2024 2nd International Conference on Foundation and Large Language Models (FLLM) | 979-8-3503-5479-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/FLLM63129.2024.10852482

RAGVAL: Automatic Dataset Creation and Evaluation for RAG Systems

Tristan Kenneweg*, Philip Kenneweg*, Barbara Hammer*
*Technische Fakultät, University of Bielefeld, Bielefeld, Germany
tkenneweg,pkenneweg,bhammer@techfak.uni-bielefeld.de

***Abstract*—Retrieval Augmented Generation (RAG) systems are widely used to augment Large Language Model (LLM) outputs with domain specific and time sensitive data. Each RAG system depends on a set of hyperparameters that strongly influence its performance. However, it is unclear how to evaluate different approaches on how to build and parameterize RAG setups for a given knowledge base. In this paper, we present a rigorous dataset creation and evaluation workflow that allows the quantitative comparison of different RAG strategies on a wide range of knowledge domains. Our method depends neither on human annotations nor handcrafted question sets and only requires the knowledge base the evaluated RAG system is given. We publish our code and exemplary dataset online.**

***Index Terms*—Retrieval Augmented Generation (RAG), Large-Language Models, Evaluation**

I. Introduction

Large Language Models have seen huge progress in recent years and are able to generate coherent text on a wide variety of topics. A popular example is GPT-4 [1], which is able to perform limited reasoning about the world [2]. However, LLMs lack domain specific and time sensitive data [3]. This is a problem for many real world applications: A chatbot that is used by a project manager needs to know about company internal data and a chatbot that is used by a sports fan needs to know about the latest sports results.

To solve this problem, Retrieval Augmented Generation (RAG) systems have been proposed. In a RAG system a LLM is augmented with a knowledge source that contains relevant data, usually a vector database. Depending on the user query relevant data is retrieved and injected into the LLM's context. This approach has been shown to be very effective [4]. RAG systems come in a variety of flavors and with different hyperparameters, the optimal choice of which is dependent on the domain and context of usage.

To find the optimal choice of RAG system and parameters for a given domain a quantitative evaluation of different RAG systems is needed. This is not a trivial task because in a typical usage scenario free-form text goes into a RAG system and free-form text comes out, which makes a quantitative measure of the given answer quality difficult.

In this work we show how to construct an LLM self evaluating scheme based on the work of [5] and [6], that leverages the data asymmetry between answer generating LLM and evaluating LLM to make meaningful quantitative evaluations. We construct our dataflow such that no human annotations are required and that it can be applied to arbitrary knowledge bases. We perform baseline tests and show exemplary usage on a knowledge base of 12,792 Wikipedia articles.

Our code and exemplary Wikipedia dataset are publicly available at https://github.com/TKenneweg/RAGVAL.

Acknowledgments. The authors were supported by SAIL. SAIL is funded by the Ministry of Culture and Science of the State of North Rhine-Westphalia under the grant no NW21-059A.

II. Related Work

Retrieval Augmented Generation was first proposed by Lewis et al. [4]. They enhanced a language model with a dense vector index of Wikipedia and outperformed multiple task-specific state-of-the-art systems.

RAG in the context of LLMs works by embedding text chunks into a vector space. Inside the vector space similar text chunks are close to each other and can be found using fast nearest neighbor search. The text chunks which are close in vector space to a given user query are injected into the LLM context. The LLM can use the chunk information to generate better and more factually grounded answers. In the following we highlight a few recent basic techniques and variations which have been proposed to improve RAG systems.

Lewis et al. used BERT [7] as an embedding model to embed sentences into a vector database. In 2022 OpenAI released Ada-002 [8] which is a state-of-the-art embedding model and has been shown to outperform other embedding models on multiple tasks such as text search, code search, sentence similarity and text classification.

A central element of RAG is chunking which refers to the process of splitting a text into multiple chunks, which subsequently get embedded into the vector database. Different chunking strategies have been proposed ranging from simple sentence chunking to more complex algorithms that rely on domain specific information. An often used generic form of chunking is recursive chunking, which recursively splits a long text into smaller chunks using a list of separators like newline characters until chunks of the required size are reached [9].

There are many methods to augmented RAG systems. Two of the most popular are Hyde retrieval [10] and RAG with guardrails [11]. Hyde Retrieval augments the retrieval process by querying a LLM to write a hypothetical document which contains the relevant data. This hypothetical document is then embedded and used to query the vector database. Since the hypothetical document is presumably closer in embedding

vector space to the relevant data than the user query this can lead to better retrieval results.

RAG with guardrails [11] checks if the embedding vector of a user query is within a certain region of the vector space. This region has been specified beforehand by embedding a set of example queries. Depending on the desired system behavior data retrieval can be rejected if the user query embedding falls within a part of the embedding space that is near to the example queries.

Automatic evaluation of LLM output has become an active subject of recent research, since manual labeling is infeasible in many cases and limits the iteration speed of research drastically. Liu et al. [5] propose G-EVAL, a method to automatically evaluate LLM output with regards to truthfulness, relevance and fluency. They use GPT-4 to evaluate different metrics of a LLM given answer and use different prompting techniques to achieve this. They compare the LLM evaluations to human generated labels and conclude that strong correlations exist. Lin et al. [6] propose LLM-EVAL, which utilizes a new prompting technique to generate scores for multiple dimensions like truthfulness and relevance in one prompt. These methods do not easily transfer to RAG systems, since the evaluating LLM needs to be aware of the ground truth information, which should be supplied by a RAG system under ideal circumstances.

The automatic generation of language datasets is a very new discipline that has emerged with the advent of LLMs. In [12] Zhuyun et al. show how to generate large amounts of synthetic data from few example data to create task specific retrievers. RAGAS [13] is a system that focuses on evaluating RAG systems regarding the match between generated answer and provided context, by defining three different metrics that are evaluated at the sentence and statement level. They also employ gpt-3.5 to generate questions about 50 Wikipedia articles, but do not supply a baseline evaluation of the generated questions. In contrast, we focus on the dataset creation process and provide an automatic evaluation method along the lines of G-EVAL [5] and LLM-EVAL [6], with specific adaptations to address the challenges of evaluating RAG systems.

III. Methods

A. Requirements

RAGVAL aims to be an automated dataset creation and evaluation workflow for RAG systems. We identify three key requirements:

1) The knowledge base of the RAG system which is to be evaluated should be the only input required. In many practical scenarios this is the only data that is available.
2) The generated datasets must be suitable to be automatically evaluated by an LLM.
3) The generated datasets must contain a limited or at least quantified amount of questions that can be successfully answered with LLM internal knowledge alone.

The third requirement arises because most public data sources are contained within the training set of modern LLMs. This results in successful answers to queries, although no relevant information was provided by the chosen RAG system, since the necessary knowledge is encoded in the LLMs internal weights. We demonstrate this effect for questions generated from Wikipedia articles in Section IV. While the ability to answer questions without RAG is pleasing in itself, it poses a challenge when evaluating the effectiveness of a RAG setup. We therefore need a dataset that contains no or a well quantified amount of questions about topics that are contained within the internal knowledge of the LLM that is used during RAG.

B. RAGVAL Workflow

In this section we describe the RAGVAL workflow. An overview of the RAGVAL workflow can be seen in Figure 1 and pseudocode is given in Algorithm 1.

Overview: The key idea of RAGVAL is to use the naturally occurring knowledge blocks in a given knowledge base and generate question about the content of those blocks. The generated questions, along with the knowledge blocks, and the RAG generated answers are subsequently passed to the evaluating LLM. A meaningful evaluation is possible because the given knowledge blocks contain all the necessary information to successfully answer the posed questions.

Step-by-step: Given a knowledge base, e.g. a set of technical documents or Wikipedia articles, the RAGVAL workflow starts with the generation of knowledge blocks. These knowledge are the natural elements of the knowledge base. Examples are a manual in the context of industrial machines or an article in the context of Wikipedia. We call the set of these knowledge blocks K_a. This set of knowledge blocks is used to create the vector database of the RAG system that is to be evaluated.

Optionally, it is possible to filter K_a in two stages to generate a set of knowledge blocks about facts that are not contained within the LLMs internal weights, before generating questions. This is useful if the baseline quality of answers without RAG system is already very good. See Section IV for details.

Firstly, we filter by metadata contained in the knowledge blocks (e.g. date) thus generating the set K_d. Depending on the dataset the metadata we filter by can differ. If the data source is publically available date is usually a good metadata filter, since anything with a publishing date before the LLMs cutoff point will probably be in the LLMs training dataset.

Secondly, we filter by prompting the answering LLM (e.g. GPT-4) to answer a question about a knowledge block that is indicative of the knowledge the LLM has about that block. In the case of technical documents we would ask if the exact part name is known, in the case of Wikipedia articles or sports results we ask if the majority of information in a knowledge block is about a topic that happened after the LLMs cutoff date. We use the OpenAI function calling API to generate reliable classifications. The resulting set of knowledge blocks

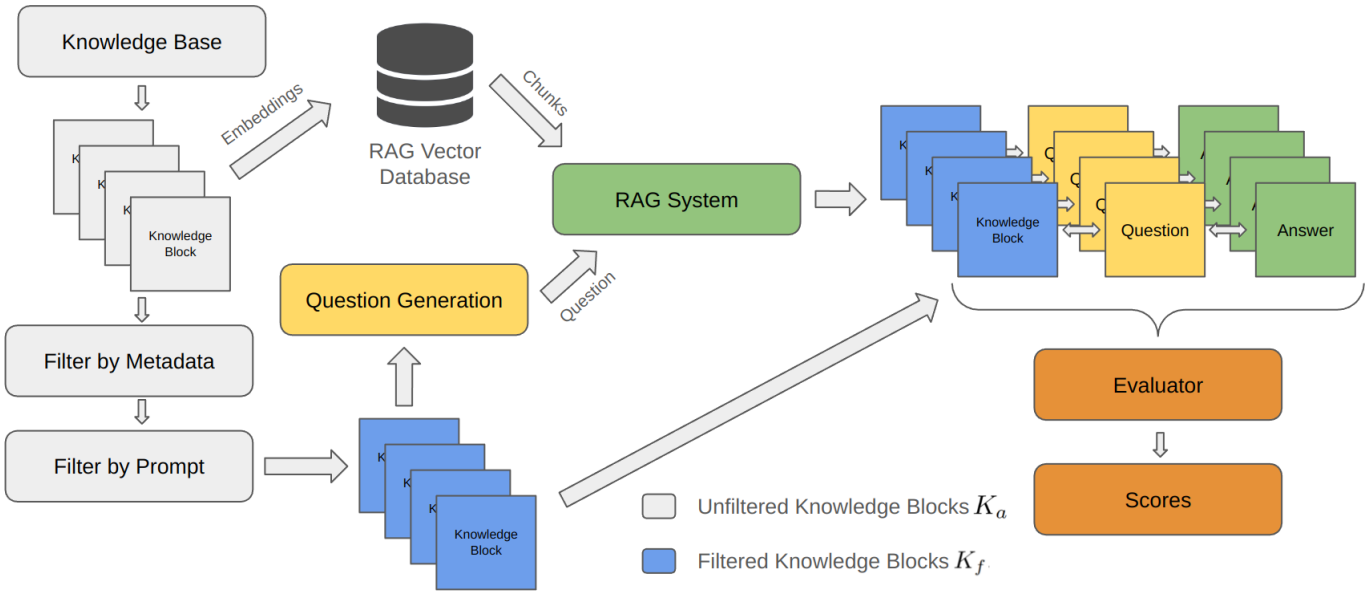


Fig. 1: Schematic overview of the proposed RAGVAL system.

K_f represents blocks of knowledge about facts that are not contained within the LLMs internal data.

For this set of knowledge blocks we generate questions using GPT-4 with a high temperature for maximum variability. This is possible with a simple question generation prompt:

Generate a creative question about the contents of the following text: {text}.

Given the questions we query the RAG system that is to be evaluated and store the generated answers. We do *not* provide the matching knowledge blocks. External information is only passed to the answering LLM by the RAG system whose design is not part of the RAGVAL workflow.

To automatically evaluate the generated answers we build upon the work of G-EVAL [5] and LLM-EVAL [6]. They use GPT-4 to evaluate metrics like truthfulness and relevance of a LLM given answer and use different prompting techniques to achieve this. We extend this approach to RAG systems by leveraging the information asymmetry between the answering LLM and the evaluator LLM. We pass the knowledge block which contains all relevant information to answer the question to the evaluator LLM along with the question and the RAG system generated answer. Note, that the LLM used in the RAG system can be arbitrarily chosen, while we strongly recommend to take the strongest LLM available to perform the evaluation. We generate two scores between 1 and 5 for relevance and truthfulness for each answer.
We set the general behavior of the evaluator in the system prompt:

Your task is to evaluate answers given by a chatbot. You are provided the user query, the chatbot generated answer and a knowledge block that contains information about the true answer. Given this information generate two scores from 1 to 5, where 5 is the best, for the chatbot generated answer, concerning relevance and truthfulness. Give a score of 1 for relevance if the answer is that the chatbot doesn't know.

For more details concerning the implementation refer to the publically available codebase.

Algorithm 1 RAGVAL Workflow Pseudocode

```
K_a=getKnowledgeBaseBlocks()
V_DB=createVectorDatabase(K_a)
for each b in K_a do
    if Metadatafilter(b) then
        K_d ← k
        if Promptfilter(b) then
            K_f ← b
        end if
    end if
end for
for each b in K_f do
    q=generateQuestion(b)
    a=answerWithRAG(q, V_DB)
    s=evaluateAnswer(q,a,b)
end for
```

IV. Evaluation

We test our workflow using Wikipedia articles. Following the RAGVAL workflow proposed in the previous section we:

1) Download $n_r = 12,792$ random Wikipedia articles we refer to as K_a.

TABLE I: Average truthfulness and relevance of GPT-4-0613 generated answers to questions about Wikipedia articles. K_a denotes 300 random articles, K_d denotes 300 articles that have been created after the knowledge cutoff date of GPT-4-0613, K_f denotes 256 articles that have been classified by GPT-4-0613 as mostly containing information about things after the knowledge cutoff date. K_fGT denotes the same articles as K_f but with the correct article supplied to the answerer. Note that 2 is the lowest possible score.

Metric	K_a	K_d	K_f	K_fGT
Truthfulness	3.59	2.62	2.48	**4.96**
Relevance	3.9	2.80	2.42	**4.96**
Sum	7.49	5.42	4.9	**9.92**

2) Filter for those that have been created after the knowledge cutoff date of the LLM that is tested. For GPT-4-0613 this cutoff date is September 2021, so we collect articles from October 2021 onward. This gives us the set of $n_d = 818$ articles K_d.
3) Use GPT-4 to classify wheter the information in an article is about a topic that happened after the cutoff date. We are left with a set K_f of $n_f = 256$ articles.
4) Generate one questions per article using GPT-4 with a high temperature for maximum variability.

For testing purposes we truncate K_a and K_d after the first 300 entries. We manually assess the quality of the generated questions and find that the majority of questions are of high quality. Some examples are:

- *What are the three venues across Oxfordshire, England from which Hinksey Sculling School operates, and which age groups train at each venue?*
- *What were the main criticisms of Manuchehr Kholiqnazarov's trial after his arrest in Tajikistan in 2022?*
- *What is the title of the short story from Philip Fracassi's 2021 collection, Beneath a Pale Sky, that was optioned for a feature film adaptation in 2022?*

A. Baseline Tests

We run baseline tests on K_a, K_d and K_f without a RAG system. During the baseline tests no additional information is supplied to the answering LLM. We also perform a ground truth test in which the correct knowledge blocks - in this case Wikipedia articles - are supplied to the answerer. We use GPT-4-0613 for all tests. Figure 2 and Table I show the results.

In Figure 2a we see that 178 out of 300 answers (59.3%) achieve perfect or near perfect scores (9/10 or 10/10) in a baseline test on questions generated about random Wikipedia articles. This clearly shows the phenomenon that knowledge bases can contain information that is encoded in an LLMs internal parameters. While this is pleasing in many circumstances, it makes the evaluation of different RAG systems challenging.

The baseline results are worst for K_f with an average score for truthfulness and relevance of 2.48 and 2.42 respectively, which shows that RAGVAL has successfully filtered knowledge blocks that are encoded in the LLMs internal parameters. Furthermore, we get near perfect results of 4.96 for truthfulness and relevance on K_f if we supply the correct article to the answerer, see Figure 2d. All generated answers are accessible in the publically available repository.

B. Baseline Test Analysis

It is notable that the answer quality for the date filtered knowledge block set K_d is better than might be expected. We investigated this further. Manual perusal of Wikipedia articles within K_d revealed that those articles contain a lot of information about events before the cutoff date. This is the case because while we can filter for creation date the contents of these articles are often about topics that are not recent. Some articles are for example about films which have been shot decades ago but only recently gotten an English Wikipedia entry.

Furthermore, we note that the drop in answer quality from K_d to K_f is also smaller than might be expected. The GPT powered filtering step reduces the number of articles in K_d from $n_d = 818$ to $n_f = 256$. Manual perusal confirms that the majority of information of the articles in K_f is about recent events. However, we find two question types that allow GPT-4-0613 to generate good answers despite its lack of knowledge. One is common sense questions. An example:

Question: What does Hayley Kiyoko mean when she says the "color palette" of her second studio album, Panorama, is "darker" than her debut album, Expectations?

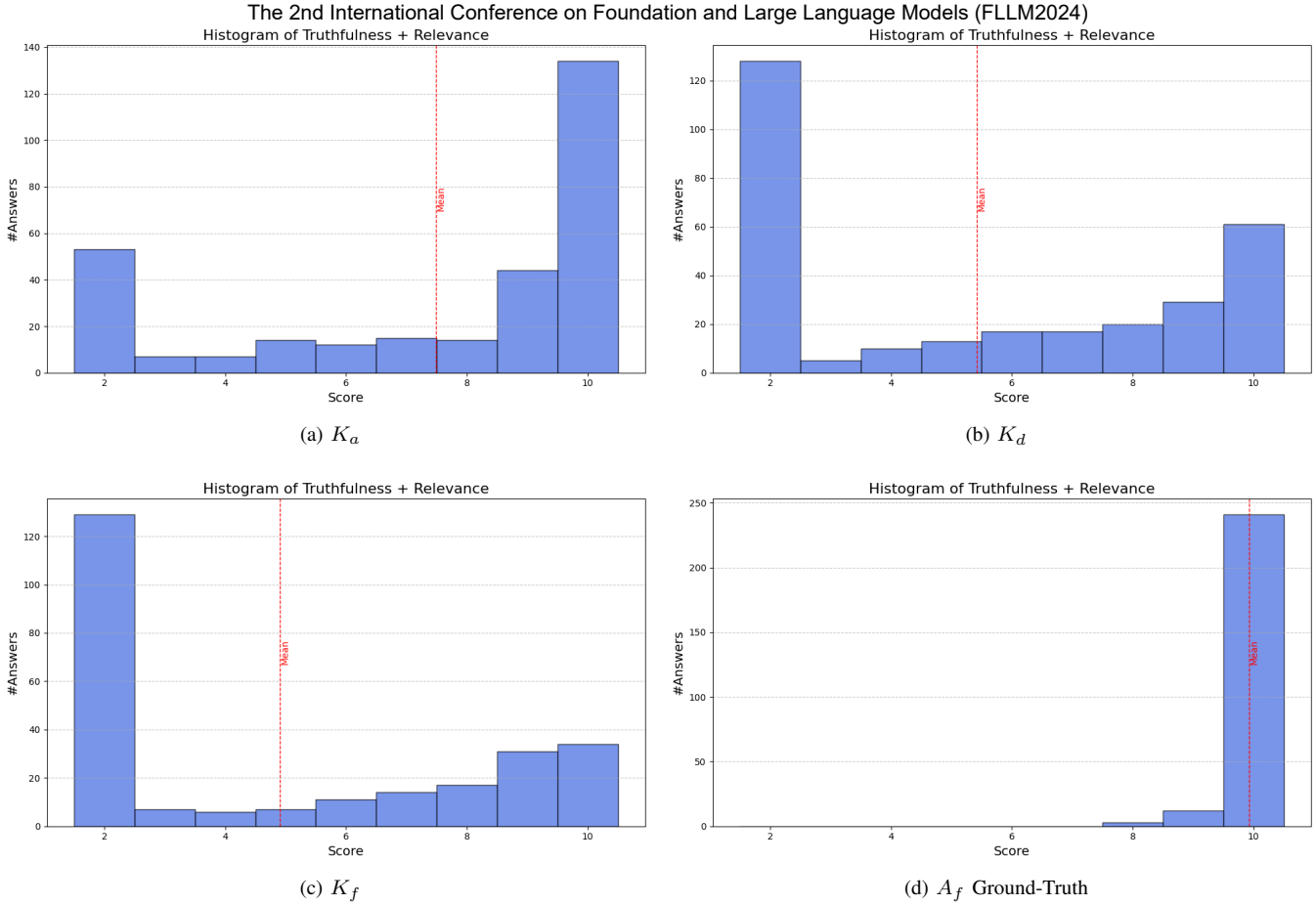
Even without knowing who Hayley Kiyoko is or what her albums are about, a good guess at an answer can be made. The other question type is about articles that contain mostly recent information but deal with topics that are not recent. An example:

Question: What role has social media played in the ZouXianZouxian phenomenon and the increase of Chinese migration through the US southern border?

The article related to this question contains mostly information about the ZouXianZouxian phenomenon in the times of Covid. However, the phenomenon itself is not recent and GPT-4-0613 can answer the question to satisfaction using its internal knowledge.

If a dataset is necessary which generates worse performance on baseline tests than K_f this could presumably be achieved by modifying the question generation prompt to focus on recent topics.

Since we build upon established automatic LLM driven evaluation methods and can find no contrary findings during manual sampling of the generated questions-answer-evaluation triplets, we are confident that the automated evaluation results



(a) K_a (b) K_d

(c) K_f (d) A_f Ground-Truth

Fig. 2: Sum of truthfulness and relevance for different baseline test setups. a) no RAG on 300 random articles K_a, b) no RAG on K_d, c) no RAG on all 256 articles from K_f, d) K_f with the correct article supplied to the answerer.

TABLE II: Chunk size experiments with a naive RAG setup on K_f. n_{in} and n_{out} denote the number of input and output tokens used during the answer generation process.

Metric / Chunk Size n_c	128	256	512	1024	2048
Truthfulness	3.86	4.56	**4.73**	4.67	4.64
Relevance	3.59	4.56	**4.7**	4.61	4.64
Sum	7.45	9.12	**9.43**	9.28	9.28
n_{in}	65.7k	94.2k	128.3k	225k	414.4k
n_{out}	12.8k	19k	21.2k	24.5k	26k
Sum	78.5k	113.2k	149.5k	265.5k	404.4k

are reliable. The distribution of the generated scores for the baseline and ground truth tests and different datasets confirms that the RAGVAL workflow works as intended.

V. Exemplary Usage

In this section we demonstrate the usage of RAGVAL to perform a hyperparameter optimization for a naive RAG system. In a naive RAG system the vector database is queried for every user input. Still many hyperparameters can be tuned, and can have different optimal values depending on the knowledge base. We focus on finding on optimal chunk size n_c and use the Wikipedia K_f dataset created in the previous section. We keep some basic parameters consistent between experiments:

- As embedding method we use OpenAIs Ada-002 model.
- As distance metric for the vector database we use cosine similarity.
- We return 5 chunks for each vector database query.
- We add the title of a Wikipedia page to every chunk that is generated from it to ensure that chunk information is not without context.

For chunking we use recursive character splitting with a variable target chunk size c_n and a maximum overlap of 48 characters. The larger the chunk size, the more information is contained in one chunk in the vector database. However, with larger chunks size token usage of the answer generating LLM increases in step. Larger chunk size may also hinder retrieval performance, since chunk embeddings are less granular.

We evaluate different chunk sizes with respect to answer quality and token usage. The results can be seen in Table II.

We see that chunk sizes of $n_c = 256$ and $n_c = 512$ characters are both good candidates. The RAG system performance decreases from $n_c = 512$ to $n_c = 256$ characters, while saving about 25% tokens. Depending on the cost of tokens and the requirements for reliability of a system, both hyperparameters are viable choices. However, we can clearly see that there is no

advantage in choosing a larger chunk size, since performance drops slightly while token usage increases significantly. On the extreme low end of 128 character chunks we can see a clear decrease in performance.

All numbers in this section were generated using the RAGVAL workflow applied to a Wikipedia data source. We have shown how RAGVAL can be used to evaluate different RAG system parameterizations. The same evaluation procedure can be extended to different RAG systems, using for example Hyde retrieval or RAG with guardrails.

VI. Limitations

RAGVAL is not applicable to scenarios in which the correct answer requires a comparison of different knowledge sources. An example is an online shop chatbot that needs to compare the prices of different products. In this case the correct answer is not contained within a single knowledge block.

RAGVAL is also not suited to setups with questions that have more ambiguous answers. For example: *What would be a good present for my son that I can find in this online shop?* Multiple good answers exist and multiple knowledge blocks might fit this question.

VII. Conclusion

In this paper, we presented RAGVAL a dataset creation and evaluation workflow specifically tailored for the automated evaluation of Retrieval Augmented Generation systems. Our workflow allows for the quantitative evaluation of different RAG systems and parameterizations without the need for human annotations during evaluation or handcrafted question sets. We also demonstrated and addressed the issue of questions about topics that are contained in an LLM's internal knowledge and can be answered without additional information, thus obscuring RAG system performance. We showcase the usage of RAGVAL on the exemplary task of hyperparameter tuning of a RAG system. We make our code and exemplary Wikipedia dataset publically available at https://github.com/TKenneweg/RAGVAL.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[2] M. Mitchell, A. B. Palmarini, and A. Moskvichev, "Comparing humans, gpt-4, and gpt-4v on abstraction and reasoning tasks," 2023.

[3] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," 2023.

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[5] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "Gpteval: Nlg evaluation using gpt-4 with better human alignment," *arXiv preprint arXiv:2303.16634*, 2023.

[6] Y.-T. Lin and Y.-N. Chen, "LLM-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models," in *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)*, Y.-N. Chen and A. Rastogi, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 47–58. [Online]. Available: https://aclanthology.org/2023.nlp4convai-1.5

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[8] OpenAI, "New and improved embedding model," https://openai.com/blog/new-and-improved-embedding-model, 2022, accessed: 2024-01-24.

[9] P. Finardi, L. Avila, R. Castaldoni, P. Gengo, C. Larcher, M. Piau, P. Costa, and V. Caridá, "The chronicles of rag: The retriever, the chunk and the generator," 2024.

[10] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," 2022.

[11] Pinecone, "Rag with guardrails," https://www.pinecone.io/learn/fast-retrieval-augmented-generation/, accessed: 2024-01-24.

[12] Z. Dai, V. Y. Zhao, J. Ma, Y. Luan, J. Ni, J. Lu, A. Bakalov, K. Guu, K. B. Hall, and M.-W. Chang, "Promptagator: Few-shot dense retrieval from 8 examples," 2022.

[13] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," 2023.