

Universiteit Leiden

Use RAG to improve RAG: evaluating RAG systems using real and synthetic data

Jesse van Haaster (s4485440)

Supervisors:
Peter van der Putten & Suzan Verberne

Master Thesis– Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

June 30, 2026

Abstract

Retrieval Augmented Generation (RAG) helps close the knowledge gap Large Language Models (LLMs) have in domains where little data is publicly available. They achieve this by retrieving documents relevant to the user query before answering the question. Since it is not trivial what the best RAG setup is, different setups need to be compared and evaluated in a structured manner to find out what setup works best. One way of evaluating RAG systems is by using question answer (QA) pairs relevant to the document collection, and manually evaluating the responses from the RAG. However, since manual evaluation takes a lot of time, this is often not possible. On top of this, sets of questions with reference answers are not always available. In this paper, we evaluate multiple methods for generating synthetic QA data in an attempt to generate meaningful synthetic QA pairs. We test the accuracy of the evaluation system by changing hyperparameters of the RAG in a controlled experiment with a predictable outcome, and checking if the expected results can be observed in practice. To assess the usefulness of the synthetic data, we evaluate various methods for generating synthetic QA data, and compare them against real QA pairs across multiple datasets using various metrics: ranking agreement, question validity, question difficulty and variance. Results show that our method is able to generate meaningful synthetic QA pairs which are both diverse in nature and difficult for the RAG to answer, however generated questions should still be checked by human annotators before being put to use.

Contents

1 Introduction **1**

2 Related Work **2**
- 2.1 RAG evaluation . . . 3
- 2.2 Multi-hop datasets . . . 3
- 2.3 Synthetic QA generation . . . 3
- 2.4 Synthetic QA evaluation . . . 4

3 Methodology **5**
- 3.1 Reference-based evaluation . . . 5
  - 3.1.1 ROUGE-L . . . 5
  - 3.1.2 METEOR . . . 6
  - 3.1.3 Context recall . . . 6
- 3.2 Reference-free evaluation . . . 6
  - 3.2.1 Faithfulness . . . 6
  - 3.2.2 Answer relevance . . . 6
- 3.3 Synthetic QA generation . . . 7
- 3.4 Synthetic QA evaluation . . . 9
  - 3.4.1 Ranking agreement . . . 9
  - 3.4.2 Question validity . . . 10
  - 3.4.3 Question difficulty . . . 11
  - 3.4.4 Diversity . . . 11

4 Experimental setup **12**
- 4.1 Corpora and reference data . . . 12
- 4.2 Testing the evaluation framework . . . 14
- 4.3 Synthetic QA evaluation . . . 15
  - 4.3.1 Ranking agreement . . . 15
  - 4.3.2 Question validity . . . 16
  - 4.3.3 Question difficulty . . . 16
  - 4.3.4 Diversity . . . 17

5 Results **17**
- 5.1 Evaluation framework test . . . 17
- 5.2 Synthetic QA evaluation . . . 18
  - 5.2.1 Ranking agreement . . . 18
  - 5.2.2 Question validity . . . 19
  - 5.2.3 Question difficulty . . . 20
  - 5.2.4 Diversity . . . 21

6 Discussion **23**
- 6.1 Limitations . . . 23
- 6.2 Implications . . . 24
- 6.3 Future work . . . 24

7 Conclusion 25

References 29

8 Appendix 30

1 Introduction

In recent years, a lot of research has gone into improving the performance of Large Language Models (LLMs), which has lead to them being able to answer general questions across many different domains, and allows them to answer specific user-tailored questions faster than regular search engines. The performance of LLMs is limited however by the contents of their training data. For this reason, they struggle in domains where little information is publicly available, such as in healthcare and enterprises. This makes them ineffective at answering domain specific questions. To circumvent this issue, many corporations make use of Retrieval Augmented Generation (RAG). RAG has proven to be an effective method for improving responses generated by LLMs. Especially in situations where domain-specific knowledge is required to answer questions, these systems help close the knowledge gap in the LLM [6, 27].[1]

RAG improves upon regular LLMs by first retrieving chunks of text relevant to the question posed by the user from a large document collection, and then passing these chunks to the LLM as context along with the user's question. This allows the LLM to answer questions about content which was not present in its training data and returns more grounded answers in specific domains. This leads to a decrease in the amount of hallucination in its answers [7].

To get the best answers it is therefore important to both build a retrieval system that retrieves the most relevant documents, and to use an LLM that generates the best answers in the RAG setup. Since the best RAG setup to use varies per use-case, it is essential to properly evaluate RAG systems on their performance to see where improvements can still be made. As human evaluation of RAG systems is very time-consuming, it becomes infeasible to do these evaluations by hand when dealing with large datasets and continuous iterations of new RAG setups. Furthermore, a simple comparison between the generated answer and the ground truth answer does not directly tell us what part of the RAG is limiting the performance, as the issue could stem from either the retrieval system or the LLM.

To help speed up the process of evaluating RAG systems, various automated evaluation metrics for RAG have been proposed. These methods often score the responses by comparing the generated answer to the ground truth answer. Some methods choose to evaluate answers using keyword based similarity scores, such as ROUGE [10] or METEOR [3]. Others make use of semantic similarity scores, such as BERTscore [26]. These methods are easy to implement, and give an insight to how well the RAG performs, but still show little insight into which part of the RAG setup is failing.

More recently, frameworks have been proposed that make use of LLM-as-a-judge capabilities, where LLMs are tasked with evaluating a given answer provided some context [5, 15]. These models are time-efficient, can provide explainable answers, and achieve the same level of agreement as human judges [28]. By scoring RAG results on multiple metrics, it becomes possible to diagnose more directly which part of the RAG is causing issues. In our experiments, we will use one of these frameworks, called RAGAS.

Despite their usefulness, these frameworks generally require a broad ground-truth dataset with QA pairs covering the domain, which can be hard and time-consuming to create. Even when it is available, keeping the dataset up-to-date in an ever-increasing document collection is very time consuming. In case such a dataset is not available, it is possible to generate synthetic QA datasets. Many methods have already been proposed for generating synthetic QA data, however most works

[1] This thesis was written in collaboration with the AdData team at DPG Media, in an effort to improve the company's internal RAG system, and to automate the process of evaluation.

focus on the quantity rather than the quality of the QA data.

To address this issue, we propose MultiChunkQA, a method to generate meaningful synthetic data based on multiple related passages found in the knowledge-base. In this paper, we answer the following research questions:

- How similar are synthetic QA evaluation results compared to real QA results?
- How effective are synthetic methods at generating "edge-case" questions which are hard to answer for the RAG?

To answer these questions, we will first confirm whether automated frameworks return accurate returns by testing it on a real document collection. Next we will introduce our method for creating synthetic QA pairs. We will then evaluate the usefulness of the resulting QA pairs based on both quantitative and qualitative metrics. Results show that synthetic questions have high agreement with real questions in ranking different RAG setups, and that questions generated from multiple chunks manage to generate questions that are harder for the RAG to answer than questions generated by a single chunk.

In this work we make the following contributions:

- We provide a clear evaluation regarding the accuracy of LLM-based evaluation frameworks, and show that these methods return reliable results.
- We propose MultiChunkQA, a new method for generating synthetic data that combines information from related chunks to generate multi-hop QA pairs.
- We evaluate the synthetic data based on metrics such as question difficulty, validity and coverage. We show that our method outperforms a baseline model at generating hard questions.
- The code, data and results of this thesis can be found on Github and have been made public to the extent possible.[2]

The remainder of this thesis is structured as follows. In section 2 we first summarize other works related to RAG evaluation and generating synthetic data. In section 3 we explain the metrics which will be used. In section 4 we give a detailed explanation of the used datasets, and we explain how the experiments will be performed. Next in section 5 we explain what the results of the experiments are. Lastly, in sections 6 and 7 we explain the limitations of our work, future directions, and the main take-away of this paper.

2 Related Work

Related work for this paper can be split into four different sections. First, we will list existing RAG evaluation methods. Second, we explain datasets containing multi-hop questions, and what characteristics they have. We want these characteristics to be reflected by our synthetic data. Third, we mention existing methods for generating QA data, and how our method relates to them. Last, we explain available methods for evaluating synthetic QA data.

[2] https://github.com/JesseVanHaaster/Synthetic-QA-generation

2.1 RAG evaluation

To determine the performance of a RAG setup, it is important to properly evaluate both the retriever and the generator. As it is very time-consuming to evaluate systems by hand, it would be beneficial to use a framework that can automate this process. A very simple approach to evaluating answers can be achieved by comparing generated answers to the reference answers using either keyword based matching methods, such as ROUGE [10] or METEOR [3], or by calculating the semantic similarity, such as with BERTscore [26]. These methods however only return simple scores, which do not tell us exactly how well the RAG performs. Recently, several frameworks have started leveraging LLM-as-a-judge capabilities, which have proven to be very effective at scoring system performance [28]. These frameworks, such as RAGAS and ARES, describe a task to an LLM, which scores the responses of the RAG based on the given rules.

RAGAS [5] allows users to score the output on various metrics, giving users flexibility to choose which metrics are important. By automating output judgment with an LLM, the task becomes scalable for evaluating large QA datasets on every iteration of the RAG setup. RAGAS is further explained in section 3, as it will be used in this paper.

ARES [15] is an evaluation framework that first fine-tunes small LLM-as-a-judge models on synthetic data before using them to evaluate answers. The frameworks uses some of the same metrics as RAGAS, but by training the models, it is able to provide confidence intervals about the RAG ranking it returns, making the results more accurate. In this paper, RAGAS was chosen over ARES, as it is easier to implement, and does not require a training dataset before it can be used.

2.2 Multi-hop datasets

In this paper we combine knowledge from multiple sources to generate questions, also known as multi-hop questions. Other datasets already exist that contain multi-hop questions, such as HotpotQA [24]. HotpotQA is a crowd-sourced QA dataset based on Wikipedia articles. By using the Wikipedia link network they were able to create a large dataset of multi-hop questions where the sources shared a connection, making the resulting questions more realistic. Another multi-hop dataset is MuSiQue [20]. MuSiQue fuses two single-hop questions into reasoning chains, and guarantees the necessity of multi-hop reasoning by making sure the question cannot be properly answered without both sources. The question pairs that satisfied this condition were then rewritten into natural questions through crowd-sourcing, creating true multi-hop questions. In our research we attempt to generate QA pairs with similar characteristics by retrieving chunk pairs that are semantically similar to each other.

2.3 Synthetic QA generation

A lot of methods have been proposed for generating synthetic QA pairs. Most works use the same approach: choose a random document from the collection and generate a QA pair that can be answered by this passage. The methods differ however in how they generate these pairs.

Alberti et al. [1] train 3 separate BERT models: one to extract an answer from the passage, one to generate a question based on the answer and the passage, and one to check whether the generated question can be answered by the passage. This way, they immediately double-check whether the generated synthetic QA pairs are valid. They use their method to generate many

synthetic questions for domain adaptation.

Nogueira et al. [13] use a transformer model that generates 10 queries given a passage. The model is trained by providing the model with a passage, and making it predict what user queries are attached to that passage. This work does not focus on generating training data or hard questions, but on query expansion. By adding these queries to the metadata of a document, it becomes easier to find for a user asking a question.

More recently, researchers have started experimenting with using larger language models such as GPT and Gemini to generate synthetic data. A great advantage to using these LLMs for synthetic data generation is the ability to guide them towards better results by using system prompts and providing more context to the model. Some works focus on prompt engineering to generate well-structured synthetic data that follows the rules of the prompt[11, 23]. Others focus on improving generated QA pairs by using in-context learning inside the prompt [9, 18]. In our work, we combine the methods of prompt engineering and in-context learning, but we also provide multiple chunks as context for the LLM in an attempt to create better, more complex questions.

2.4 Synthetic QA evaluation

Most of the above mentioned methods focus on generating training data for ML models. Since ML models require a lot of training data, a lot of synthetic data needs to be generated. Therefore they ask the model to generate a lot of questions for each document. For this reason these works focus on the diversity of the generated data, as it would not be favorable if all questions related to a document are similar to each other. To calculate the diversity of the data, various metrics can be used. The simplest methods for measuring diversity are based on lexical similarity between texts, such as the N-gram frequency in questions [25]. This method measures how often a specific sequence of words occurs in a text. If questions often share N-gram sequences, they are likely very similar. Another lexical diversity method is distinct-n [8]. Instead of measuring the amount of overlapping N-grams, this method calculates the total ratio of unique unigrams and bigrams in the entire generated data. This shows whether the data is all based around the same set of words, or if some questions are actually different from others.

Lexical similarity keeps track of similar word structures, but does not capture the meaning of sentences. Wang et al. make use of the cosine similarity [22] between question embeddings to measure similarity. This way questions with different structures that share the same meaning are caught by the model. Zhu et al. [29] also use the embeddings space to calculate diversity, but they expand on this idea by creating a method called DCScore. In this method, each text embedding is treated as its own separate class, and the DCScore between two embeddings is the probability that a classifier puts the embeddings into the correct class. The larger this probability is, the more diverse the texts are.

Another method of evaluating the usefulness of synthetic data is by testing the performance of the downstream task that was trained using this data [25], giving a direct insight into the usefulness of the data.

These methods do not directly focus on the quality of the data, but do give insights into the usability. In our use-case however, the goal is to generate evaluation data instead of training data. Since less data is needed for evaluation than for training, the diversity of the questions is less important in our case. The pairs that *are* generated however, should be of similar quality as real questions. This means that the scope of the project shifts more towards the actual validity of

the data, where the related work focuses more on the diversity of the data, and the performance increase it brings in training models.

The work that most closely relates to our work comes from van Elburg et al. [21]. In their work, they test the usefulness of synthetic data for evaluating RAG models. Their work mostly focuses on finding out whether synthetic data is usable as evaluation data, by testing whether curated QA pairs and synthetic QA pairs return the same RAG rankings based on the performance of the RAG on a number of metrics. Our work reproduces this, and goes a step further by testing if we can increase the difficulty of the generated questions.

3 Methodology

This section will first go into detail about the metrics used for evaluation, both methods that need a reference answer and reference free methods. We will then go on to explain the methods used for the synthetic QA pair generation pipeline, and the metrics used to evaluate the synthetic QA data.

3.1 Reference-based evaluation

Reference-based metrics require all questions to have a ground truth answer, as these answers will be compared against the outputs from the RAG. It is the most direct form of evaluation, as we compare the returned response or context from the RAG directly to the actual answer.

3.1.1 ROUGE-L

ROUGE is a metric that measures the number of matching n-grams present in both the reference answer and the generated text [10]. This allows us to measure how similar the two strings are. ROUGE has many variants as to how it is calculated. In this work we choose to make use of the ROUGE-L variant. With ROUGE-L, we look for the Longest Common Subsequence (LCS) between two strings. LCS is defined as the length of the longest sequence of words that is shared by both texts. The words do not necessarily have to be consecutive to each other, but they do have to occur in the same order in both strings. The length of the LCS is then divided by the length of the reference answer (recall) and the length of the generated answer (precision). These scores are then taken to calculate the F1 score over LCS, resulting in the ROUGE-L score. The full equation can be seen in equation 1, where β is a hyperparameter used to control the importance of the precision and the recall.

$$
\begin{aligned}
Recall_{LCS} &= \frac{\text{LCS}(\text{Reference}, \text{Generated})}{m} \\
Precision_{LCS} &= \frac{\text{LCS}(\text{Reference}, \text{Generated})}{n} \\
ROUGE-L &= \frac{(1+\beta^2) Recall_{LCS} Precision_{LCS}}{Recall_{LCS} + \beta^2 Precision_{LCS}}
\end{aligned}
\tag{1}
$$

3.1.2 METEOR

METEOR, like ROUGE, matches n-grams present in both strings of text. However where ROUGE only counts exact word-matches, METEOR also accounts for words that share semantic similarity [3]. This makes METEOR more flexible to changing words in sentences that still carry the same meaning. This is very useful for LLM generated answers especially, as the generating of responses is a stochastic process, and outcomes can vary slightly each time.

3.1.3 Context recall

Context recall is one of the metrics used by the RAGAS framework. For the formalization, we will use the notation as described in the RAGAS paper. We consider a setting where given a question q and its ground truth answer $a_{truth}(q)$, the RAG retrieves some chunks $c(q)$ from the collection C which are then used to generate an answer $a_{gen}(q, c(q))$. The RAGAS source-code contains a variety of different evaluation metrics which can be found on their Github page, however in this paper we focus on 3 specific metrics: faithfulness, answer relevance, and context recall.[3]

Given the ground truth answer $a_{truth}(q)$ and the retrieved contexts $c(q)$, the context recall measures how many of the statements in $a_{truth}(q)$ are supported by $c(q)$. By splitting $a_{truth}(q)$ into statements and checking if each statement can find support in the context, the context recall score is calculated. As with the faithfulness metric, each statement is scored with either a 1 or a 0, and the final score is calculated with $CR = \frac{|G|}{|S|}$, where $|G|$ is the amount of statements grounded by the retrieved context. With the recall score, we can determine how much of the information necessary for the answer is present in the retrieved context.

3.2 Reference-free evaluation

Both of the following metrics are reference-free methods. It should be stated that these methods still require the reference question to generate a score, but are called reference-free, as they do not require a reference answer. Since reference answers are often the hardest to come by, these methods are more easily accessible.

3.2.1 Faithfulness

Given a generated answer $a_{gen}(q, c(q))$ and its supporting context $c(q)$, $a_{gen}(q, c(q))$ is deemed faithful to $c(q)$ if the statements made in the answer are all supported by $c(q)$. RAGAS measures this by splitting $a_{gen}(q, c(q))$ into separate statements, and comparing each statement to the context. Each statement is scored with either a 1 if it is faithful, or a 0 if it is not. The final faithfulness score is then calculated with $F = \frac{|V|}{|S|}$, where $|V|$ is the amount of correct scored statements and $|S|$ is the total number of statements. This metric tells us if the LLM stayed true to the provided context when answering the question, or if it hallucinated an answer.

3.2.2 Answer relevance

Given a generated answer $a_{gen}(q, c(q))$ and the original question q, an answer is deemed relevant to the question if it directly answers the question. To measure this, RAGAS attempts to transform

[3] https://github.com/vibrantlabsai/ragas

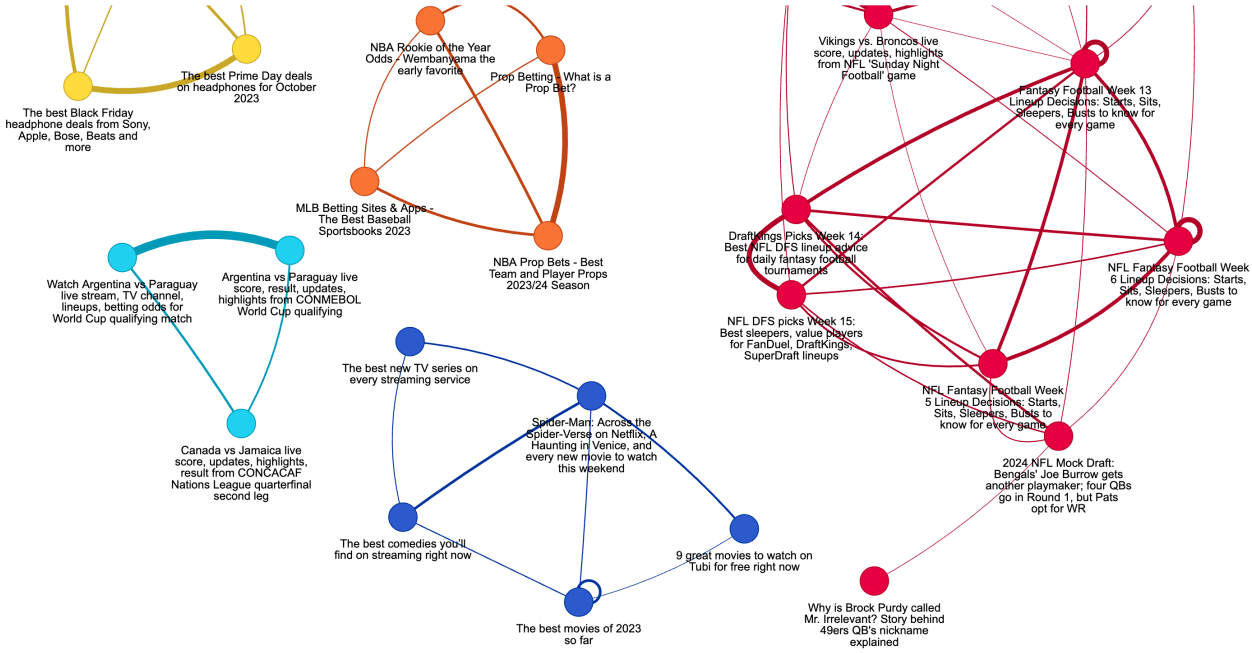


Figure 1: Figure showing a snippet of the document clusters we created. The clusters can be linked to various topics: TV/Movies (blue), Sports betting (orange), NFL (red), South American Football (light blue), and Black Friday deals (yellow).

$a_{gen}(q, c(q))$ back into q, by prompting an LLM to generate a question q_{gen} that could be answered by $a_{gen}(q, c(q))$. q_{gen} is then embedded in the same way as q. The final answer relevance score is calculated as the cosine similarity between the two questions: $\text{sim}(q, q_{gen})$. This metric captures how well the LLM understood the original question, and whether the resulting answer actually answers the posed question. Since the generation of example questions is a stochastic process, the resulting scores can change a lot per run. To account for this, we prompt the model to generate multiple questions and we average over the resulting scores, leading to more stable similarity scores. In this paper we choose to generate 3 different questions each time, as this is also the amount recommended by the authors of RAGAS. This metric does not use the retrieved context, but instead focuses on semantic similarity between the question and the answer. Instead of telling us if the correct knowledge was received, it tells us if the LLM understood the context and the question. This gives us an indication of whether the question was actually properly answered.

3.3 Synthetic QA generation

There are various ways of creating synthetic QA pairs [1, 17]. Recently a new method has started to gain popularity: using LLMs to generate QA pairs based on some context [12]. The idea is simple: we randomly select a chunk from the knowledge collection, and prompt an LLM to generate a QA pair that can be answered by that respective chunk. The method scales well, and can generate a lot of QA pairs relatively fast. This method may however lead to bad results, for a couple of reasons:

- The generated question is too simple.

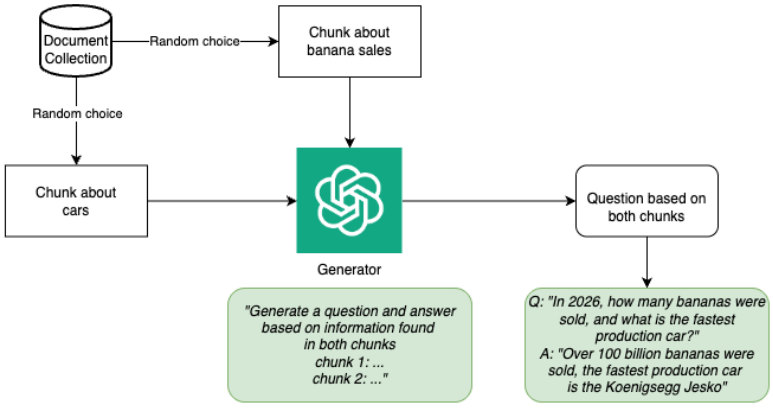


Figure 2: An example of a bad chunk combination. The chunks are completely unrelated, but the prompt forces the LLM to generate a question combining the two chunks, leading to the model asking two unrelated questions in one.

- The chunk contains too little information for the LLM to be able to generate a good question.
- The generated question does not directly relate to a subject from the text, making it too difficult to retrieve the original chunk.

A possible solution to the first and second point would be to supply the LLM with multiple chunks at the same time, and asking it to generate a question based on information from all chunks. This way, generated questions should become more complex, as the RAG will need to retrieve all used chunks to generate the correct answer. On top of this, the extra context means there is more content to base the question on, which will make it easier to generate a proper question.

However if all chunks provided to the generator are chosen randomly, there is a substantial chance that the chunks contain no overlapping information, and no logical question can be formed on the basis of those two chunks. To support this idea, we created a graph showing similarities between documents for a collection of news articles. We embedded each document using semantic embeddings. We then calculated the cosine similarity between all pairs of documents, and filter for similarity scores higher than 0.5. A sample of the resulting clusters are shown in Figure 1. Documents are often only related to a small subset of other documents. Because of this, the chances of selecting two random documents containing overlapping information is very small. This issue leads to the LLM asking two separate questions in one sentence, and provides no real information-aggregation in the question. An example of this is shown in Figure 2.

Using RAG to improve RAG To solve the problem with unrelated chunks, we propose MultiChunkQA. With our method, instead of just feeding the model with two randomly chosen chunks, we attempt to find chunks that are related to each other, by first putting the initial random chunk through the existing RAG. By doing so, we retrieve the chunks that are most closely related to this chunk. We then use the initial chunk and the chunk most closely related to this chunk as input for the LLM, and prompt it to generate a QA pair that requires information from both chunks. MultiChunkQA increases the chances of two chunks containing overlapping information, and makes the questions more difficult by requiring information from multiple chunks. This method should

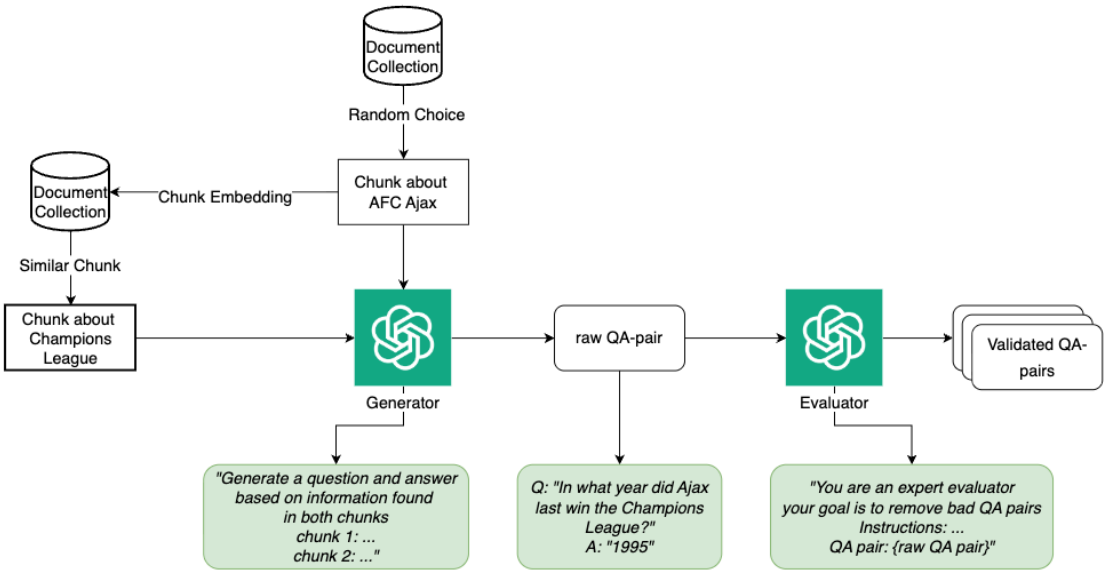


Figure 3: Flowchart showing how QA pairs are generated with Multi-ChunkQA. Chosen chunks are related to each other, which causes the generated questions to be well-formed questions based on information from both chunks. As an extra safety measure, generated QA pairs are evaluated again by an LLM to guarantee quality.

make it more likely that the resulting questions are of good quality. To make sure all resulting QA pairs are coherent and of good quality, we finally put all generated pairs together with their respective context chunks through an LLM-as-a-judge step, which will validate for all QA pairs whether the answer to the question can be found in the context, content from the chunks is directly addressed, and that the questions are answerable. In Figure 3 we show a high-level overview of our method. Both the generation prompt and the evaluation prompt used in our method can be found in the appendix in Table 17 and 18.

3.4 Synthetic QA evaluation

To evaluate the quality of the synthetic data, we use various qualitative and quantitative metrics, which we will now explain.

3.4.1 Ranking agreement

To find out whether the synthetic questions actually give us meaningful insights about the performance of the RAG setup, we have to find out whether evaluation with synthetic QA data ranks different RAG setups in the same order as evaluations done on real ground-truth reference data. This way we can guarantee that synthetic data will return the same results as real data. For this test, we assume that the real QA data always returns the correct rankings. To calculate the agreement in ranking between real and synthetic data, we make use of Kendall's Tau τ. Kendall's

Tau is calculated as follows:

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)} \tag{2}$$

Where n_c is the number of concordant (correct) pairs, n_d is the number of discordant (wrong) pairs, and n is the total number of observations. As input we separately give the faithfulness, answer relevance and context recall scores from RAGAS. The formula then calculates for all possible pairs of RAG setups whether the two RAGs are ranked correctly relative to each other. τ returns a score between 1 (absolute agreement) and -1 (absolute disagreement). If τ returns a score above 0.5, we show a strong agreement between the real and the synthetic data.

3.4.2 Question validity

To check whether the generated data is actually meaningful, we introduce a qualitative test on the validity of the generated QA pairs. For each QA pair, the questions and answers are evaluated by two human annotators to find out if generated pairs meet the following criteria:

- The generated questions is grounded by the context that was provided to the LLM.
- The questions should be answerable for the RAG.

We define a question as answerable, when it asks a clear question related to the text. It should be clear from the question what information is necessary to answer it.

Some examples of good questions: "Which iPhone did Apple release in 2012?", "Which player scored the winning goal in the Champions League final of 2025?". These questions ask direct questions that require a clear answer, and there is only a single answer to these questions.

Some examples of bad questions: "Which phone did the company release?", "Which player scored the winning goal?". These questions do not directly contain any keywords from the context, and therefore do not directly relate to the original chunk. For this reason they do not have a clear defined answer, and multiple different answers could be found for the question (not clear which company is being referred to, not clear for which game we want the winning goal).

The percentage of grounded and answerable questions will give an indication of how strictly the questions should be checked before being put to use. It should be stated that we will not judge how well-formed the questions are. The sentence structure can be very inhumane, or a question could be asked that a human would never ask, but as long as the questions remain answerable, the questions will be deemed valid in this test.

To calculate the agreement between the two annotators, Cohen's Kappa will be used. The formula for Cohen's Kappa is seen in formula 3

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \tag{3}$$

Where p_0 is the observed agreement between annotators, and p_e is the probability of chance agreement between annotators. We aim for agreement scores above 0.5, which indicates moderate to strong agreement between annotators.

3.4.3 Question difficulty

The main purpose of RAG system evaluation is to find questions that the RAG struggles to answer. For this reason it is important that the generated questions are relatively difficult to answer. To test the difficulty, we will make the assumption that all generated QA pairs are valid, and based on information that can be found in the knowledge collection of the RAG. If the model manages to generate questions that the RAG struggles to find an answer to, the synthetic questions can be considered difficult to answer. In this test, we will make a comparison between real questions, single chunk synthetic questions, multi-chunk synthetic questions, and an existing open source method which we will take as a baseline.

The used baseline model comes from Kiln-AI, a company focused on improving LLM solutions such as building agents, optimizing prompts, and building RAG models.[4] A part of their focus lies in improving RAG models. For this reason they built a method for generating synthetic QA data from the collection. In our experiment, we will make use of their synthetic data generation setup and use it as a baseline for synthetic data. The method used by Kiln-AI is very simple: a document is presented as input to the LLM, and it uses the following system prompt: "generate a high-quality QA pair based on the following document". Note that the main difference between this method and our method lies in Kiln taking full documents as input for their questions, instead of smaller chunks.

By comparing all these methods, we can determine whether using multiple chunks actually leads to more difficult questions, and whether the synthetic questions are more difficult to answer than the real questions. All models will be run using the semantic embedding from Cohere, and compared on context recall performance. Cohere is a company that specializes in AI solutions.[5]Their semantic embedding has proven to be very effective at retrieving the right documents at a relatively low latency, making it effective as a baseline setup.

3.4.4 Diversity

Another important measure is the diversity in the questions. To accurately map the performance of the RAG across the whole document collection, it is important that the questions are diverse in nature. To test how alike the generated questions are, we will calculate the semantic similarity between questions. To do this, we semantically embed the questions using cohere, and then calculate the cosine similarity between questions, as seen in equation 4. A higher similarity in this case means that the questions are more alike. We want the similarity between questions to remain small, as this means that the types and topics of questions are very diverse.

$$\text{similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4}$$

By comparing the similarity between synthetic questions and real questions, we can get a sense of the variability of the questions. We will return two similarity scores: the average maximum similarity (AMS) score as seen in equation 5, and the overall average similarity (OAS) score as seen in equation 6, where q_i and q_j are different questions from collection Q and N is the total number of questions in Q. The AMS tells us for each question which other question in the collection is most

[4] https://docs.kiln.tech/docs/evals-and-specs/evaluations

[5] https://docs.cohere.com/docs/semantic-search-with-cohere

similar to it. If the AMS score is very high, the collection most likely contains a lot of identical questions.

$$\text{AMS}(Q) = \frac{1}{N}\sum_{i=1}^{N} \max_{j \neq i} \left(\text{similarity}(q_i, q_j)\right) \tag{5}$$

The OAS tells us for each question how similar it is to all other questions. This gives a more global overview of the variance of the questions in the collection. A higher OAS score means that all questions are more focused around the same general topic.

$$\text{OAS}(Q) = \frac{2}{N(N-1)}\sum_{i=1}^{N}\sum_{j=i+1}^{N} \text{similarity}(q_i, q_j) \tag{6}$$

Lastly it is also important that the questions cover a large portion of the document collection. If all questions in the test set are related to the same few documents, we do not receive an accurate estimation about the performance of the RAG, as it might perform better or worse on other parts of the collection. To calculate the coverage of the dataset we will use the retriever for all of the questions in the dataset, and create a set containing the union of all of the retrieved chunks for this dataset. We compare the size of this set to the size of the theoretical largest set that could have been obtained (k new chunks per question), which will show whether some chunks are retrieved more often than others.

4 Experimental setup

In this section, we explain the various experiments that will be performed. First we will elaborate on the datasets that will be used for the experiments. Second we will explain the experiment for validating the accuracy of the RAGAS framework for our experiments. Lastly we will explain the experiments that will be performed to evaluate the synthetic QA pairs.

4.1 Corpora and reference data

For the evaluation, we make use of 3 datasets: MultiHop-QA, SQuAD, and DPG-QA. The starting point of our project is a real production-grade RAG which is used on enterprise scale at DPG.[6] It consists of frequently asked questions from actual users. We supplement this with well known open-source QA datasets to allow for reproducibility, and to explore similarities and differences between different document collections. SQuAD [14] is a widely used dataset for evaluation, but contains mostly simple questions from small passages, as can also be seen in Table 1 and 2. MultiHop-QA [19] on the other hand contains very elaborate questions, which are often more difficult to answer. The differences in these datasets will allow us to give a broader overview of the usefulness of synthetic data across different domains.

Both MultiHop and SQuAD consist of questions, answers, and the original context, stored in the form of (q, a, c) tuples. The contexts are important for our synthetic data purposes, as it allows us to generate synthetic questions over the same chunks as the real questions. In our experiments

[6] https://www.dpgmediagroup.com/nl-NL?referrer=https

Dataset	Collection size	average doc size (words)	no. chunks	no. questions
MultiHop-QA	609	1.746	4.914	2.556
SQuAD (dev-set)	1.204	129	1.249	5.928
DPG-QA	1.486	9.738	57.148	50

Table 1: Table showing a comparison of dataset sizes. The collection size is taken as the total number of documents in the collection. A chunk is a small passage from a document containing 512 characters.

we split all documents up into chunks with a maximum size of 512 characters, with a chunk overlap of 16 characters.

SQuAD, also known as the Stanford Question Answering Dataset, is a QA dataset based on Wikipedia articles. All articles were split up into small passages of text, and through crowd-sourcing, people were asked to write several questions that can be asked for a given passage. The splitting of the articles into small passages means that there are several related chunks of data present for each chunk, which is ideal for our purposes. The original dataset contains more than 100.000 questions, however in our work we will focus on the developer dataset, as we will only need to use a small portion of the collection in our work.

MultiHop-QA is a dataset which was created to test MultiHop-RAG capabilities. The dataset consists of news articles from various sources across various categories (business, health, sports, technology, science and entertainment). The dataset tests Multi-Hop capabilities by asking questions based on multiple different articles. This means that like in the SQuAD dataset, all articles are related to at least one other article, which is important for our strategy of creating hard questions.

DPG-QA contains questions relevant to employees at DPG Media, which is a European media firm. The collection is used for various use-cases: helping computer scientist understand existing tools, providing information about customers to marketing, helping sales with setting up new ad campaigns, and more.

The collection contains information about B2B functionalities which are used by DPG, information about campaigns and advertisers, prices for advertising, contact information for colleagues, and much more. The RAG collection used in our experiments is based on the actual collection used in production at DPG. As this dataset contains proprietary and mostly non-public information about DPG, it is not possible to make this dataset publicly available, and only the results will be shown. In Table 1 we show the size of the collections, and the amount of questions available for each dataset.

The DPG-QA collection is significantly larger than the other datasets. There are however only a very limited amount of curated QA pairs available for this collection, highlighting the need for synthetically generated questions.

Another large difference between the datasets is the types of questions that are asked, and their respective reference answers. In Table 2 we show the difference in these questions. As shown in the examples, SQuAD contains general knowledge questions about varying subjects. The questions are relatively short, and often have a direct answer. Because of this most answers are relatively short. Since the dataset was fully crowd-sourced, the quality of the questions and answers vary.

MultiHop-QA contains more elaborate questions, focusing on news articles. The questions are very long and targeted at the required documents, and contain a lot of keywords from the article. The answers are not direct answers to the question, but rather facts from the articles that together

Dataset	Question	Reference Answer
MultiHop-QA	Does the 'Sporting News' article suggest that 'Sports betting apps and live streaming' have a positive impact on the golf betting experience, while 'Essentially Sports' indicates a negative trend in 'Golf competitiveness'?	Sports betting apps and live streaming have made betting on golf easier and more exciting than ever. Moreover, there are a growing number of voices who feel golf has become way less competitive nowadays...
SQuAD (dev-set)	1. What city has the biggest port in Germany? 2. Which cidippid is used as a description on ctenophores in most textbooks?	1. Duisburg with the largest river port in Europe (Duisport)... 2. Pleurobrachia.
DPG-QA	1. What resolutions must an advertiser deliver for Carousel? 2. Can we target people who have read a certain article? 3. Where is the advanced mode in datalab?	1. For a Seamless Carousel add an advertiser must deliver between 3 and 5 slides, with a resolution of ... 2. Yes, segments can be created based on an article ID ... 3. The advanced mode can be found in the audience builder ...

Table 2: Table showing a comparison of the types of questions and answers present in the QA sets.

allow the LLM to return the right answer.

Lastly, DPG-QA contains questions mostly focused around helping the internal sales staff. It contains questions about functions relevant to advertising. The questions in the dataset mostly consist of FAQ questions, and are therefore often rather short and direct, as they are natural questions. The reference answers tend to be longer, as they give elaborate explanations to fully answer the original question, and any follow up questions. Because the dataset are mainly based on frequently asked questions, the questions in the dataset focus on a relatively small part of the document collection, and do not fully capture the collection as a whole. This dataset differs strongly from the open-source datasets, mainly in the sense that none of the questions in the dataset can be answered by the knowledge from the LLM. For this reason the performance of the RAG heavily relies on the retriever returning the necessary chunks.

4.2 Testing the evaluation framework

In this experiment, we take the RAGAS framework, and test it using a cohere-only setup. In this experiment, we change variables in a controlled manner. The variables we change are the amount of retrieved documents (also known as top-k), and the used LLM.

For top-k retrieval, we expect that if we start at k=1, the context recall will increase if we increase k, as more relevant documents will get retrieved. After a certain value of k, the context recall will stop increasing (as all relevant documents are already being retrieved). For top-k, we will experiment using top-k values of 1, 5, 10 and 20. With the knowledge we have from the DPG collection, we expect to see strong improvements in context recall up to k=10, and only minor

improvements in recall when increasing to k=20, as we already expect most of the important documents to be present in the top 10 documents.

For the LLM, we will test three models on the DPG dataset: gpt-5-chat, gpt-5.4-nano and aws-nova-lite. We only experimented with larger models, as DPG works via API calls to available models, making local compute difficult.[7] We did however choose models with large differences in total model parameter counts, with gpt-5-chat containing approximately 2-5 trillion, and gpt-5.4-nano and aws-nova-lite containing around 20 billion parameters. We expect to see strong performance with gpt-5, as this strong model should be able to provide good answers based on the provided context. With gpt-5.4-nano we expect to see a drop in faithfulness and answer relevance, as a smaller model will handle complex questions and large contexts worse. This can lead to more hallucinations in the answer, and to responses that are less directed at the question posed by the user. We expect aws-nova-lite to perform the worst, as it is the oldest model of the three. The context recall should stay the same regardless of the used generator, as the retrieved context is independent from the generator model.

In this experiment we know what the expected outcome should be, and we want to test whether RAGAS will show the expected results. If the results follow the same pattern we expect to see, we prove that RAGAS reliably determines the relative performance of a RAG setup. The test is run using the DPG dataset, as the largest dataset will likely give the most clear results in this experiment. For this part of the experiment we will only make use of the real 50 questions and answers available for DPG. The test can however also be run using the open source datasets.

4.3 Synthetic QA evaluation

To evaluate the generated synthetic QA pairs, we want to find out a couple of defining characteristics for the data: ranking agreement, variance, question validity, and question difficulty.

4.3.1 Ranking agreement

To find out whether the synthetic QA pairs are meaningful for RAG evaluation purposes, we will test multiple RAG setups on the RAGAS framework using both real and synthetic data. For the evaluation we test four different RAG setups. The setups are evaluated using both the questions and the answers of the QA pairs. In total 5 different metrics will be used. Faithfulness and answer relevance both make use of the question to help calculate scores, ROUGE and METEOR use the reference answer to calculate scores, and lastly the context recall uses both the question *and* the reference answer to calculate scores.

The resulting scores of all RAG setups can be used to create a ranking of which setup scores best.By comparing the rankings of the RAG setups on real and synthetic data, we can measure the agreement of the rankings using Kendall's Tau τ. The equation for Kendall's Tau can be found at equation 2. If the rankings have a strong agreement, we can conclude that synthetic QA pairs can accurately estimate RAG performance.

During testing, the used RAG setups will vary per dataset. This is because the RAG setup used for the DPG collection is very strong, as the whole collection contains more than 50,000 chunks. Thus a very strong RAG setup is necessary to find the right chunks. If we use the same setup with smaller datasets such as SQuAD, the differences between setups are a lot harder to distinguish.

[7]DPG is licensed to use all models used in our experiments.

DPG-QA (k=10)	Squad-QA (k=3)	Multihop-QA (k=3)
Cohere	Cohere	Cohere
Cohere → Jina Cross-encoder	Cohere → Jina Cross-encoder	Cohere → Jina Cross-encoder
(BM25+Cohere) → ColbertV2	BM25	BM25
(BM25+Cohere) → (ColbertV2+FusionRank)	BM25 → Flashrank	BM25 → Jina Cross-encoder

Table 3: Table showing for each dataset which RAG setups were tested. A "+" means two tasks are run in parallel, a "→" means the top results on the left are used as input for the right (the reranker), since we only rerank a small subset of documents.

In Table 3 we show for each dataset which RAG setups have been used. We evaluate a total of six retrieval and reranking models, beginning with BM25, a well-known TF-IDF-based ranking method that provides a fast and effective baseline. For dense retrieval we use the dense embeddings from Cohere. ColBERTv2 is a late-interaction retrieval method optimized for both computational efficiency and high retrieval performance [16]. Finally we test three different reranking methods. First, the Jina cross-encoder by Jina-AI.[8] Second we use FlashRank, a cross-encoder optimized for fast reranking [4]. Lastly we use FusionRank, a method for fusing the top results from multiple rankings [2].

4.3.2 Question validity

For this experiment we evaluate the validity of a total of 50 generated questions. All questions will be generated using MultiChunkQA, using 2 chunks as input. 25 of the questions will be based on the MultiHop collection, and the other 25 will be about the DPG collection. As described in section 3.4.2, the questions should be grounded by the context, and need to be answerable. To prevent biased judging, the evaluation will be performed by two different qualified evaluators. The annotator agreement will be reported to measure consistency. This experiment will give an insight into how often our method actually produces valid QA pairs, which tells us to what extent the generated questions still need to be checked by humans before they are put to use.

4.3.3 Question difficulty

The question difficulty will be determined based on the context recall scores of the RAG given the questions. The context recall directly measures what percentage of important information was retrieved for the questions. This gives us an indication of whether some of the generated questions are actually hard to retrieve the right information for. It should be noted that this is just one of the dimensions of question difficulty, as we will not look at whether the generated questions are actually hard to answer, merely whether the correct information is retrieved.

For this test, we make the assumption that all of the questions are fully answerable, and that a retrieval score of 100% is possible. In the test, we will compare the recall scores of the human annotated questions against synthetic methods. We will test our own method MultiChunkQA using one to four chunks, to show the impact of using more chunks on the recall score of the RAG. We

[8] https://jina.ai/models/jina-reranker-v3

Top-k	Faithfulness	Context Recall	Answer Relevance	ROUGE	METEOR
k=1	0.811	0.531	0.680	0.228	0.265
k=5	0.786	0.639	0.712	0.247	0.278
k=10	0.790	0.722	0.755	0.239	0.287
k=20	0.828	0.748	0.766	0.255	0.296

Table 4: Table showing RAGAS scores across various top-k hyperparameter settings on the DPG collection. All runs were performed using gpt-5.4-nano for both the judge and the generator model. Runs show a strong improvement in context recall up to k=10, and a slight improvement when going to k=20. Increasing k also seems to lead to improved ROUGE, METEOR, and answer relevance scores.

will also test against the existing baseline method from Kiln-AI, which uses full documents as input instead of chunks from a document.

4.3.4 Diversity

Lastly, the variability of the questions will be evaluated. LLMs tend to generate questions that are semantically similar. To make sure that the questions are diverse enough to pose questions about all parts of the dataset, we will calculate the AMS and the OAS, as seen in equations 5 and 6. These metrics will be calculated over all datasets.

We will also calculate the coverage of the questions. Since both SQuAD and MultiHop were built to have questions based on every chunk, these datasets do not apply to our experiment. The DPG collection shows a much more realistic scenario about QA availability in companies, and will therefore be used as the basis for this test. The DPG question set will first be compared against a set of 50 synthetic questions, to find out whether the synthetic questions cover more chunks than real questions. Then a synthetic dataset will be generated containing 1000 questions about the DPG collection. For each 100 questions reached, we will report the total amount of distinct retrieved chunks, and the marginal gain in newly reached chunks since the previous questions. This will give an indication of whether all chunks have an equal chance of being selected, or that some chunks are reached more often than others. Questions will be generated using MultiChunkQA with 2 chunks as input, and for each pair of 2 chunks we prompt the model to generate 2 questions. In this test, we will use the last setup in the DPG column found in Table 3, with k=10.

5 Results

In this section we will present the results of the experiment, starting with the RAGAS framework evaluation, and then explaining the results found in the synthetic QA experiments.

5.1 Evaluation framework test

In Table 4 we show the effect of various settings for top-k on the returned RAGAS scores. As shown, the context recall scores steadily increase between k=1 and k=10. As was expected, the

Used LLM	Faithfulness	Context Recall	Answer Relevance	ROUGE	METEOR
aws-nova-lite	0.851	0.739	0.684	0.304	0.327
gpt-5.4-nano	0.790	0.722	0.755	0.239	0.287
gpt-5-chat	0.876	0.712	0.781	0.256	0.322

Table 5: Table showing RAGAS scores across three different LLMs, the runs were done using k=10 on the DPG collection.

recall only slightly increases when moving up to k=20 from here, as most important documents have already been retrieved. What is interesting to see is that the answer relevance also seems to increase with larger top-k. This likely indicates that more context is required to generate an answer that fully answers the question posed by the user.

Surprisingly, increasing the size of the context to k=20 does not seem to affect the faithfulness and answer relevance scores when we use gpt-5.4-nano. Older LLM models often struggle with larger contexts, as they often forget parts of the context as the size of the context grows. This also aligns with the findings from van Elburg et al. [21], where larger context windows seemed to have no influence on the model performance.

In Table 5 we show the results for testing multiple LLMs given the same retrieval setup. The table shows that the difference in faithfulness between different LLM models is much more clear than between different runs with the same LLM, as seen in Table 4. The results show a clear gap between larger and smaller LLM models, with gpt-5-chat outperforming gpt-5.4-nano by 8.6% points.

The results also show the importance of evaluating both the faithfulness and the answer relevance. Aws-nova-lite returns higher faithfulness scores than gpt-5.4-nano, but it returns a lower answer relevance score. This indicates that the aws model hallucinates less in its responses given the information in the documents, but that the responses tend to answer the actual posed question worse.

Lastly, the context recall was expected to remain the same in each run. This is because the same retrieval setup is used in every run, and the retrieval process is deterministic. As shown in the table, the largest difference in score is 2.7% over a test set of 50 questions. This tells us that the returned RAGAS scores can fluctuate slightly per run, depending on how confident the judge model is about its answer. This also means that small improvements in RAGAS scores do not guarantee actual improvements, as a single run could give slightly inaccurate results. For more certainty, multiple runs should be performed and averaged over.

5.2 Synthetic QA evaluation

We will now explain the results of the 4 separate experiments run for evaluating the synthetic QA data.

5.2.1 Ranking agreement

In Figure 4 we show the agreement in RAG rankings between real and synthetic QA sets. The results show that they agree in most cases. Especially the context recall, the answer relevance, and METEOR manage to return a ranking similar to the real dataset, with all τ scores being higher

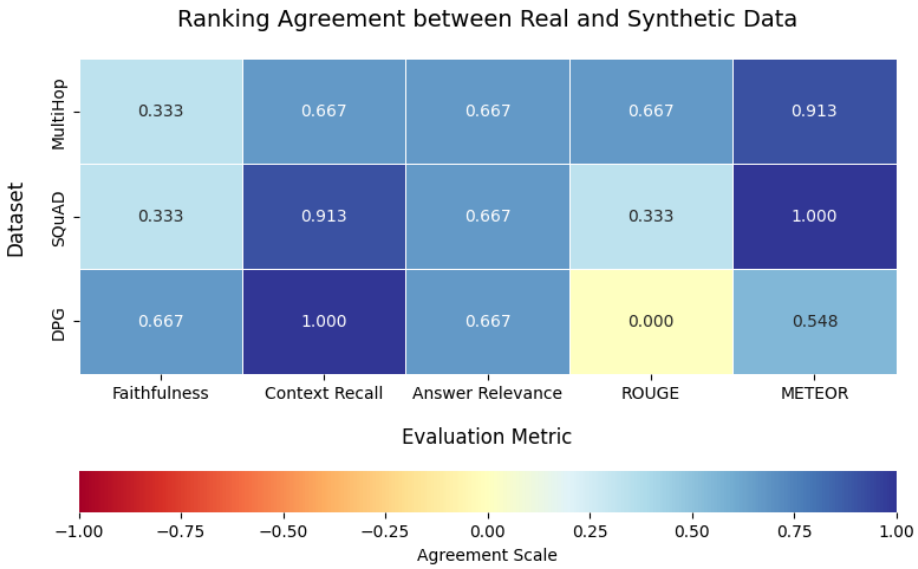


Figure 4: Table showing ranking agreement across the tested metrics for all 3 datasets. The Context Recall and the Answer Relevance are showing a constant strong agreement across all datasets. Faithfulness and ROUGE seem harder to rank, showing mostly moderate agreement.

than 0.5, which indicates that there is an overall strong agreement between real and synthetic data. Interestingly, the agreement for ROUGE is a lot lower than METEOR, although they measure roughly the same aspects of the answer. The lower agreement for ROUGE likely stems from the metric needing exact word-matches. Since an LLM produces a slightly different answer each time, the ROUGE score can vary with each run. METEOR is more robust to this issue, as it also allows matching with semantically similar words, and therefore suffers less from this problem.

These results tell us that synthetic datasets are capable of accurately determining the performance of a RAG setup, and that real questions are not a necessity. The actual RAGAS scores on which the Kendall's Tau calculations were performed can be found in the appendix in tables 11 to 16.

5.2.2 Question validity

To test whether the generated questions are answerable, a small qualitative test is done on both the MultiHop and the DPG collection. For each set, 25 generated QA pairs are generated for which we check the validity of the question and answer in their respective contexts. For each question, we check both whether a question is answerable, and whether the answer is grounded by the context, as described in section 3.4.2. The test is performed by two separate annotators, and the agreement between annotators is calculated using Cohen's Kappa. In Table 6 we show the results of this test.

In the MultiHop collection, on average we judged 80% of the QA pairs is answerable. The remainder of the questions are often overly specific, leading to confusing questions. It should be stated that this does not necessarily mean that the remaining 20% will always be answered wrongly, but rather that the chance of a mistake occurring is relatively high. On average 88% of the generated answers are factual, showing that the LLM had less issue with grounding the answer in the context. Most mistakes in grounding the answer stem from the LLM making assumptions

	DPG		MultiHop	
	Answerable	Grounded	Answerable	Grounded
Annotator 1	84%	92%	84%	92%
Annotator 2	80%	84%	76%	84%
Cohen's Kappa	0.86	0.63	0.50	0.63

Table 6: Table showing validity results per dataset for both annotators, and the amount of agreement between the two annotators.

Question Type	**SQuAD**	**MultiHop**	**DPG**
Real questions	0.936	0.371	0.777
Synthetic 1 chunk	0.950	0.972	0.946
Synthetic 2 chunk	0.985	0.872	0.832
Synthetic 3 chunk	0.862	0.632	0.740
Synthetic 4 chunk	**0.803**	**0.591**	**0.675**
Kiln-AI	0.980	0.907	0.720

Table 7: Table showing context recall scores using a cohere-only setup. For each dataset, the synthetic generation method with the lowest recall score is bolded, indicating the most difficult questions.

about the context, leading to answers that are not factually correct.

The DPG collection shows results that are very similar to those found in MultiHop, with 82% of questions being answerable, and 88% of answers being grounded. The types of mistakes that are made in this dataset are also very similar to those made in the MultiHop dataset. This shows that the quality of the generated questions is consistent across datasets, and that generating QA pairs for other collections will likely return similar results. The annotators show a moderate to strong agreement across all metrics, further reinforcing these results.

The results show that LLMs are able to generate QA pairs relatively well, but that generated questions should still be scanned by a human for validity before putting the questions into use, as a small portion of questions might still contain errors.

5.2.3 Question difficulty

In Table 7 we show the results of the context recall scores across datasets for real questions and various synthetic QA generation methods. The real questions gave relatively low recall scores across all datasets.

The MultiHop dataset has an especially low recall. This is mostly due to the given reference answers for the dataset. Rather than being answers to the posed questions, the answers consist of statements that together with reasoning lead to the answers of the question. Because of this, not all statements in the answer can be explicitly found in the retrieved text, leading to the low score.

SQuAD overall has very high recall scores. This is likely because of the small size of the collection, which makes it easier to retrieve the correct documents. On top of this the topics in SQuAD are quite diverse, which might make it easier to distinguish the correct documents when certain keywords are present in the question.

Dataset	AMS	OAS	ROUGE-L	METEOR
MultiHop-QA	0.708	0.405	0.145	0.157
MultiHop-QA-synthetic	**0.645**	**0.323**	**0.095**	**0.118**
SQuAD	**0.645**	0.431	**0.101**	**0.102**
SQuAD-synthetic	0.654	**0.425**	0.109	0.126
DPG-QA	0.711	0.500	**0.082**	**0.091**
DPG-QA-synthetic	**0.677**	**0.463**	0.118	0.120

Table 8: Table showing a comparison in the Average Max Similarity (AMS), Overall Average Similarity (OAS), ROUGE-L and METEOR scores within the dataset. Bold values show the best results.

The real questions in the DPG dataset return a retrieval score of 0.777, showing that these questions are already hard to answer for the system. The real questions in the DPG dataset likely also contain some MultiHop characteristics, combining knowledge from various sources to come to an answer, which increases the difficulty of the questions.

The results of our MultiChunkQA method show the effectiveness of using multiple chunks in a question. As we can see, using a single chunks returns simple questions, achieving nearly full context recall for all questions. When we increase the amount of chunks used in generating a question, we see a strong decrease in context recall, indicating the questions become harder when we use more chunks. The largest drops in recall happen between going from 1 to 2 chunks, and from 2 to 3 chunks. 4 chunks still returns lower recall scores, however the decrease in recall is smaller, indicating that there might be a limit to the possible difficulty of questions.

The performance of the Kiln-AI method varies strongly per document collection. The questions it creates for SQuAD are very simple, with almost full accuracy. For MultiHop it performs a little better, but still manages to score about 90%. On the DPG set however, it returns very solid results, only being outperformed by the 4 chunk version of MultiChunkQA. These results are interesting, as Kiln-AI is less restrained in what it can ask questions about, being fed full documents at a time, yet question are often still simple. The performance of the Kiln method is heavily dependent on the size of the documents it uses. As shown in Table 1, the DPG collection consists of very large documents, which are ideal for the method by Kiln, as it makes use of full documents instead of chunks to generate questions. For documents consisting of many chunks, this essentially leads to the model creating multi-hop questions. It should be stated that feeding full documents to the model like this will quickly lead to high LLM costs, as the model will consume large amounts of input tokens.

The results tell us that synthetic questions can outperform human annotated QA pairs in terms of question difficulty, which will make it easier to find the performance limits of a RAG setup during the evaluation.

5.2.4 Diversity

In Table 8 we show the results of calculating diversity within a question dataset, and in Table 9 we show the results of calculating diversity between real and synthetic question datasets.

In Table 8 the results show that synthetic questions seem to be able to generate more diverse questions than human annotated questions. The LLM likely has more diverse question asking styles

Dataset	AMS	OAS	ROUGE-L	METEOR
MultiHop	0.467	0.313	0.086	0.073
SQuAD	0.553	0.346	0.083	0.120
DPG	0.587	0.452	0.080	0.108

Table 9: Table showing a comparison in the Average Max Similarity (AMS), Overall Average Similarity (OAS), ROUGE-L and METEOR scores calculated between the real and synthetic dataset. Bold values show the best results between real and synthetic datasets.

	Retrieved Chunks	% of maximum
Real Questions	341	68.2%
Synthetic Questions	467	93.4%

Table 10: Table comparing the amount of distinct chunks retrieved over 50 questions between real and synthetic questions. At most 500 distinct chunks could have been retrieved.

than humans do, meaning it is able to ask more semantically different questions. Only in SQuAD does the generator not outperform human annotated questions. This is likely due to the small size of the chunks, limiting the amount of diverse questions that can be asked.

As shown in Table 9, the AMS scores are much lower than in Table 8. This means that in general the questions are quite diverse in nature, and most synthetic questions are not closely related to questions found in the real data. The other metrics also show slightly lower scores than those found within the datasets, indicating that both the structure and the topics of real and synthetic questions are quite different. Synthetic questions might therefore have a more unnatural structure when compared to real ones.

Finally we test how well the synthetic questions can cover the collection. To do this, we first compare the 50 questions present at DPG against 50 synthetic questions based on the amount of distinct chunks that these questions retrieve. In Table 10 we show the results for this experiment. The real questions contain more duplicate chunks than the synthetic questions at just 50 questions. The real questions are already reaching a lot of duplicate chunks, while still only having visited less than 1% of the collection. This shows that the synthetic questions are a lot more diverse in nature than the real questions, and more often require different chunks to be answered.

To further show the capabilities of the synthetic method to cover the collection, we generate a question set of 1000 questions, and test how many different chunks are being selected. For each question the top 10 chunks will be retrieved, thus leading to a maximum of 10000 chunks that can theoretically be reached after 1000 questions. In Figure 5 we show the coverage of the collection in both the amount of distinct chunks retrieved, and the gain in new retrieved chunks per 100 questions. The gain in newly reached chunks decreases the more questions have been asked, with the marginal gain halving as we reach 1000 questions. At 1000 questions, 5166 chunks have been reached. This is approximately half of the chunks that theoretically could have been reached. Since the DPG collection consists of more than 50.000 chunks, this still only covers less than 10% of the collection. This shows that creating a QA collection that covers the entire collection is difficult, and would likely require specific questions for all chunks in the collection.

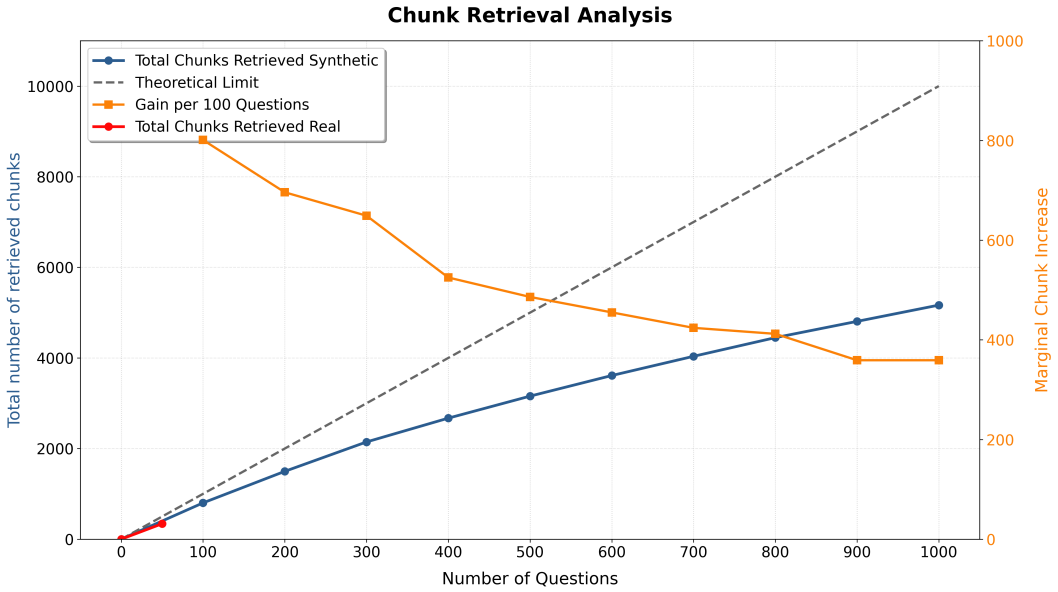


Figure 5: Figure showing the total amount of retrieved chunks, the gain in reached chunks per 100 questions, and the theoretical limit to the amount of retrieved chunks. As shown in the figure, the gain per 100 questions decreases fast over the amount of questions. The total number of chunks is 50.000+.

6 Discussion

In this study, we researched the possibility of using synthetic QA as a replacement for real data to evaluate RAG systems, and the possibility of generating questions that are difficult for the RAG to answer. Where most works for generating synthetic QA data focus on generating large quantities of training data to train retrievers, our work focuses on generating high-quality reference data to evaluate RAG systems with. To test the usefulness of our method, we first performed an experiment to assess the accuracy of the automated evaluation framework RAGAS, and then performed various quantitative and qualitative experiments on our proposed synthetic data generation method MultiChunkQA. Overall the experiments returned positive results. Synthetic QA data returned similar RAG rankings to real data, and our proposed method for generating synthetic data was able to generate more difficult questions by using multiple chunks for generating data. Our experiments did however make some assumptions about the results which we will discuss now.

6.1 Limitations

Results of our experiments show that RAGAS returns the expected relative results in the controlled experiments, therefore we assume that any results with the framework in further experiments are accurate. It should however be stated that running the same experiment multiple times can still lead to slightly varying results, as the LLM-as-a-judge models in RAGAS can output different scores for the same question across runs. In Table 5 for example the exact same documents were retrieved in all runs, meaning the context recall should remain the same for each run. Yet the context recall scores fluctuate by 2.7% between the 3 runs. Because of this, the experiments for

ranking agreement and question difficulty could return slightly different results if it were to be run again.

In the ranking agreement experiment 4 different RAG setups were tested for each dataset, using both real and synthetic data. The experiment was limited to 4 different setups due to LLM token limits and overall experiment run times. Overall results of the experiment are positive, however since only 4 setups were tested, the results are not statistically significant. As the experiment shows moderate to strong agreement for nearly all metrics, it is expected that the addition of more setups would further reinforce the current results.

Since the question validity task is very time-consuming, a relatively small amount of questions was evaluated by annotators, with 25 QA pairs being evaluated for both the DPG and MultiHop dataset by each annotator. On top of this, the questions were only validated for the MultiChunkQA method using 2 chunks. As using more chunks also means more chunks need to be considered in evaluation, the task becomes more time-consuming the more chunks we use. For this reason, the validity test is quite limited. However since annotator agreement is moderate to strong, and overall validity scores are quite high, the synthetic QA pairs should be of relatively high quality.

In the question difficulty experiment we assume that all questions can be answered, and that an accuracy of 100% is achievable. In reality this is likely not be the case, and some questions might not be fully answerable. For this reason the results might not be entirely accurate.

6.2 Implications

DPG has made use of the evaluations performed on RAGAS to test out new RAG setups, however they have (as of yet) not incorporated synthetic questions into their QA dataset. This is mostly due to the need for a final human check on the questions. As this still takes up time, and most experts are too busy with more important work to make time for this task. On top of this, the need for improving the RAG falls as the performance reaches acceptable levels.

6.3 Future work

In the first experiment we performed showed that context recall still increased going from k=10 to k=20, without any perceived reduction in faithfulness or answer relevance. This is interesting, as current beliefs would suggest that larger contexts will lead to the LLM forgetting parts of the context, and not returning the correct answer. It could be tested whether this phenomenon is unique to the DPG dataset, or that newer LLM models are able to hold larger contexts without forgetting any earlier context. To do so, models should be tested to find their "limits" by increasing the context windows even further, and testing if the answers from the LLM will start deteriorating eventually.

The cross-dataset diversity experiment showed that the similarity between real and synthetic questions was smaller than the similarity within the real questions dataset. This could indicate that synthetic questions do not form questions in the same way humans do. In future work it could be tested whether it is possible to mimic the structure of human formulated questions by providing examples to the LLM in its prompt, or by fine-tuning a model with real questions before prompting it to generate questions.

In our experiment for testing the ranking agreement between real and synthetic data, the amount of RAG setups tested was limited to 4 due to large runtimes and the LLM costs. For more

conclusive results with significant results, the test should be performed using more setups. For absolute certainty the tests should also be run for multiple iterations, as the RAGAS scores can fluctuate slightly over different runs.

As our experiment for question difficulty showed the strong performance of Kiln-AI on the DPG dataset, it could also be attempted to generate multi-hop questions using multiple documents instead of multiple chunks, in an attempt to generate more difficult questions.

In our work we mainly focused on creating difficult questions for RAG evaluation purposes, however difficult questions might also improve training performance of retrieval models such as cross-encoders. In future work it could be attempted to incorporate these difficult questions in the training data to find out if harder questions can also increase retrieval performance of cross-encoders.

Lastly, MultiChunkQA was created to generate synthetic QA data for RAG evaluation, but the method could also be applied to various other use-cases. For example, creating document summaries by combining related chunks in the document instead of one summary for each chunk. This could lead to more comprehensive and accurate document summaries for very large documents. Another idea would be to use the multi-hop questions to create test sets that evaluate the reasoning abilities of a model, by creating questions that require information from multiple chunks where the connection is not explicit. This forces the model to find the answer itself through reading comprehension and reasoning skills.

7 Conclusion

In this work, we proposed and evaluated a new method for generating synthetic QA data for RAG evaluation. To find out how useful the synthetic data is at evaluating RAG systems, we posed two research questions:

- How similar are synthetic QA evaluation results compared to real QA results?
- How effective are synthetic methods at generating "edge-case" questions which are hard to answer for the RAG?

Our findings suggest that synthetic QA data returns evaluation results similar to evaluation results from real data when used on the automated evaluation framework RAGAS, indicating that synthetic QA data returns accurate results for RAG performance. We also show that our method is able to generate questions which are difficult for the model to answer, with results showing that using 2 or more related chunks for generating questions leads to lower context recall scores, meaning the questions are more difficult. Context recall scores continued to decrease the more chunks were used to generate questions. With this in mind, using synthetic data generated with our method can find the limitations of the RAG performance, while also being able to compare the performance of various retrieval models accurately.

From a small experiment in question validity we find that roughly 80% of questions generated by our method are high-quality questions. This indicates that our synthetic QA generation method can be used for evaluation, but that generated questions should still be checked by human experts before being put into practice, as the generated QA pairs might contain faulty data.

For future work with our method, multiple directions can be followed. First, a test can be performed to test the context limits of newer LLM models. The usefulness of our synthetic data as

training data could also be tested. Our method could also be improved by generating more natural human-like questions, through either fine-tuning or improved prompting. Last, MultiChunkQA could also be applied to various other use-cases, such as improved document summarization, and testing the reasoning skills of LLMs.

References

[1] Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. Synthetic QA Corpora Generation with Roundtrip Consistency. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy, July 2019. Association for Computational Linguistics.

[2] Maya ALGhafri, Imran Khan, and Abdelhamid Abdessalem. Fusionrank: An ensemble approach for unsupervised extractive summarization using probabilistic and sentence embedding-based features. *IEEE Access*, 2026.

[3] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[4] Prithiviraj Damodaran. Flashrank, lightest and fastest 2nd stage reranker for search pipelines. *Zenodo*, 2023.

[5] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAS: Automated Evaluation of Retrieval Augmented Generation. In Nikolaos Aletras and Orphee De Clercq, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta, March 2024. Association for Computational Linguistics.

[6] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[7] Jiarui Li, Ye Yuan, and Zehua Zhang. Enhancing LLM Factual Accuracy with RAG to Counter Hallucinations: A Case Study on Domain-Specific Queries in Private Knowledge-Bases, 2024.

[8] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 110–119, 2016.

[9] Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10443–10461, Singapore, December 2023. Association for Computational Linguistics.

[10] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[11] Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. Best Practices and Lessons Learned on Synthetic Data, 2024.

[12] Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11065–11082, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[13] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document Expansion by Query Prediction, 2019.

[14] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 2383–2392, 2016.

[15] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[16] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, 2022.

[17] Siamak Shakeri, Cicero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. End-to-End Synthetic Data Generation for Domain Adaptation of Question Answering Systems. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5445–5460, Online, November 2020. Association for Computational Linguistics.

[18] Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D. Cox, and Akash Srivastava. LAB: Large-Scale Alignment for ChatBots, 2024.

[19] Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*, 2024.

[20] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[21] Jonas van Elburg, Peter van der Putten, and Maarten Marx. Can We Evaluate RAGs with Synthetic Data? In Irena Koprinska, João Mendes-Moreira, and Paula Branco, editors,

Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pages 177–192, Cham, 2026. Springer Nature Switzerland.

[22] Ruida Wang, Wangchunshu Zhou, and Mrinmaya Sachan. Let's Synthesize Step by Step: Iterative Dataset Synthesis with Large Language Models by Extrapolating Errors from Small Models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11817–11831, Singapore, December 2023. Association for Computational Linguistics.

[23] Ran Xu, Hejie Cui, Yue Yu, Xuan Kan, Wenqi Shi, Yuchen Zhuang, May Dongmei Wang, Wei Jin, Joyce Ho, and Carl Yang. Knowledge-Infused Prompting: Assessing and Advancing Clinical Text Data Generation with Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15496–15523, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[24] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[25] Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander J Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. Large language model as attributed training data generator: A tale of diversity and bias. *Advances in neural information processing systems*, 36:55734–55784, 2023.

[26] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[27] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *Data Science and Engineering*, pages 1–29, 2026.

[28] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[29] Yuchang Zhu, Huizhe Zhang, Bingzhe Wu, Jintang Li, Zibin Zheng, Peilin Zhao, Liang Chen, and Yatao Bian. Measuring diversity in synthetic datasets. *arXiv preprint arXiv:2502.08512*, 2025.

8 Appendix

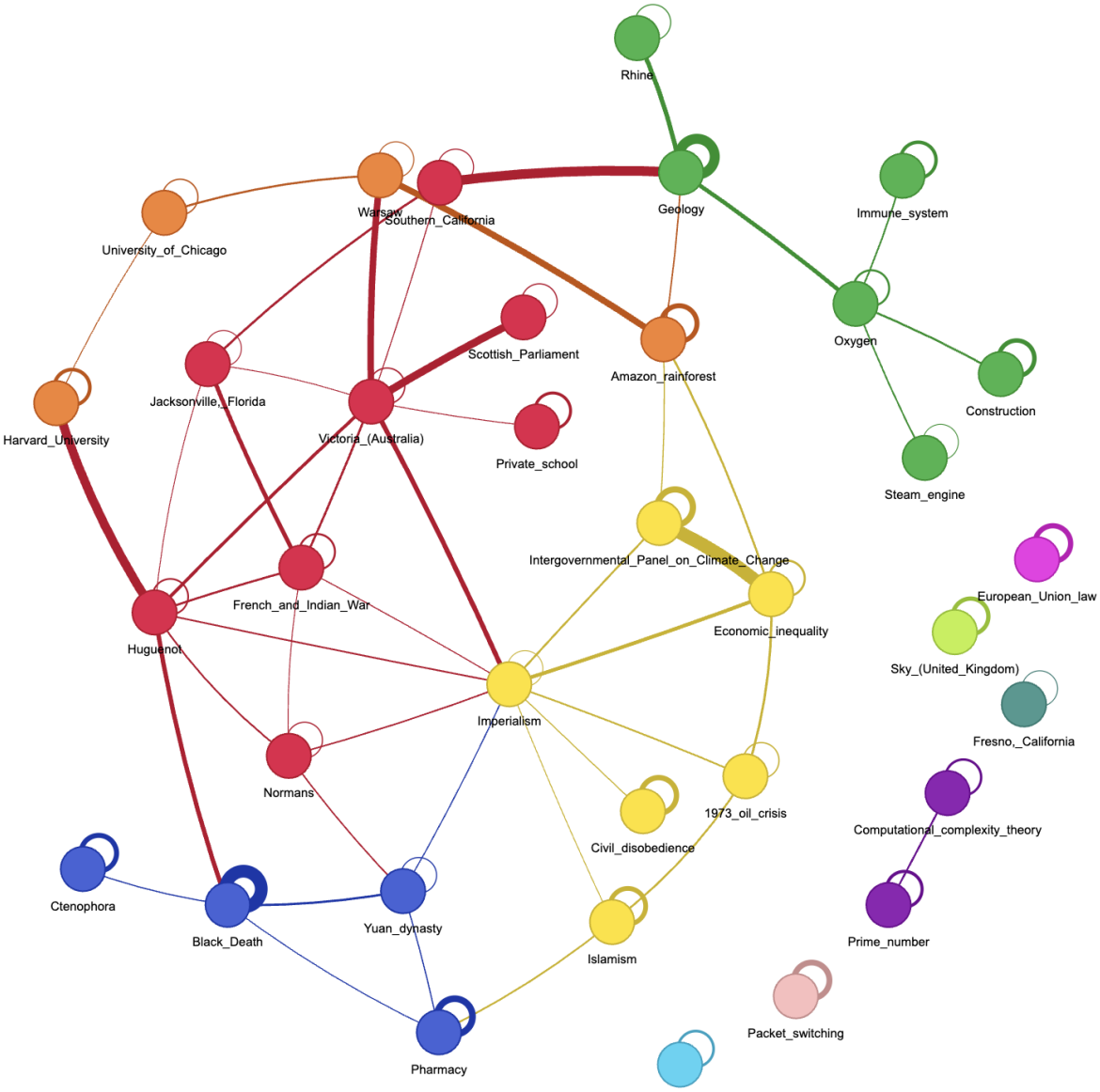


Figure 6: Page clusters found in a small subset of the SQuAD dataset. The graph shows less distinct clusters than the MultiHop dataset, as the Wikipedia articles have a wide range of topics. Self-links indicate that different documents with the same title are also related to one another.

DPG-real	Faithfulness	Context Recall	Answer Relevance
Cohere	0.776	0.777	0.738
Cohere → Jina Crossencoder	0.832	0.801	0.765
(BM25+Cohere) → ColbertV2	0.860	0.880	0.760
(BM25+Cohere) → ColbertV2+FusionRank	0.871	0.871	0.781

Table 11: Table showing the returned scores for all RAG setups tried on the real DPG dataset

DPG-synthetic	Faithfulness	Context Recall	Answer Relevance
Cohere	0.724	0.803	0.760
Cohere → Jina Crossencoder	0.780	0.822	0.822
(BM25+Cohere) → ColbertV2	0.816	0.864	0.832
(BM25+Cohere) → ColbertV2+FusionRank	0.808	0.853	0.851

Table 12: Table showing the returned scores for all RAG setups tried on the synthetic DPG dataset

SQuAD-real	Faithfulness	Context Recall	Answer Relevance
Cohere	0.975	0.936	0.855
Cohere → Jina Crossencoder	0.980	0.980	0.862
BM25	0.948	0.945	0.843
BM25 → Flashrank	0.825	0.739	0.761

Table 13: Table showing the returned scores for all RAG setups tried on the real SQuAD dataset

SQuAD synthetic	Faithfulness	Context Recall	Answer Relevance
Cohere	0.933	0.950	0.853
Cohere → Jina Crossencoder	0.964	1	0.891
BM25	0.969	1	0.882
BM25 → Flashrank	0.636	0.645	0.715

Table 14: Table showing the returned scores for all RAG setups tried on the synthetic SQuAD dataset

MultiHop-real	Faithfulness	Context Recall	Answer Relevance
Cohere	0.633	0.371	0.624
Cohere → Jina Cross-encoder	0.674	0.437	0.662
BM25	0.702	0.455	0.655
BM25 → Jina Cross-encoder	0.664	0.489	0.636

Table 15: Table showing the returned scores for all RAG setups tried on the real MultiHop dataset

MultiHop-synthetic	Faithfulness	Context Recall	Answer Relevance
Cohere	0.856	0.872	0.722
Cohere → Jina Cross-encoder	0.917	0.927	0.766
BM25	0.864	0.880	0.748
BM25 → Jina Cross-encoder	0.900	0.938	0.762

Table 16: Table showing the returned scores for all RAG setups tried on the synthetic MultiHop dataset

Validation System Prompt

```
You are an expert Question Quality Auditor. Your goal is to identify
high-quality, natural-sounding questions for a search engine.

Evaluate the provided question based on these STRICT criteria:

1. NO ANSWER LEAKAGE: The question must not contain the answer within itself.
(Fail if: "Who is the CEO of Apple, Tim Cook?")
2. NATURAL PHRASING: The question must sound like something a human
would actually type or say.
3. LANGUAGE: The question must be in Dutch or English.

REJECTION CRITERIA (Mark as '0'):
- The question contains the specific answer it is looking for.
- The question is gibberish or uses excessive technical metadata
(like "Chunk 1 says...").
- The question is too ambiguous to answer (like "which group is
the company trying to target?" It is unclear which company is being referred to.)
- Only mark a question as '0' if there is a clear violation of
one of these rules.

Otherwise, if the question is a clean, stand-alone inquiry
that can be answered using the given context, it is valid.

Output '1' for PASS or '0' for FAIL. Do not provide any explanation.
```

Table 17: The system prompt used for validating synthetic multihop QA pairs.

Generation System Prompt

```
You are creating a 'Multihop' QA dataset. You are given multiple distinct
text chunks (Chunk 1, Chunk 2, etc.).

TASK:
Generate {num_q} high-quality, challenging questions that require information
from ALL chunks to answer.

RULES:
1. THE CONNECTION: The question should use a detail from one chunk to ask about
something that can be found in the other chunks.
2. NO SPOILERS: Do not include the answer in the question.
3. BE NATURAL: Ask like a human who is curious, not a computer performing a
text-extraction task. Avoid phrases like "In Chunk 1...".
4. LANGUAGE: Match the language of the chunks (Dutch or English).
5. CLEAR QUESTIONS: Generated questions should not be ambiguous, such that the
context can still be retrieved given the question.
6. RESULTING QUESTION: The resulting QA pair should consist of a single question
and answer. Try to keep questions and answers short and concise.

GOOD EXAMPLE:
Chunk 1: "John Smith founded a tech company in 1992."
Chunk 2: "The company founded in 1992 eventually released the X-Phone."
Chunk 3: "Their most recent release, the Y-Phone, is a big hit for consumers."
Question: "Which phones have been released by the company
established by John Smith?"
Reference: "John Smiths company released the X-Phone and the Y-Phone."

BAD EXAMPLE (Leaky):
Question: "What products did John Smith's 1992 company release, which were called
the X-Phone and the Y-Phone?"
Reference: "John Smiths company released the X-Phone and the Y-Phone."
(This is bad because the answer to the question is present
in the question itself).

Structure output as a JSON list:
[{"question": "...", "reference": "..."}]
Do not include markdown tags like ```json.
```

Table 18: The system prompt used for generating synthetic multihop QA pairs.