Member-only story

Survey on Retrieval Augmented Generation (RAG)

RAG development so far and the future

Kyosuke Morita · Follow

Published in Towards AI · 21 min read · Feb 29, 2024

Given the improvement of the Large Language Models (LLMs), the popularity of Gen AI is also increasing. Introducing a successful Gen AI tool has multiple advantages, such as reducing costs by having a chatbot rather than a human and increasing productivity. Retrieval Augmented Generation (RAG) system [1] is one of the most popular methods that can be applied in the real world. This method uses the pre-trained LLMs and injects external knowledge into them so that the LLM can answer something about the injected knowledge that was not available when it was trained but injected knowledge.

This article reviews existing literature to summarise the RAG developments and known challenges and explores their solutions.

Table of Contents

RAG overview

As briefly mentioned above, RAG is a method that enhances the ability of LLMs by letting LLMs access external vectorised data storage. This technique is especially effective for knowledge-intensive NLP tasks such as question-answering tasks.

The image below demonstrates the RAG proposed conceptual flow overview.

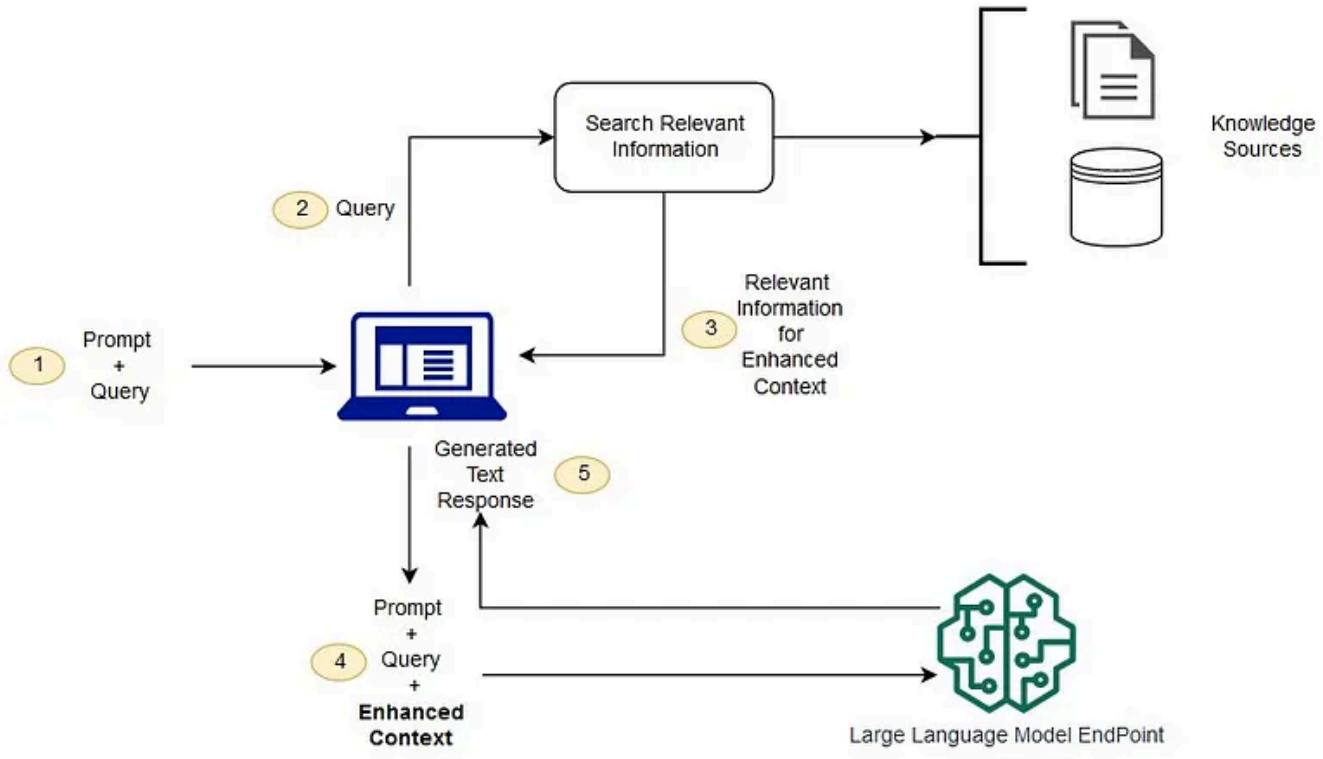


Image from https://aws.amazon.com/blogs/machine-learning/question-answering-using-retrieval-augmented-generation-with-foundation-models-in-amazon-sagemaker-jumpstart/

In the RAG system, there are 2 important components namely, a Retriever and a Generator. Given a Prompt/Query from an end user, a retriever searches the relevant information from the vectorised knowledge storage. The pipeline queries a generator (LLM) with the retrieved relevant context and receives the response text.

The main advantage of using RAG is that it can **generate more specific, diverse and factual language than solely querying the LLM** [1]. In addition, **the RAG requires much less computational cost comparison to fine-tune the LLM with the external information.**

The remainder of this post explores several different aspects of the RAG system. Firstly, the pros and cons of the RAG system compared to fine-tuning in terms of their performance. Then Investigate the evaluation frameworks and improvement of the RAG performance by implementing

different structures. Finally, it concludes with a discussion of the future developments of the RAG system.

Photo by Kevin Noble on Unsplash

RAG vs Fine-tuning

As briefly mentioned above, there are two common techniques to incorporate external knowledge into LLMs - RAG and fine-tuning. The main difference between those two methods is how to deal with the external data. The RAG system retrieves relevant information to the given query from a storage, hence the LLM itself still is untouched, on the other hand, the fine-tuning method incorporates those additional data by solving a downstream task so that the LLM is updated with specific additional knowledge. The RAG system requires fewer computational resources, yet the performance difference between those two methods is not well studied. This section

explores the pros and cons of those two techniques by going through two papers from Microsoft.

Ovadia et al. [2] aim to evaluate the performance of fine-tuning and a RAG on four STEM subjects and one humanity subject from Massively Multilingual Language Understanding Evaluation (MMLU) benchmark [3] by utilising LM-Evaluation-Harness [4] as an evaluation framework. The experiment compares three models: Llama2-7B [5], Mistral-7B [6] and Orca2-7B [7] with and without RAG and fine-tuning. Additionally, bge-large-en [8] was used as an embedding model for the RAG system and FAISS [9] was employed as its vector storage. In terms of fine-tuning, this experiment uses an unsupervised method, i.e., predicting the next token. The results of MMLU tasks demonstrate that the base models with the RAG system outperform fine-tuned models. The figure below shows the mean improvement from the plain base models on different MMLU tasks with 0 and 5 shots. For all the models, the base model with the RAG system exhibits significantly better results, especially, for Mistral 7B and Orca2 7B models, it is even better than combining with the fine-tuned models.

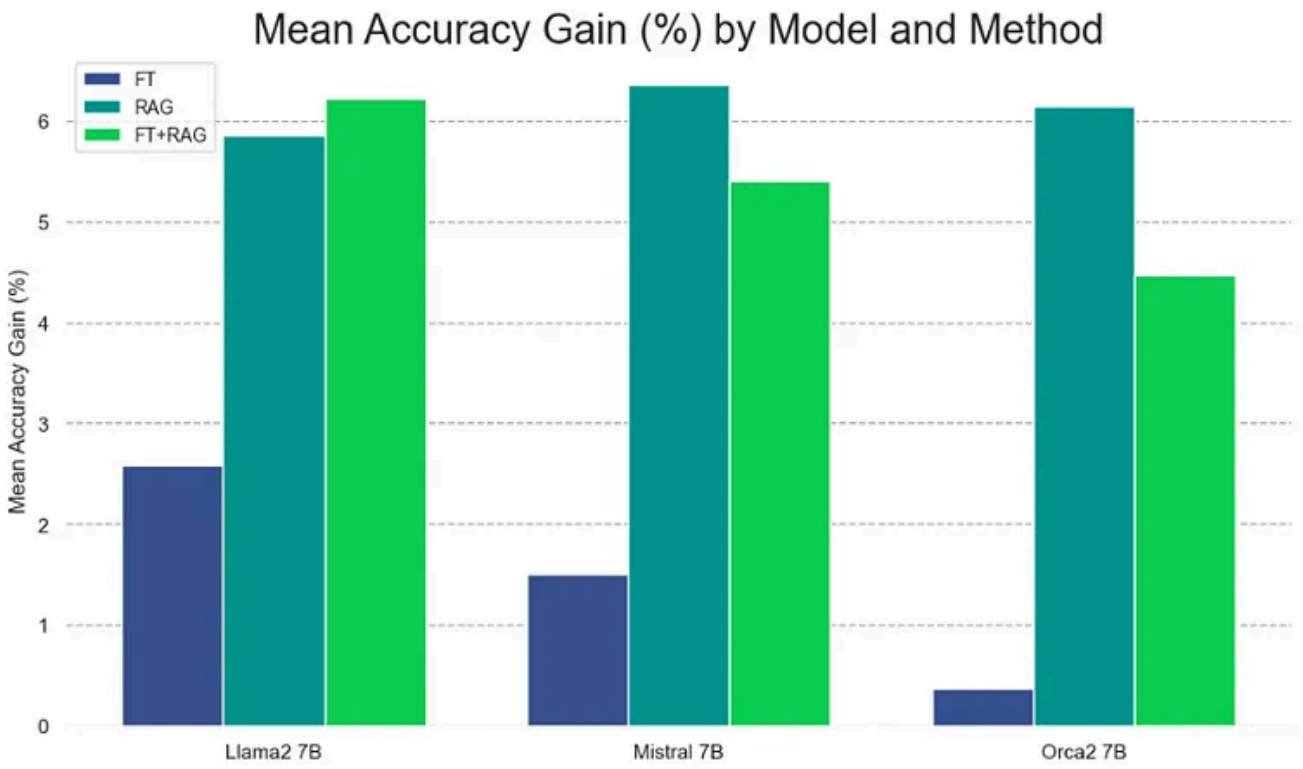


Figure 2. The relative accuracy gain (as explained in Equation (5)) for each knowledge-injection method, averaged (columnwise) across all experiments in Table 1.

From Ovadia et al. (2024) - Improvement result summary

The literature analyses that this observed behaviour might be caused by 1) the RAG system being able to access the additional context by a given prompt, 2) catastrophic forgetting [10] for fine-tuned models and 3) a capability of unsupervised fine-tuning comparison to other fine-tuning methods such as supervised learning and reinforcement learning - reinforcement learning from human feedback (RLHF).

Another team from Microsoft [11] conducted a study of a comprehensive comparison of the RAG and fine-tuning for question-answering tasks in the Agriculture industry. They created a RAG pipeline to generate a question-answer pair in the context of agriculture. They compared the performances

of popular LLMs such as Llama2-13B [5], Vicuna [12], GPT-3.5 and GPT-4 [13]. The RAG pipeline starts with data acquisition, which collects a diverse dataset in the industry domain from various high-quality repositories such as government agencies and scientific knowledge databases. The following process extracts structured documents from unstructured PDF files by using GROBID (GeneRation Of BIbliographic Data) [14], which is a machine learning package tailored for extracting data from the scientific literature. Then the pipeline generates question-and-answer pairs to generate contextually grounded and high-quality questions given the extracted content. Finally, the pipeline generates answers for the query with the given context. For the fine-tuning part, Low-Rank Adaptation (LoRA) [15] with generated question-and-answer pairs was used.

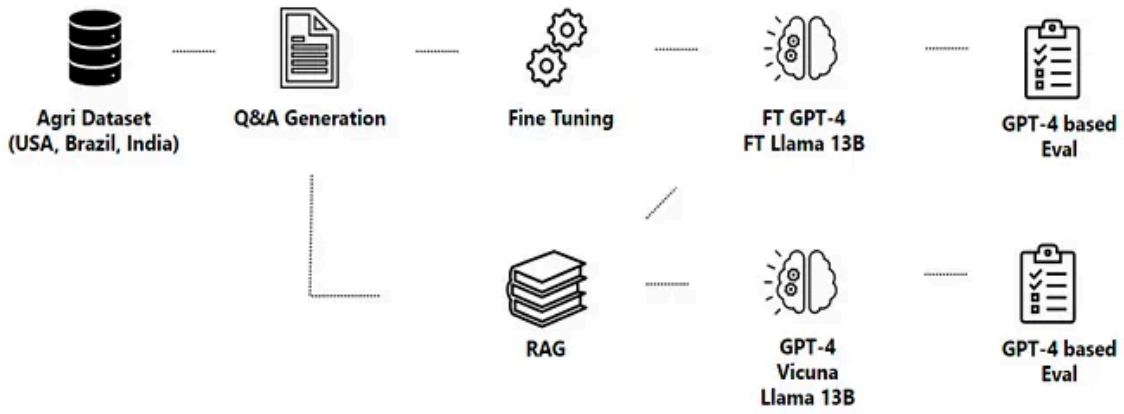
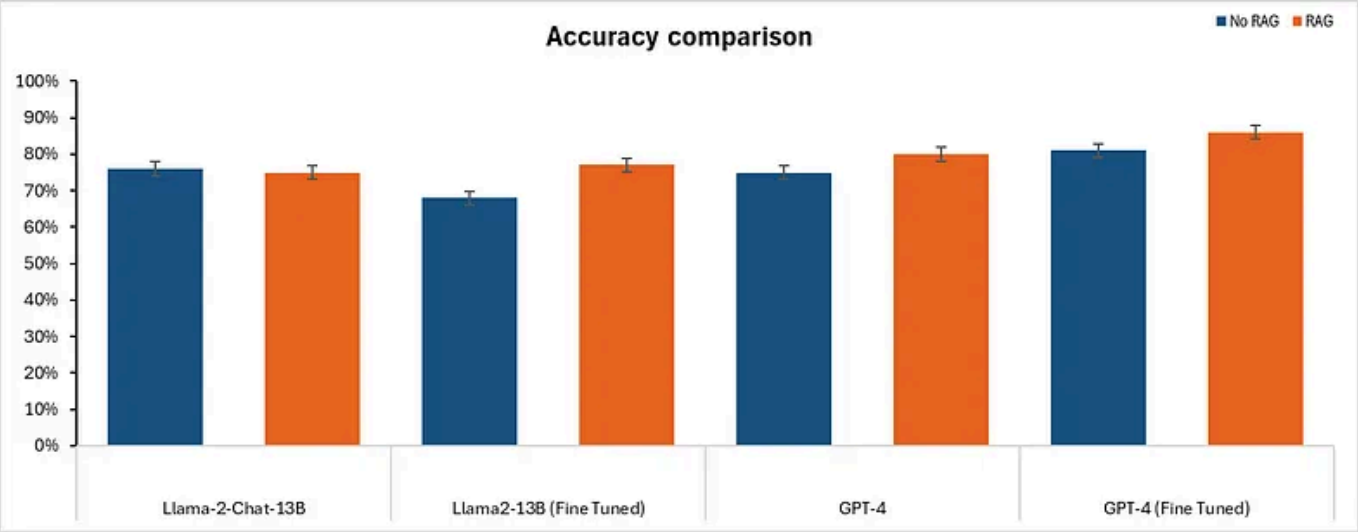


Figure 1: Methodology pipeline. Domain-specific datasets are collected, and the content and structure of the documents are extracted. This information is then fed to the Q&A generation step. Synthesized question-answer pairs are used to fine-tune the LLMs. Models are evaluated with and without RAG under different GPT-4-based metrics.

Pipeline flow chart (Balaguer et al., 2024)

The paper conducted several experiments — called *Evaluation with Guideline* (explained in more detail in the RAG evaluation section). This examines if the generated answer contains a “correct” answer. The “correct answer” guideline is generated by GPT-4 for each Q&A ground-truth pair as human evaluation is expensive and the correctness of technical question-answer

might be difficult for non-experts. The results are summarised in the figure below. It shows the accuracies of 2 LLMs, namely Llama-2-Chat-13B and GPT-4 with and without RAG pipeline and fine-tuning.



Redesigned results from Balaguer et al., 2024

Compared with or without the RAG pipeline, except for Llama-2-Chat-13B without a fine-tuning model, we could observe the accuracy improvement with the RAG system. Interestingly, this result shows that the Llama-2 model performs worse with either the RAG pipeline or fine-tuning but slightly improves its performance with both the RAG and fine-tuning, however, the differences are negligible. GPT-4 model is overall the best model of all in the experiment. A 5%-point improvement was confirmed by the RAG system (from 75% to 80%) but the performance with fine-tuning is slightly better (81%). Combining both RAG and fine-tuning achieves the best score of 86%.

This study demonstrated a few points:

- RAG system generally outperforms the LLMs
- Fine-tuning GPT-4 slightly outperforms GPT-4 with RAG but the difference is negligible

- Combining with both RAG and fine-tuning yields the best score
- It is worth noting that the cost of fine-tuning is high — a few days of training on multiple GPUs
- RAG is effective where data is contextually relevant (see the literature)
- Fine-tuning is effective in incorporating new skills in a specific domain into LLMs (see the literature)

In conclusion, the literature above demonstrated the significant performance improvement of knowledge-intensive tasks by incorporating the RAG system regardless of the subjects. The fine-tuning method also exhibited its performance improvement, yet the computational cost along with it is to be considered. Moreover, methods of fine-tuning such as (un)supervised learning and reinforcement learning, might impact the performance of the knowledge-intensive tasks, thus this needs to be studied in the future. Overall, given those experiments, implementing a RAG system over a fine-tuning method would be an effective first step considering the performance and its computational cost.

RAG evaluation

Since the RAG system comprises two primary components, our objective in terms of its evaluation also consists of both entities, namely evaluation of retrieval and evaluation of generation output. Having a solid evaluation process in the RAG pipeline is crucial as it identifies its capability in each component and helps to understand the capabilities and limitations of the system. This section explores standard metrics used in the literature and

their advantages and disadvantages for both components as well as several automated evaluation frameworks.

Evaluation of retriever

Evaluation of the retrieved documents aims to measure the effectiveness of the retriever. The retriever is a crucial part of the RAG system and its performance directory leads to the overall performance of the RAG system, especially for knowledge-intensive tasks. The evaluation of the retriever is often measured by Hit Rate, Mean Reciprocal Rank (MRR) and Normalised Document Cumulative Gain (NDCG) [16]. Although those metrics are suitable for measuring context relevance, those metrics simply compare the retrieved documents and the query. Yet, there is, as far as I am aware, no comprehensive evaluation framework that can assess the retriever performance and its effect on the overall RAG performance. Therefore it is important to compare the retriever performance and the overall RAG performance.

Evaluation of generation output

The evaluation of generation quality is a complex task. Often it involved investigating various aspects of generated text such as *context relevance*, *faithfulness* and *answer relevance* [17]. Several existing literature employ classical metrics such as F1, Precision, Recall, ROUGE [18], BLEU [19] and Exact Match (EM) scores to evaluate the general performance of the RAG [1, 20, 21]. Yet, to evaluate the quality of the generated answers, those traditional metrics would not be appropriate due to the characteristics of generated answers. For instance, simple word-matching metrics such as EM score evaluate the generated answers only compared against the reference answer, which is not suitable for evaluating the alignment of the model

response in the given context [22]. This section explores evaluation metrics of the RAG system for question-answer tasks.

Chen et al. [23] establish evaluation metrics called Retrieval-Augmented Generation Benchmark (RGB), In RGB framework, four key abilities need to be considered to evaluate the output quality, namely *noise robustness, negative rejection (unknown robustness), information integration*, and *counterfactual robustness* [23, 24]. More explanations of those abilities are listed below:

- *Noise Robustness* - evaluates the model's capacity to handle documents with noisy information related to a question but lacking substantial information
- *Negative rejection (unknown robustness)* - examines the model's ability to refrain from responding when the retrieved documents don't contain the necessary knowledge to answer a question also known as unknown robustness
- *Information Integration* - assesses the model's proficiency in synthesising information from multiple documents to address complex questions
- *Counterfactual Robustness* - tests the model's capability to recognise and dismiss known inaccuracies within documents, even when alerted to potential misinformation such as "Greek is the official language of the UK"

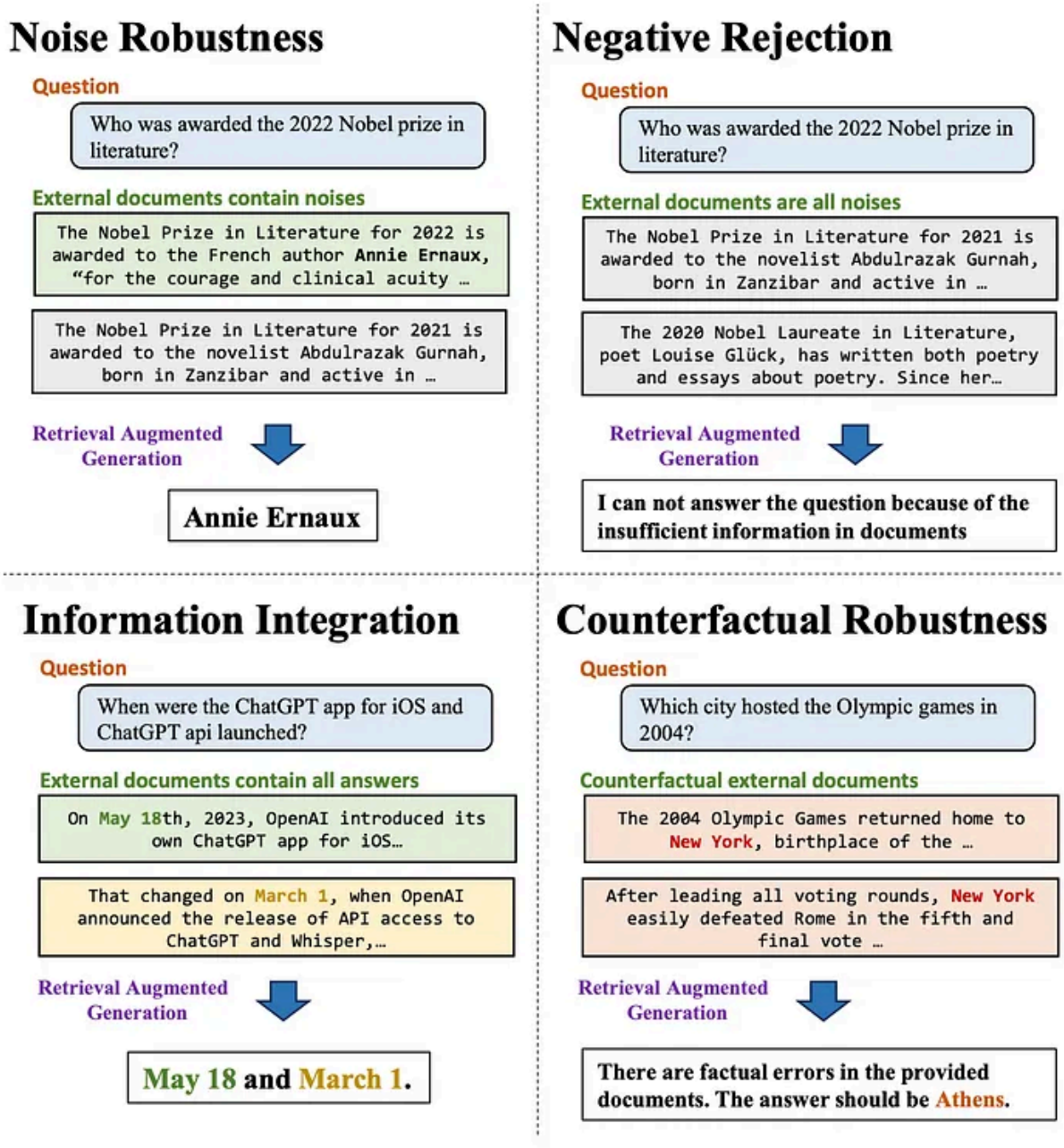


Figure 1: Illustration of 4 kinds of abilities required for retrieval-augmented generation of LLMs.

Chen et al. (2023) summarised 4 kinds of abilities for RAG evaluation

When evaluating the overall performance of the RAG output quality (answer faithfulness), answer relevance, negative rejection, information integration and counterfactual robustness are crucial for assessing overall performance. Gao et al. [25] summarised the evaluation metrics and their effectiveness in specific aspects of RAG evaluation in the table above. It is worth noting that

the listed metrics are all traditional and there needs to be a strong consensus or standard to measure the RAG output comprehensively.

Table 2: Summary of metrics applicable for evaluation aspects of RAG

	Context Relevance	Faithfulness	Answer Relevance	Noise Robustness	Negative Rejection	Information Integration	Counterfactual Robustness
Accuracy	✓	✓	✓	✓	✓	✓	✓
EM					✓		
Recall	✓						
Precision	✓			✓			
R-Rate							✓
Cosine Similarity			✓				
Hit Rate	✓						
MRR	✓						
NDCG	✓						

Gao et al., 2023 Summary of RAG evaluation metrics ability

Automated evaluation frameworks

Several studies propose a new evaluation method by employing LLMs as evaluators [26, 27]. Balaguer et al. [10] extended it to the RAG evaluation - utilised GPT-4 to evaluate the RAG output by implementing “Evaluation with Guideline”, which is a GPT-4 output that lists what the answer should contain. Given the strong performance of GPT-4, it seems to be a solid evaluation method for the RAG output evaluation.

Other studies propose automated evaluation frameworks for the RAG system [17, 28]. Automating the evaluation makes the RAG development faster and has a more robust judgement of the RAG capability.

Saad-Falcon et al. [28] proposed a method called *Automated RAG Evaluation System (ARES)*. This system evaluates the **context relevance**, **answer faithfulness** and **answer relevance** automatically by fine-tuned lightweight LLMs such as FLAN-T5 XXL [29] and DeBERTa-v3-Large [30] by synthetic

data. Furthermore, ARES uses a small size of human-annotated data to mitigate potential prediction errors. One of the advantages of this method is unlike other existing automated evaluation frameworks such as RAGAS [17], it does not rely on hard-coded static hand-written prompts, which limits their adaptability to various evaluation contexts. Moreover, this method provides statistical guarantees for its predictions by utilising prediction-powered inference (PPI) [31], which generates confidence intervals for its scoring. This method outperformed existing automated evaluation frameworks across context relevance and answer relevance evaluation accuracy on knowledge-intensive NLP tasks.

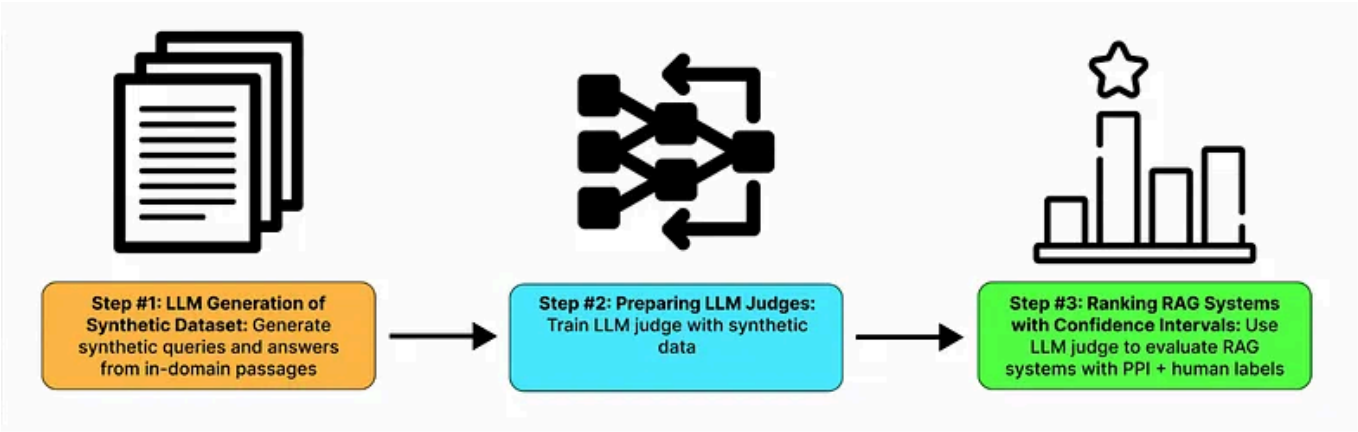


Figure 1: **Overview of ARES**: As inputs, the ARES pipeline requires an in-domain passage set, a human preference validation set of 150 annotated datapoints or more, and five few-shot examples of in-domain queries and answers, which are used for prompting LLMs in synthetic data generation. To prepare our LLM judges for evaluation, we first generate synthetic queries and answers from the corpus passages. Using our generated training triples and a constrastive learning framework, we fine-tune an LLM to classify query–passage–answer triples across three criteria: context relevance, answer faithfulness, and answer relevance. Finally, we use the LLM judge to evaluate RAG systems and generate confidence bounds for the ranking using PPI and the human preference validation set.

ARES overview (Saad-Falcon et al. (2023))

Although the development of automated evaluation frameworks suggests a solid theoretical foundation to be implemented in real life, there are several challenges such as the reliability of LLM-generated synthetic datasets and LLM-based evaluators. Those challenges can be studied in the future.

RAG performance improvement

Numerous methods to improve the RAG have been studied and this study is still rapidly developing. This section recaps the developments from the plain RAG system [1] to more recent studies that effectively improve the overall performance and its robustness to hallucination. More advanced RAG systems have been developed to address the drawbacks such as the poor retriever performance of the plain RAG system. To address such shortcomings, advanced RAG systems incorporate pre and post-retrieval processes to enhance the overall quality of the RAG system. This section investigates existing methods to improve the RAG system performance by exploring several established practical methods and existing literature.

Pre-Retrieval processes:

To improve the performance of the RAG system, there are several pre-retrieval process methods to consider. Conducting pre-retrieval processes makes the retriever perform better and often this should be the first thing to implement. There are primarily two techniques namely **document preprocessing** and **query transformation**.

Document preprocessing

Document preprocessing is a method that optimises the way to store documents. Since the performance of the RAG system hugely relies on the quality of the stored document, this process has a significant impact on the overall performance. Often documents that are to be extracted are in PDF files. There are several Python packages such as pypdf to do so, yet the document comes in various formats, for instance, some documents have multiple columns and images. Due to those diverse document formats and contents, extracting and loading the document is a tedious process. As

mentioned previous section, using tailored PDF extracting methods often improves the quality of text extraction [10] and this would lead to higher performance of the RAG system as well.

Also *adding metadata information* is effective for improving the retriever performance. Metadata here refers to the documents' information that describes their characteristics such as the title, author, date, etc. Intuitively, it helps the efficiency of the retriever as it can refer to the characteristics of the documents and can filter documents by leveraging that information. For example, given a query regarding the author of Harry Potter, the retriever can index the author's data, which is a more efficient retrieval and returns more relevant and precise information.

Chunking is also one of the popular methods to increase the retriever's performance. Chunking involves segmenting the text into smaller chunks. By chunking the document, the processing time of the retriever can be optimised. However, the size of a chunk is closely related to be performance of the retriever - if a chunk is not appropriate e.g. too big or too small or even cross-paragraph, it may harm the overall performance. The optimal length of a chunk varies depending on a specific task/topic, thus it is important to experiment with the chunk size.

Query Transformation

The retriever searches similar documents to a given query that is written by an end-user. However, in practice, not all the queries are written perfectly. Therefore, it is effective to optimise the query to shape it to be easily consumed by the retriever. Especially, the literacy of an end-user varies for publicly open chat-bots - often there are grammatical mistakes and typos in a query. To address this issue, literature suggests several methods.

Hypothetical Document Embeddings (HyDE) is a method that a given query, instructs a LLM to generate a hypothetical document, embed it and use it for document search [32]. Moreover, Ma et al. [33] suggests a framework called R*etrieve-Rewrite-Read.* This pipeline prompts an LLM to rewrite the query so that the query has better characteristics to be consumed by the retriever. Those methods improve the overall performance of the RAG system.

Post-Retrieval processes:

After the retriever gets top-k relevant documents, implementing the post-retrieval process can improve the generator's performance. There are two widely utilised methods namely, *Re-ranking* and *Information compression [16, 25]. Re-ranking* method re-orders the retrieved information to prioritise the most relevant content. This ensures that the generator is fed with the most relevant information, enhancing the quality and coherence of the generated output. *Information compression* aims to condense the retrieved documents in terms of length and information since it is believed that containing less noise in the retrieved documents achieves better generation performance.

Even though those two methods are effective and relatively easy to implement, recent studies show that several other methods can improve the overall performance of the RAG system more than traditional techniques. For instance, Yu et al. [34] introduced **Chain-of-Noting (CoN)** that improves the robustness of a RAG system when it faces noisy and irrelevant information. Given retrieved documents, CoN uses Llama-2 7B [5] to generate 3 kinds of notes, 1) CoN finds relevant information in the document and detects the answer, 2) CoN does not find the answer in the document but can deduce the answer by pre-trained knowledge and 3) CoN does not find the answer in the document and it also does not know the answer. This

method enhances the performance of the RAG system substantially and it reported that this also reduces factual hallucination.

A team from Microsoft [35] proposes a method called LongllmLingua, which uses LLMLingua [36] as a backbone framework for prompt compression. This method tackles the challenge of previous prompt compression methods such as LLMLingua [36] and Selective-Context [37, 38] that do not consider the context of the question during the compression and this may cause noises in the compression result. However, LongLLMLingua is a "question-aware coarse-to-fine" compression method, the compression process calculates the importance of each retrieved document (document-level perplexity conditioned on the different contexts to represent the association between them), thus the previous challenge is addressed and the overall performance was also improved.

Although noisy information in the retrieved documents is believed to make the overall performance of the RAG system worse [35], Cuconasu et al. [39] demonstrate contradictory results - the study found that including irrelevant documents in the right place could improve the performance. The cause of the improvement is still under investigation and further study to reveal this behaviour of the LLM and develop a new framework to exploit this finding in the future.

Retriever improvement:

A poor retriever is one of the causes of a poorly performed RAG system. Although you may have pre and post-retrieval processes, if the performance of the retriever itself is poor, the overall performance of the RAG system is still poor. There are a few common techniques that are often suggested such as Prompt engineering, Custom document chunks, and Embedding model

fine-tuning, [16, 25]. These methods are often tangible and incorporated in Python package frameworks like Llamaindex [16] and LangChain [40].

Recent literature suggests systematical improvement in retrieving processes. Asai et al. [41] proposed a method called *Self-Reflective RAG (Self-RAG)*, which incorporates a fine-tuned LM to "reflect" the outputs. The overview of the suggested architecture is shown in Figure 1 below. The architecture follows 3 steps: 1) determines if augmenting the continued generation with retrieved documents would be helpful, 2) outputs a retrieval token and concurrently processes multiple retrieved passages, assessing their relevance and generating the outputs and 3) generates critique tokens to reflect its outputs and select the best one.

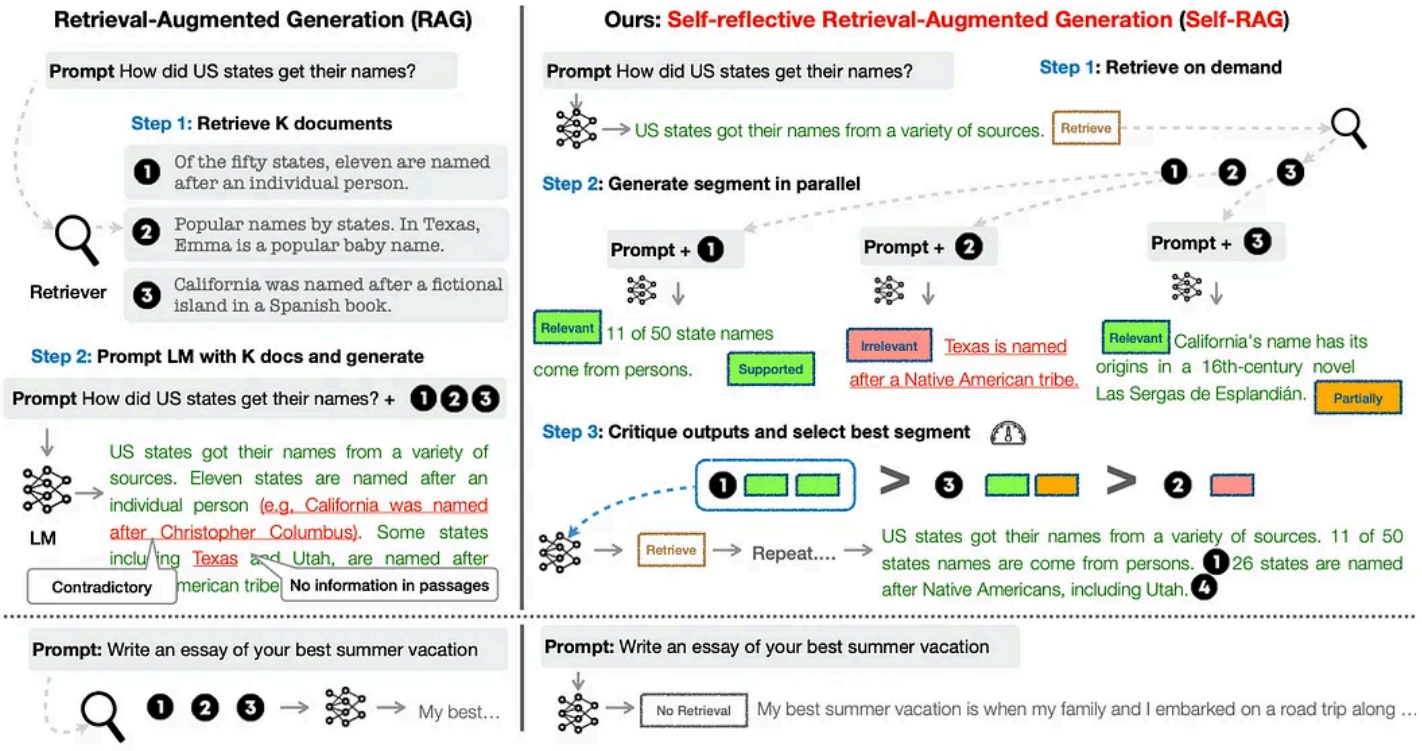


Figure 1: Overview of SELF-RAG. SELF-RAG learns to retrieve, critique, and generate text passages to enhance overall generation quality, factuality, and verifiability.

Self-RAG overview (Asai et al., 2023)

The experiment results demonstrate that Self-RAG often generates precise and shorter answers than SOTA LLMs with the RAG system and outperforms them in terms of accuracy.

Yan et al. [42] suggested a framework, Corrective RAG (CRAG) to improve the robustness of generation. CRAG embodies a lightweight LM retrieval evaluator such as T5-large [43] to evaluate the quality of retrieved documents for a query and return a confidence degree {Correct, Incorrect, Ambiguous}. Also, this framework includes a large-scale web search when the corpora are limited and cannot find the optimal documents as a further augmentation.

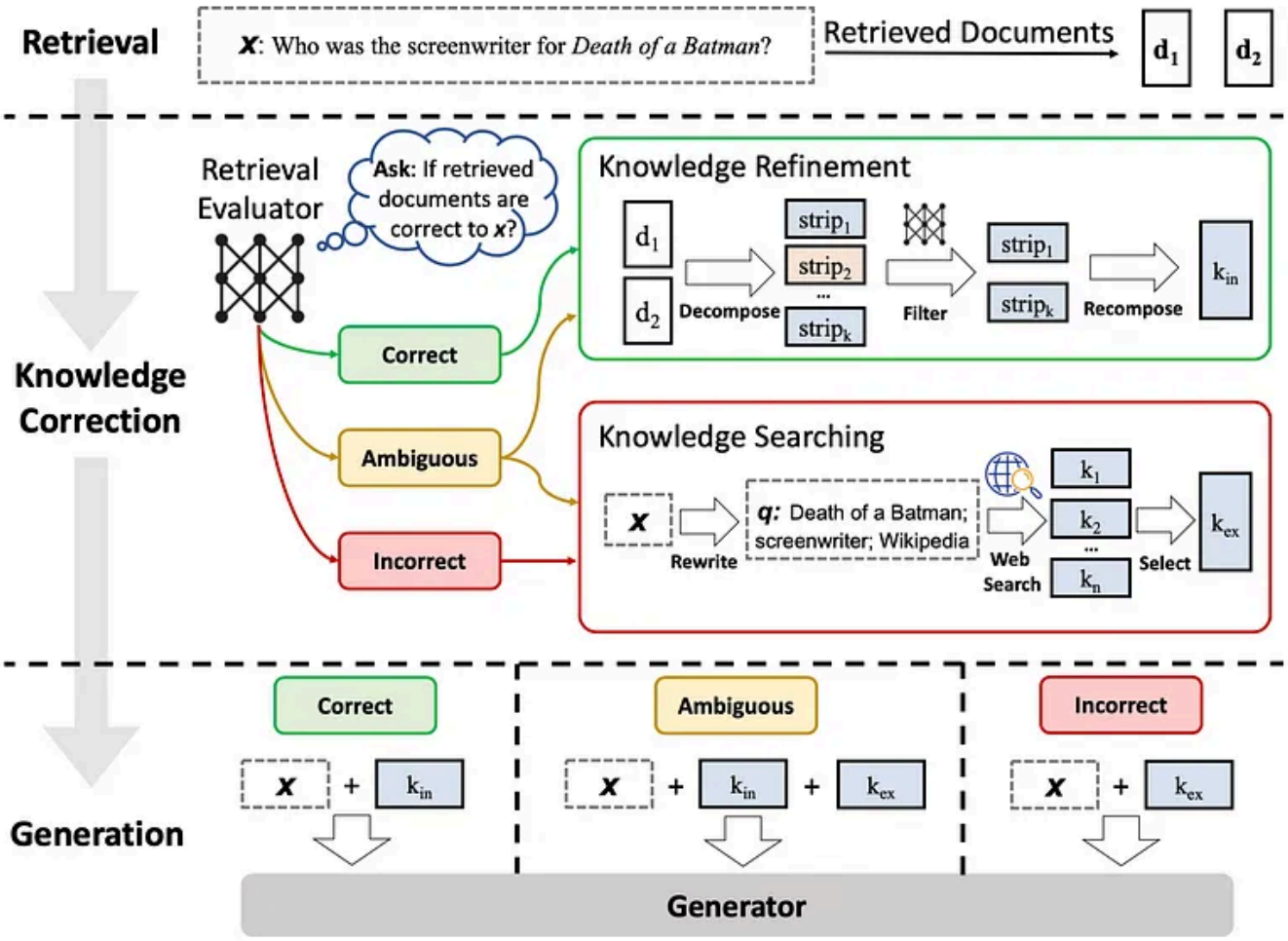


Figure 2: An overview of CRAG at inference. A retrieval evaluator is constructed to evaluate the relevance of the retrieved documents to the input, and estimate a confidence degree based on which different knowledge retrieval actions of {Correct, Incorrect, Ambiguous} can be triggered.

CRAG overview (Yan et al., 2024)

A few of the advantages of this method over existing frameworks are the evaluator is lightweight and the evaluator does not need any extra human or LLM annotation. Furthermore, CRAG is compatible with other existing methods like Self-RAG, thus those methods can be combined and achieve the SOTA performance.

Discussion

This post has covered the overview of the RAG system, its advantages and disadvantages compared to the fine-tuning method and recent developments focusing on evaluation and performance enhancement. Despite significant

development in the RAG system, there remains a need for a comprehensive automated evaluation framework and further exploration in terms of performance optimisation. Challenges such as handling irrelevant contexts and addressing hallucinations, which are common to LLMs, are being explored in existing literature [44, 42, 45, 1], yet they still persist as unresolved issues. Additionally, the scope of RAG research expands into the multimodal domain [46, 47]. Furthermore, from an end-user perspective, the importance of generating quick responses from the system cannot be compromised, as it significantly impacts the overall user experience. Therefore, further studies on the trade-off of answer accuracy and response time can be conducted [16, 41, 42].

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. K ¨uttler, M. Lewis, W.-t.
Yih, T. Rockt ¨aschel, et al., “Retrieval-augmented generation for knowledge-intensive nlp tasks,”
Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.
[2] O. Ovadia, M. Brief, M. Mishaeli, and O. Elisha, “Fine-tuning or retrieval? comparing knowledge
injection in llms,” arXiv preprint arXiv:2312.05934, 2023.
[3] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, “Measuring
massive multitask language understanding,” arXiv preprint arXiv:2009.03300, 2020.

[4] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell,
N. Muennighoff, et al., “A framework for few-shot language model evaluation,” Version v0. 0.1.
Sept, 2021.
[5] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand,
G. Lengyel, G. Lample, L. Saulnier, et al., “Mistral 7b,” arXiv preprint arXiv:2310.06825, 2023.
[6] A. Mitra, L. Del Corro, S. Mahajan, A. Codas, C. Simoes, S. Agarwal, X. Chen, A. Razdaibiedina,
E. Jones, K. Aggarwal, et al., “Orca 2: Teaching small language models how to reason,” arXiv
preprint arXiv:2311.11045, 2023.
[7] S. Xiao, Z. Liu, P. Zhang, and N. Muennighof, “C-pack: Packaged resources to advance general
chinese embedding,” arXiv preprint arXiv:2309.07597, 2023.
[8] J. Johnson, M. Douze, and H. J ´egou, “Billion-scale similarity search with gpus,” IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2019.
[9] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan,
J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., “Overcoming catastrophic forgetting in neural
networks,” Proceedings of the national academy of sciences, vol. 114, no. 13, pp. 3521–3526, 2017.
[10] A. Balaguer, V. Benara, R. L. de Freitas Cunha, R. d. M. Estev˜ao Filho, T. Hendry, D. Holstein,
J. Marsman, N. Mecklenburg, S. Malvar, L. O. Nunes, et al., “Rag vs fine-tuning: Pipelines,
tradeoffs, and a case study on agriculture,” arXiv e-prints, pp. arXiv–2401,

2024.
[11] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra,
P. Bhargava, S. Bhosale, et al., “Llama 2: Open foundation and fine-tuned chat models,” arXiv
preprint arXiv:2307.09288, 2023.
[12] W.-L. Chiang, Z. Li, Z. Lin, Y. Sheng, Z. Wu, H. Zhang, L. Zheng, S. Zhuang, Y. Zhuang, J. E.
Gonzalez, I. Stoica, and E. P. Xing, “Vicuna: An open-source chatbot impressing gpt-4 with 90%*
chatgpt quality,” March 2023.
[13] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt,
S. Altman, S. Anadkat, et al., “Gpt-4 technical report,” arXiv preprint arXiv:2303.08774, 2023.
[14] “Grobid.” https://github.com/kermitt2/grobid, 2008–2023.
[15] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora:
Low-rank adaptation of large language models,” arXiv preprint arXiv:2106.09685, 2021.
[16] J. Liu, “LlamaIndex,” 11 2022.
[17] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “Ragas: Automated evaluation of retrieval
augmented generation,” arXiv preprint arXiv:2309.15217, 2023.
[18] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in Text summarization
branches out, pp. 74–81, 2004.
[19] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of
machine translation,” in Proceedings of the 40th annual meeting of the

Association for Computational Linguistics, pp. 311–318, 2002.
[20] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite,
V. Karpukhin, J. Maillard, et al., “Kilt: a benchmark for knowledge intensive language tasks,”
arXiv preprint arXiv:2009.02252, 2020.
[21] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin,
S. Riedel, and E. Grave, “Few-shot learning with retrieval augmented language models,” arXiv
preprint arXiv:2208.03299, 2022.
[22] V. Adlakha, P. BehnamGhader, X. H. Lu, N. Meade, and S. Reddy, “Evaluating correctness and faithfulness of instruction-following models for question answering,” arXiv preprint
arXiv:2307.16877, 2023.
[23] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, “Retrieval-augmented generation for large language models: A survey,” arXiv preprint arXiv:2312.10997,
2023.
[24] J. Chen, H. Lin, X. Han, and L. Sun, “Benchmarking large language models in retrieval-augmented
generation,” arXiv preprint arXiv:2309.01431, 2023.
[25] Y. Liu, L. Huang, S. Li, S. Chen, H. Zhou, F. Meng, J. Zhou, and X. Sun, “Recall: A benchmark
for llms robustness against external counterfactual knowledge,” arXiv preprint arXiv:2311.08147,
2023.

[26] C.-H. Chiang and H.-y. Lee, “Can large language models be an alternative to human evaluations?,”
arXiv preprint arXiv:2305.01937, 2023.
[27] Y. Dubois, X. Li, R. Taori, T. Zhang, I. Gulrajani, J. Ba, C. Guestrin, P. Liang, and T. B.
Hashimoto, “Alpacafarm: A simulation framework for methods that learn from human feedback,”
arXiv preprint arXiv:2305.14387, 2023.
[28] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, “Ares: An automated evaluation frame-
work for retrieval-augmented generation systems,” arXiv preprint arXiv:2311.09476, 2023.
[29] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al., “Scaling instruction-finetuned language models,” arXiv preprint arXiv:2210.11416, 2022.
[30] P. He, J. Gao, and W. Chen, “Debertav3: Improving deberta using electra-style pre-training with
gradient-disentangled embedding sharing,” arXiv preprint arXiv:2111.09543, 2021.
[31] A. N. Angelopoulos, S. Bates, C. Fannjiang, M. I. Jordan, and T. Zrnic, “Prediction-powered
inference,” arXiv preprint arXiv:2301.09633, 2023.
[32] L. Gao, X. Ma, J. Lin, and J. Callan, “Precise zero-shot dense retrieval without relevance labels,”
arXiv preprint arXiv:2212.10496, 2022.
[33] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, “Query rewriting for retrieval-augmented large
language models,” arXiv preprint arXiv:2305.14283, 2023.
[34] W. Yu, H. Zhang, X. Pan, K. Ma, H. Wang, and D. Yu, “Chain-of-note: Enhancing robustness in retrieval-augmented language models,” arXiv

preprint arXiv:2311.09210, 2023.
[35] H. Jiang, Q. Wu, X. Luo, D. Li, C.-Y. Lin, Y. Yang, and L. Qiu, “Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression,” arXiv preprint arXiv:2310.06839, 2023.
[36] H. Jiang, Q. Wu, C.-Y. Lin, Y. Yang, and L. Qiu, “Llmlingua: Compressing prompts for accelerated inference of large language models,” arXiv preprint arXiv:2310.05736, 2023.
[37] R. Litman, O. Anschel, S. Tsiper, R. Litman, S. Mazor, and R. Manmatha, “Scatter: selective context attentional scene text recognizer,” in proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 11962–11972, 2020.
[38] Y. Li, “Unlocking context constraints of llms: Enhancing context efficiency of llms with self-information-based content filtering,” arXiv preprint arXiv:2304.12102, 2023.
[39] F. Cuconasu, G. Trappolini, F. Siciliano, S. Filice, C. Campagnano, Y. Maarek, N. Tonellotto, and F. Silvestri, “The power of noise: Redefining retrieval for rag systems,” arXiv preprint arXiv:2401.14887, 2024.
[40] H. Chase, “Langchain,” 10 2022.
[41] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, “Self-rag: Learning to retrieve, generate, and critique through self-reflection,” arXiv preprint arXiv:2310.11511, 2023.
[42] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, “Corrective retrieval augmented generation,” arXiv preprint arXiv:2401.15884, 2024.
[43] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu,

"Exploring the limits of transfer learning with a unified text-to-text transformer," The Journal
of Machine Learning Research, vol. 21, no. 1, pp. 5485–5551, 2020.
[44] O. Yoran, T. Wolfson, O. Ram, and J. Berant, "Making retrieval-augmented language models
robust to irrelevant context," 2023.
[45] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval augmentation reduces hallucination in conversation," arXiv preprint arXiv:2104.07567, 2021.
[46] M. Yasunaga, A. Aghajanyan, W. Shi, R. James, J. Leskovec, P. Liang, M. Lewis, L. Zettle-moyer, and W.-t. Yih, "Retrieval-augmented multimodal language modeling," arXiv preprint
arXiv:2211.12561, 2022.
[47] W. Chen, H. Hu, X. Chen, P. Verga, and W. W. Cohen, "Murag: Multimodal retrieval-augmented
generator for open question answering over images and text," arXiv preprint arXiv:2210.02928,
2022.

Retrieval Augmented | Llm | Llm Evaluation | Rags | Genai

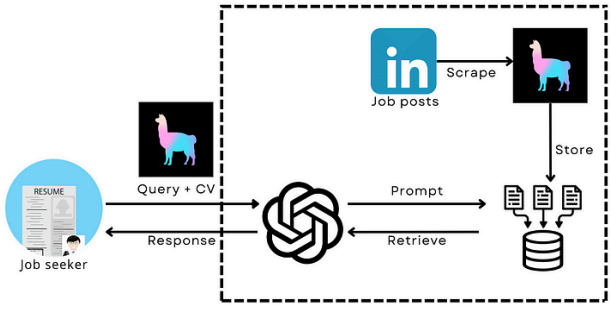
Written by Kyosuke Morita

246 Followers · Writer for Towards AI

Follow

Senior data scientist at a bank in London.

More from Kyosuke Morita and Towards AI



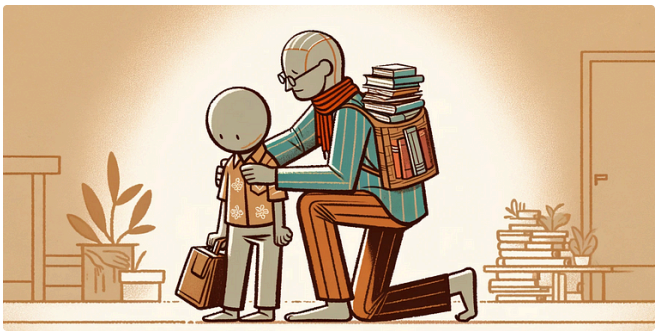
Kyosuke Morita in Towards AI

RAG-based Job Search Assistant

Use RAG to optimize your job search and job application

· 10 min read · Mar 5, 2024

182 1



Ignacio de Gregorio in Towards AI

RAG 2.0, Finally Getting RAG Right!

The Creators of RAG Present its Successor

· 9 min read · Apr 10, 2024

2.3K 16

Vatsal Saglani in Towards AI

Llama 3 + Groq is the AI Heaven

Llama 3 shines on Groq with blazing generation

· 9 min read · Apr 21, 2024

Kyosuke Morita

Counterfactual Explanations in ICML 2023

Paper Notes of Counterfactual Explanations in ICML 2023

11 min read · Nov 15, 2023

See all from Kyosuke Morita

See all from Towards AI

Recommended from Medium

Lars Wiik

Best Embedding Model 🌟 — OpenAI / Cohere / Google / E5 /...

An In-depth Comparison of Multilingual Embedding Models

12 min read · Apr 7, 2024

249 2

Paul Iusztin in Decoding ML

The 4 Advanced RAG Algorithms You Must Know to Implement

Implement from scratch 4 advanced RAG methods to optimize your retrieval and post-...

15 min read · May 4, 2024

1.2K 6

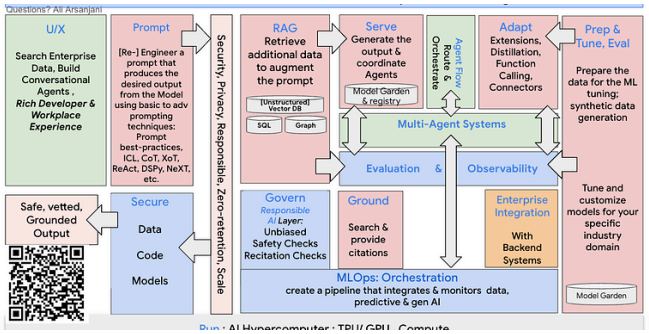
Lists

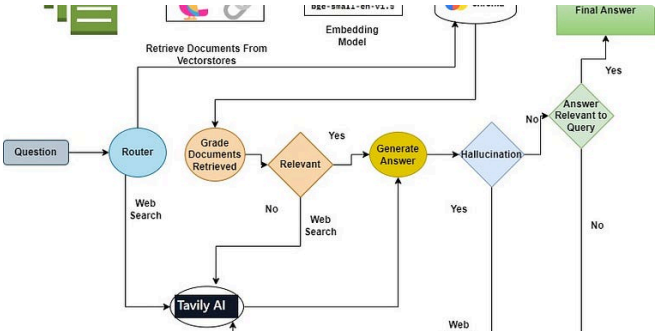
Natural Language Processing

1447 stories · 950 saves

ChatGPT prompts

47 stories · 1544 saves





Ali Arsanjani

The GenAI Reference Architecture

26 min read · Apr 28, 2024

592 6

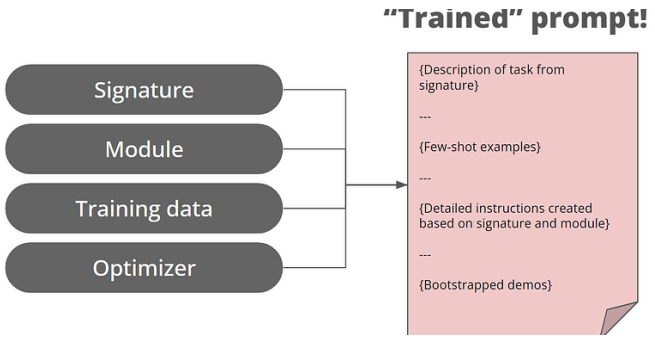
Plaban Nayak in The AI Forum

Build a Reliable RAG Agent using LangGraph

Introduction

13 min read · May 4, 2024

323 3



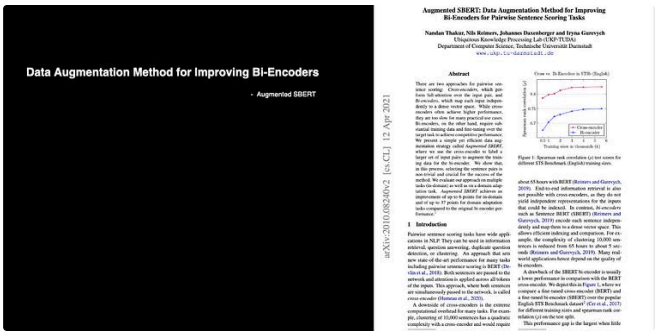
Julian Yip in Towards Data Science

Prompt Like a Data Scientist: Auto Prompt Optimization and Testing...

Applying machine learning methodology to prompt building

40 min read · May 5, 2024

553 5



Deltaaruna in Effectz.AI

Optimizing RAG, Fine-tuning Embedding and Reranking model...

1. Introduction

13 min read · Apr 16, 2024

136