranker plays a significant role in the effectiveness and efficiency of a RAG system. Moreover, this paper also aims to study the efficiency and effectiveness of the four different LLMs in an RAG framework.

II. METHODOLOGY

This section will describe the models used for both the retrieval and generation stages of the RAG system.

A. Retrieval Stage

1) **Ranker A - BAAI/llm-embedder**
   Ranker A is a pre-trained dense retriever that utilizes transformer-based architecture called *BAAI/llm-embedder*. It works by embedding queries and documents into individual vector spaces before efficiently calculating the similarity between the vectors, typically using cosine similarity, and retrieving the most similar (relevant) documents related to each query [3]. In this study, Ranker A was used as the baseline model for retrieval evaluation.
2) **Ranker B - BAAI/bge-large-en-v1.5**
   Ranker B's working mechanism is almost the same as Ranker A's except that it uses a more advanced transformer-based retriever named *BAAI/bge-large-en-v1.5*. It also enables more accurate retrieval on large-scale document collections as it was trained using more parameters [4].
3) **Reranker A - BAAI/llm-embedder with BAAI/bge-reranker-base**

RerankerA uses a cross-encoder-based reranker called *bge-reranker-base* that reorders the top 10 documents retrieved by Ranker A. Moreover, rather than treating documents and queries as separate embeddings like dense retrieval models, cross-encoders assess the query in conjunction with the specific document which improves relevance re-ranking based on semantic content [4].

4) **Reranker B - BAAI/llm-embedder with BAAI/bge-reranker-large**
The retrieval of Reranker B works the same as Reranker A since they both use the same retrieval model (Ranker A). However, Reranker B uses a cross-encoder-based reranker with a bigger parameter set, *bge-reranker-large*, that should improve its powers of semantic analysis [4].
5) **Reranker C – BAAI/bge-large-en-v1.5 with BAAI/bge-reranker-base**
Reranker C makes use of Ranker B for retrieval and uses the same cross-encoder-based reranker used in Reranker A, *bge-reranker-base*.
6) **Reranker D - BAAI/bge-large-en-v1.5 with BAAI/bge-reranker-large**
The Reranker D makes use of Ranker B for retrieval and uses the same cross-encoder-based reranker used in RerankerB, *bge-reranker-large*.

The coding for the rankers and rerankers was written in Python files before they were staged using the batch script 'stage1.sh' and were run using the GPU cluster on the UQ Cloud Zone. After the evaluation of JSON files obtained by the Retrieval Stage, the resulting top 10 documents by the best ranker and reranker were included in the question/query for the four LLMs in the Generation Stage.

B. Generation Stage

The llama-index framework was used to connect the data sources to the four supported open-source LLMs with only a few changes [5] to the same Python code. Two major changes needed were the model's name and the split token.

1) **Llama 2 - meta-llama/Llama-2-7b-chat-hf**
The Llama-2-7b-chat-hf is an open-source pre-trained and fine-tuned text generation model by Meta/Facebook having 7 billion parameters which are optimized for cases such as dialogue use in English. The model is recent, as the pre-training data cutoff was in September 2022. The split tokens in the code were changed to [/INST] and </s> to correctly identify the generated outputs by llama2 [6].
2) **Llama 3 - meta-llama/Llama-3.1-8B-Instruct**
The Llama-3.1-8B-Instruct is the more recent version of the Llama 2 above as pretraining data has a cutoff of March 2023 and they were both published by the same developer company. The Llama-3.1-8B-Instruct has a higher parameter with 8 billion parameters. The split tokens used are different to llama2 where llama3 were observed using <|end_header_id|> and <|eot_id|> [7].
3) **Mistral - mistralai/Mistral-7B-Instruct-v0.1**
Mistral-7B-Instruct-v0.1 was developed by MistralAI and released under the Apache2.0 license. It is the fine-tuned Instruct version of the Mistral-7B-v0.1 generative text model which was trained on a bunch of publicly available conversation datasets, and it has 7 billion parameters. The generated text consists of the prompt wrapped around [/INST] and the output, located before the end of the token, </s> [8, 9]. Thus, the split tokens used by Mistral-7B-Instruct-v0.1 are the same as Llama-2-7b-chat-hf.
4) **Zephyr - HuggingFaceH4/zephyr-7b-alpha**
Zephyr-7B-alpha is the fine-tuned version of Mistral-7B-v0.1 that removed the in-built alignment of the training datasets, which boosted the performance. However, when prompted to generate controversial text, it is likely to do so [10]. The split tokens used were <|user|> and </s>.

II. EXPERIMENTS

A. Retrieval Stage Evaluation Metrics

The rankers and rerankers were evaluated using a few metrics as below:

1) **Hits@4**
The proportion of queries with at least one relevant document in their top 4 retrieved documents.
2) **Hits@10**
The proportion of queries with at least one relevant document in their top 10 retrieved documents.
3) **Mean Average Precision (MAP) @ 10**
The mean average precision of relevant documents at each top 10 retrieved documents for all queries [11, 12]. The formula is $AP = \frac{1}{R} \sum I(k) P@k$ where:
   - R is the total number of relevant documents.
   - k is the rank position of a document.
   - I(k) is an indicator function for relevance ($1 \vee 0$) at k.
   - P@k is the precision at rank k.

The MAP can be calculated by dividing the AP by the total queries (2556 queries).

4) **Mean Reciprocal Rank (MRR) @ 10**

The mean of the reciprocal rank of the first relevant document in the top 10 retrieved documents of all queries. A high Reciprocal Rank of a query means that the first relevant document is highly ranked.

The formula is $RR = \sum \frac{1}{k}$, where k is the rank of which the first relevant document is found in the top 10 retrieved documents [11]. The MRR can be calculated by dividing the RR by the total queries (2556 queries).

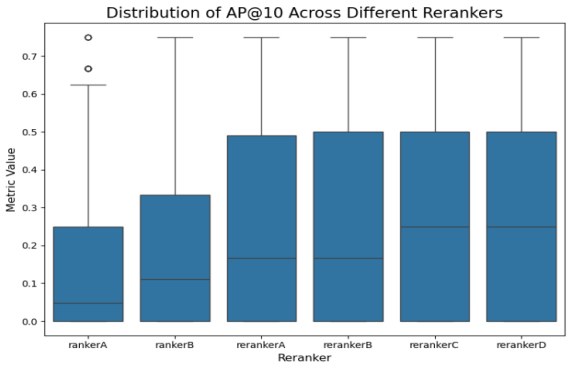
5) **R-Precision (R-Prec) @ 10**

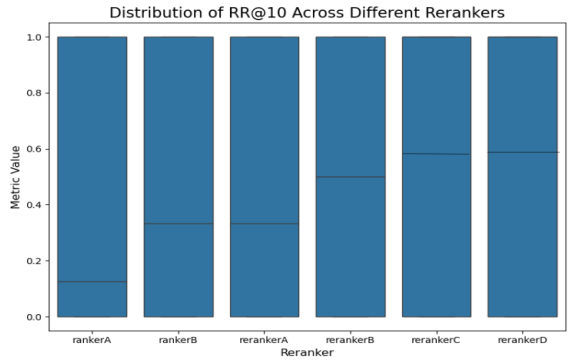
$Rprec = \frac{r}{R}$, where:

- r is the number of all relevant documents from the top 10 retrieved documents for a specific query.
- R represents the actual number of relevant documents available for a query.

Suppose there are 8 relevant documents to a query out of 609 documents in the corpus (R = 8), and the number of relevant documents retrieved in the top 10 is 4 (r = 4). The R-precision for the query will be 4/8 = 0.5 [12].

B. Retrieval Stage Results





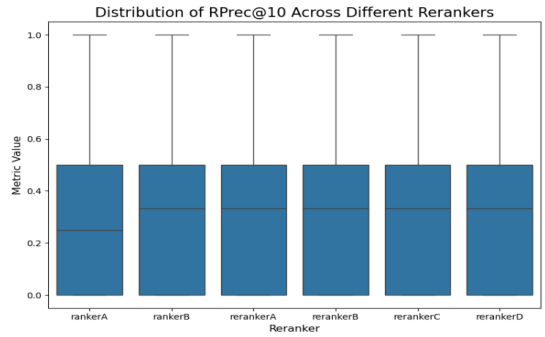


Fig. 1. The distribution of AP@10, RR@10, and Rprec@10 for the top 10 retrieved documents of the queries across different rankers and rerankers.

Figure 1 shows that Ranker B performed significantly better than Ranker A on Average Precision (AP) @ 10, Reciprocal Rank (RR) @ 10, and R-Precison (R-prec) @ 10 by having a higher mean. Thus, Ranker B (*BAAI/bge-large-en-v1.5*) will be used in the Generation Stage.

Since this paper aims to study whether the reranker has an impact on the performance of the RAG system, the best-performing reranker was also chosen based on the evaluation metrics. After analysis, the best reranker was Reranker D (*BAAI/bge-large-en-v1.5 with BAAI/bge-reranker-large*) which had the highest evaluation on all the used metrics.

Reranker C's performance is almost as good as Reranker D's, but it fell a bit short in all the evaluation metrics. Moreover, Reranker A and Reranker B are better than the two rankers but are not as good as Reranker C and Reranker D which uses the *BAAI/bge-large-en-v1.5* as the base retrieval model.

C. Generation Stage Evaluation Metrics

The BERT Score metrics were used as the basis for evaluating the generative performance of the RAG systems using Ranker B and Reranker D retrieval models on four different Large Language Models (LLMs).

Unlike traditional exact-string matches, the BERT Score provides the soft measure of contextual similarity between each token from the embeddings (vector) of the generated text, A and the gold-label text, B using cosine similarity where $cosine\ similarity = \frac{A.B}{||A||||B||}$. The BERT Score includes the calculation of Precision, Recall, and F1-Score [13].

1) **BERT Precision**

The BERT Precision measures the number of tokens in the *generated text* that closely matches tokens in the reference (gold-label) text. It is computed by calculating the cosine similarity for

each token embedding, t_g of generated text, G, to each token embedding, t_r in the reference text, R. Then, it greed matches them to maximize the matching similarity score [13, 14]. The average was then taken as BERT Precision which can be represented as:

$$Precision = \frac{1}{|G|}\sum_{tg\in G} \max_{tr\in R} \text{cosine similarity}(tg, tr)$$

2) **BERT Recall**

On the other hand, BERT Recall measures the number of tokens in the *reference (gold-label)* text that closely matches tokens in the generated text, which can be represented in the formula below [13, 14]:

$$Recall = \frac{1}{|R|}\sum_{tr\in R} \max_{tg\in G} \text{cosine similarity}(tr, tg)$$

Note the difference between the Recall and Precision formula. Figure 2 shows the illustration of calculating BERT Recall.

3) **BERT F1-Score**

The BERT F1-Score represents the harmonic means between the BERT precision and recall by the following formula [13, 14]:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

This study simply used the bert_score library to calculate the metrics[1].

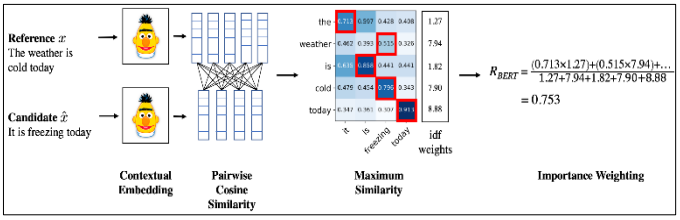


Fig. 2. Illustration of computation of the BERT Recall represented by R_{BERT}. Using the gold-label text (Reference, x) and the generated text (Candidate, $\hat{x}$) BERT embeddings and pairwise cosine similarity were computed. The greedy maximizing matches are represented by red boxes.

D. Generation Stage Results

1) **Efficiency in Terms of Time Taken**

The time taken to run each query for every combination of Ranker B and Reranker D with the four Large Language Models (LLM) was recorded[2] and the total time taken is depicted in Figure 3. Llama 3 proves to be the fastest LLM to generate outputs (around 45 minutes) for all the 2,665 queries followed by Mistral around 1.5 hours. Zephyr is the most inefficient LLM in terms of running time for the set of queries.

Additionally, Reranker D does not seem to play a significant role in reducing the total time taken to generate outputs in an RAG system when compared to Ranker B.

Moreover, figure 4 shows that Llama3 recorded the fastest time taken with around 1 second to generate text outputs for all the four question types. Meanwhile, Llama 2 had the worst performance in generating *comparison* and *temporal* query - more than 5 seconds. For the *inference* query, both Mistral and Zephyr recorded the worst average time taken with around 3 seconds. Finally, the slowest average time taken to answer a *null* query was more than 7 seconds when it was queried through Zephyr.

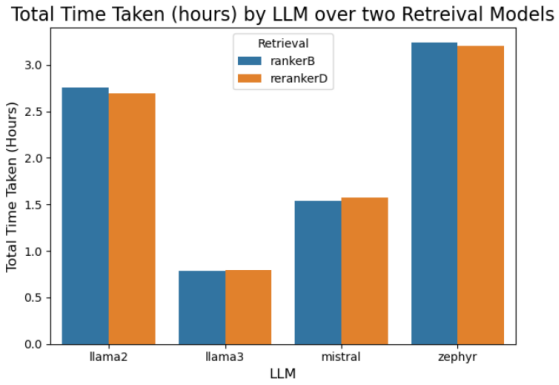


Fig. 3. Illustration of total run time for the 4 different LLMs to generate output based on the question that includes the top 10 retrieved documents for each query using Ranker B and Reranker D.

2) **Effectiveness in Terms of Bert Score**

Figure 5 shows the effectiveness of the generated output by the four LLMs by measuring the distribution of BERT Precision, Recall, and F1-score. Generally, Llama 3 recorded extraordinary average BERT scores for all three BERT metrics despite having a larger distribution. Additionally, the BERT F1-Score seems to agree with both BERT precision and recall, thus, the F1-Score metric will be used as the main metric to evaluate the effectiveness of the RAG systems.

Llama 2, Mistral, and Zephyr had around the same average BERT F1-Score. However, Llama 2 showed the worst distribution of the metric, ranging

[1] Available on https://github.com/Tiiiger/bert_score?tab=readme-ov-file

[2] The time taken to generate output per query for each LLM were recorded in the submission file of path s4705454/time/output/.

from 0.7 to 1.0, while having a lower mean than Llama 3.

Moreover, reranking the top 10 retrieved results from Ranker B using Reranker D showed less to no increased effectiveness of the RAG systems in terms of BERT scores.

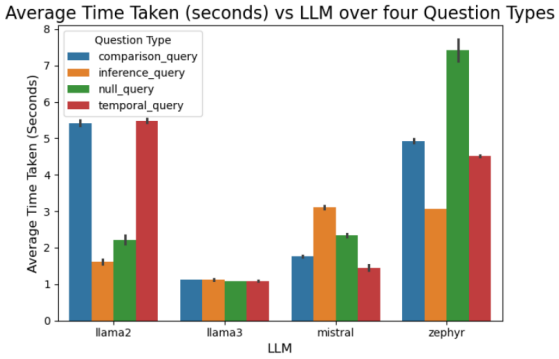


Fig. 4. Illustration of average time taken (seconds) by LLMs to generate outputs grouped by question type.

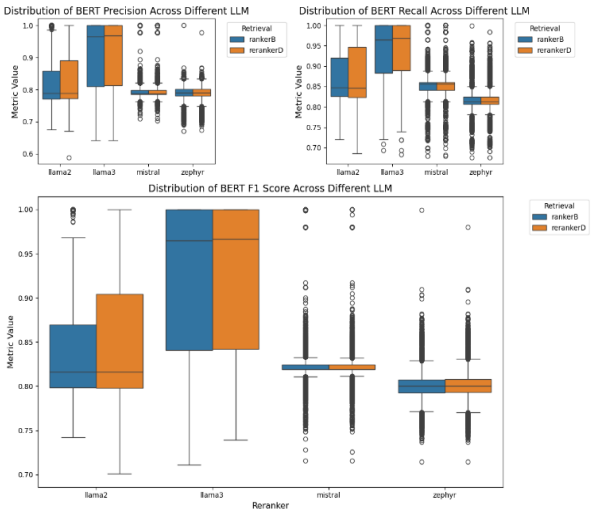


Fig. 5. Illustration describing the distribution of BERT Precision, Recall, and F1-score of the output generated across the four different LLMs using both Ranker B and Reranker D.

3) **Effectiveness and Efficiency of RAG Systems by Question Type**

The scatterplot in Figure 7 was generated to illustrate the average BERT scores of the generated outputs grouped by the question type from each of the 4 LLMs along with the time taken to generate them.

Llama 3 is the best LLM for the RAG system as it is both efficient and effective by being the fastest LLM to generate outputs that have high BERT scores. Llama 2 is the best in generating outputs for inference queries next to Llama 3. Meanwhile, Mistral performs better than Llama 2 for the other three question types. Finally, Zephyr still provides a good average BERT F1-Score but it was still ineffective and inefficient against RAG systems that used the other three LLMs.

Figure 7 also solidifies that the reranking, in this case, does not provide any significant improvement in efficiency and effectiveness as the same symbols of the same color in the scatterplot are not far apart from each other – they describe the same LLMs with Ranker B and Reranker D of the same question type.

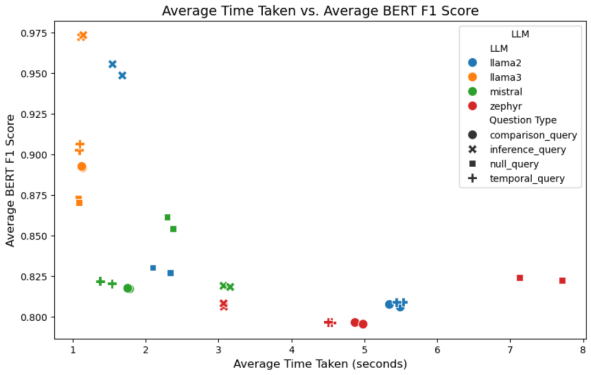


Fig. 6. Illustration describing the average time taken (seconds) for queries grouped by LLM and question type vs the average BERT F1-Score.

IV. Conclusions

All in all, *meta-llama/Llama-3.1-8B-Instruct* proved to be the most efficient and effective LLM to be used in an RAG system that uses Ranker B or Reranker D as the retrieval model.

Additionally, it is important to note that most of the intricate system interactions are independent of each other which may suggest that the best models in the retrieval phase may not yield the best outcomes in the generation phase. In comparing the results, it was discovered that Reranker D did not show any noticeable impact on effectiveness or efficiency compared to Ranker B, possibly due to both utilizing the same retrieval model (BAAI/bge- large-en-v1.5) which could already be highly proficient in retrieving relevant top 10 documents, thus diminishing the impact of reranking.

Using varied retrieval models for the ranker and reranker could lead to different outcomes and may display improvement in the RAG system's efficiency and effectiveness due to using a reranker. However, this change may be influenced by unidentified factors as the base retrieval model remains different. Additional research is required to further investigate the impact of rerankers on RAG systems.

REFERENCES

[1] P. Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 2021. [Online]. Available: https://arxiv.org/abs/2005.11401.

[2] A. Asai et al. Reliable, Adaptable, and Attributable Language Models with Retrieval. 2024. [Online]. Available: https://arxiv.org/abs/2403.03187.

[3] P. Zhang, S. Xiao, Z. Liu, Z. Dou, and J.-Y. Nie. Retrieve Anything To Augment Large Language Models. 2023. [Online]. Available: https://arxiv.org/abs/2310.07554.

[4] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie. C-Pack: Packed Resources For General Chinese Embeddings. 2024. [Online]. Available: https://arxiv.org/abs/2309.07597.

[5] Using LLMs - LlamaIndex — docs.llamaindex.ai. https://docs.llamaindex.ai/en/stable/module_guides/models/llms/#concept. [Online]. Available: https://docs.llamaindex.ai/en/stable/module_guides/models/llms/#concept.

[6] meta-llama/Llama-2-7b-chat-hf · Hugging Face. https://huggingface.co/meta-llama/Llama-2-7b-chat-hf#model-details. [Online]. Available: https://huggingface.co/meta-llama/Llama-2-7b-chat-hf#model-details.

[7] meta-llama/Meta-Llama-3-8B-Instruct · Hugging Face. https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct. [Online]. Available: https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct.

[8] mistralai/Mistral-7B-Instruct-v0.1 · Hugging Face. https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1. [Online]. Available: https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1.

[9] A. Q. Jiang et al. Mistral 7B. 2023. [Online]. Available: https://arxiv.org/abs/2310.06825.

[10] HuggingFaceH4/zephyr-7b-alpha · Hugging Face. https://huggingface.co/HuggingFaceH4/zephyr-7b-alpha#model-card-for-zephyr-7b-alpha. [Online]. Available: https://huggingface.co/HuggingFaceH4/zephyr-7b-alpha#model-card-for-zephyr-7b-alpha.

[11] J. S. Culpepper. COMP4703: Natural Language Processing with Python Week8: RAG and Evaluation. The University of Queensland, 2024.

[12] What is the difference between R precision and precision at K? https://cs.stackexchange.com/questions/67736/what-is-the-difference-between-r-precision-and-precision-at-k, 2017. [Online]. Available: https://cs.stackexchange.com/questions/67736/what-is-the-difference-between-r-precision-and-precision-at-k..

[13] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating Text Generation with BERT. 2020. [Online]. Available: https://arxiv.org/abs/1904.09675

[14] H. Özbolat. Text Summarization: How to Calculate BertScore. https://haticeozbolat17.medium.com/text-summarization-how-to-calculate-bertscore-771a51022964. [Online]. Available: https://haticeozbolat17.medium.com/text-summarization-how-to-calculate-bertscore-771a51022964