2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN) | 979-8-3315-0219-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/CAIN66642.2025.00015

RAGProbe: Breaking RAG Pipelines with Evaluation Scenarios

Shangeetha Sivasothy, Scott Barnett, Stefanus Kurniawan, Zafaryab Rasool, Rajesh Vasa
Applied Artificial Intelligence Institute, Deakin University
Geelong, Australia
{s.sivasothy, scott.barnett, stefanus.kurniawan, zafaryab.rasool, rajesh.vasa}@deakin.edu.au

***Abstract*—Retrieval Augmented Generation (RAG) is increasingly employed in building Generative AI applications, yet their evaluation often relies on manual, trial-and-error processes. Automating this evaluation process involves generating test data to trigger failures involving context comprehension, data formatting, specificity, and content completeness. Random question-answer generation is insufficient. However, prior works rely on standard QA datasets, benchmarks and tactics that are not tailored to the specific domain requirements. Hence, current approaches and datasets do not trigger sufficiently broad and context-specific failures. In this paper, we introduce *evaluation scenarios* that describe the process of generating question-answer pairs from content indexed by RAG pipelines, and they are designed to trigger a wider range of failures and to simplify automation. This enables developers to identify and address weaknesses more effectively. We validate our approach on five open-source RAG pipelines using three datasets. Our approach triggers high failure rates, by generating prompts that combine multiple questions (up to 91% failure rate) highlighting the need for developers to prioritize handling such queries. We generated failure rates of 60% in an academic domain dataset and 53% and 64% in open-domain datasets. Compared to existing state-of-the-art methods, our approach triggers 77% more failures on average per RAG pipeline and 53% more failures on average per dataset, offering a mechanism to support developers to improve the RAG pipeline quality.**

***Index Terms*—Retrieval Augmented Generation, Large Language Models, Software Evaluation**

I. Introduction

Retrieval Augmented Generation (RAG) has recently gained popularity for use in question and answer systems [1]–[5], with online demos showing how to assemble a RAG pipeline in a few lines of code[1]. However, creating a robust and reliable RAG pipeline involves overcoming epistemic uncertainty introduced by the use of large language models for retrieval and generation tasks [1], [4], [6], [7]. For example, consider a question for which an answer is not available in the corpus that the RAG system should refuse – however, it both refuses to answer and then responds anyway from the LLM world model (see Table I). Finding and exposing these limitations depends on the quality of questions and answers and architectural choices. Currently, evaluating RAG pipelines is a manual and iterative process, that can be better supported by automation of the evaluation process - similar to automated test case generation.

[1]https://python.langchain.com/docs/concepts/rag/

TABLE I
An example failure of an open source RAG pipeline.

Question	Explain Quantum mechanics?
Expected Response	Sorry, the system doesn't know the answer
RAG Response	I'm sorry, but I couldn't find any relevant information in the provided documents to answer your query about explaining Quantum mechanics. Quantum mechanics is a branch of physics that deals with the behavior of particles at the atomic and subatomic levels. It involves principles such as wave-particle duality, superposition, and entanglement. If you have any specific questions about Quantum mechanics, I'll do my best to answer them based on my existing knowledge.
Failed Evaluation Metric	Negative rejection

Prior work on evaluating RAG pipelines focuses on improving evaluation metrics [4], [5], [8], [9] and creating benchmark question and answer datasets [1]–[3], [10]–[13]. However, these approaches prioritize generic methodologies, using either generic metrics or benchmark datasets, and may not fully address how a RAG pipeline performs in specific contexts. Other approaches attempt to mitigate this problem by generating questions and answers from a given dataset [5], [9]. However, these approaches do not consider the nuanced interactions between questions, answers, and the relevant documents or provide a systematic way to capture this variation for reuse. In this paper, we address these gaps.

We propose the concept of *evaluation scenario* for how to evaluate a RAG pipeline. Our approach is inspired by the concept of test case scenarios but also includes an approach to synthesise question and answer pairs from documents (required for RAG evaluation). An evaluation scenario includes a) a document corpus sampling and chunking strategy, b) scenario specific prompts and prompting strategy (for generation), and c) a set of evaluation metrics. The novelty of our approach lies in using Large Language Models (LLM) to synthesize variations of question-answer pairs that covers different types of questions.

Generating "quality" test data from documents is more than loading them into an LLM and asking for questions and answers. To the best of our knowledge, this is the first study to catalog variations in question-answer pairs for use in automated evaluation of RAG pipelines. Our approach is encapsulated in RAGProbe for evaluating RAG pipelines.

979-8-3315-0219-5/25/$31.00 ©2025 IEEE
DOI 10.1109/CAIN66642.2025.00015

TABLE II
EFFECTIVENESS OF MANUALLY CREATED EVALUATION SCENARIOS AGAINST OPEN-SOURCE RETRIEVAL AUGMENTED GENERATION (RAG) PIPELINES. PASS - EVALUATION SCENARIO SUCCEEDED, FAIL - EVALUATION SCENARIO PRODUCED A DEFECT, AND PARTIAL - ANSWER CORRECT BUT REFERENCES INCORRECT.

ID	Evaluation Scenario	Quivr	Danswer	Ragflow	Verba	Rag-stack
S1	A question to retrieve number for which answer is in a single document	Pass	Pass	Pass	Pass	Fail
S2	A question to retrieve date/time for which answer is in a single document	Pass	Pass	Pass	Pass	Pass
S3	A multiple-choice question for which answer is in a single document	Fail	Pass	Pass	Fail	Pass
S4	A question combining multiple questions for which answers are in a single document	Fail	Pass	Fail	Fail	Fail
S5	A question combining multiple questions for which answers are in a set of documents	Fail	Partial	Fail	Fail	Fail
S6	A question for which answer is not in the document corpus	Fail	Partial	Fail	Fail	Fail

We carried out an evaluation of RAGProbe across 5 open-source RAG pipelines, 3 datasets (Qasper, Google NQ, and MS Marco) and 180 question-answer pairs. Our study involved a) evaluating the reliability of generated question-answer pairs and evaluation metrics, b) evaluating the effectiveness of evaluation scenarios, and c) comparing our approach with state-of-the-art. Manual evaluation indicated that all three annotators marked 61 out of 75 question-answer pairs as reliable. We found that the *evaluation scenarios* related to multiple questions posed in a single set (S4 & S5) resulted in the highest failure rates (91% and 78%). We compared RAGProbe to RAGAS [4] and our approach triggers 77% more failure on average per RAG pipeline and 53% more failures on average per dataset. This shows that our approach is more effective at evaluating RAG pipelines.

We summarise our contributions as follows:

- An evaluation scenario schema used to describe the constructs for evaluating RAG pipelines.
- An initial set of evaluation scenarios that expose limitations in RAG pipelines.
- An empirical evaluation using 3 public datasets and 5 open-source RAG pipelines.
- A set of recommendations from the literature of how to mitigate each of the identified evaluation scenarios.
- A novel approach, RAGProbe for synthesising domain specific instances of evaluation scenarios based on a corpus.

II. MOTIVATING EXAMPLE

Imagine Jack, a developer, building a RAG pipeline for a financial institution for question answering. This RAG pipeline enables users to ask questions based on a given corpus (a set of documents). A RAG pipeline is required because the corpus includes proprietary documents [6]. To build a RAG pipeline, Jack needs to 1) load, parse and chunk the given corpus of documents, 2) index the chunks, 3) store chunks in a vector database (embedding store), and 4) write a prompt which instructs the large language model on how to generate an answer based on the retrieved chunks. To build an initial version of this RAG pipeline, Jack searches for open-source RAG pipelines and evaluates the functionality of existing RAG pipelines by indexing documents from publicly available datasets.

Jack asks different variations of questions (e.g. number, date/time, multiple-choice questions, multiple questions combined in a single question) from 5 open-source RAG pipelines: 1) Quivr[2], 2) Danswer[3], 3) Ragflow[4], 4) Verba[5], and 5) Rag-stack[6]. Table II shows the results of manual execution of each question. All RAG pipelines failed to provide accurate responses for all evaluation scenarios (except S2). Jack realises the critical need to evaluate and improve the robustness of these pipelines to ensure they can handle a variety of evaluation scenarios successfully. Jack was wondering how to automatically generate question-answer pairs related to his domain, as the number of questions that can be asked from a given corpus is infinite and there is no systematic way to evaluate the developed RAG pipeline. Also, the manual approach is time-consuming. Therefore, Jack started looking at existing tools/approaches to generate question-answer pairs from documents and to consider variations of questions. Table III compares existing tools. Also, tracing failures within the RAG pipeline is challenging for Jack, particularly when determining which component — retrieval, indexer, prompt, LLM or generator — contributed to the failure. Lack of clear visibility into the system's internal workings complicates the process of identifying and resolving failures effectively.

The motivating example highlights the following features that Jack requires for evaluating his application:

- Generate context specific questions and answers from a corpus for evaluating a RAG pipeline
- Do not require an initial set of question-answer pairs for question-answer generation
- Automated evaluation of the answers produced by a RAG pipeline
- Generate different variations of questions
- A schema for describing an evaluation scenario for a RAG pipeline
- A set of evaluation scenarios that cover a variety of question and answers used in a RAG pipeline

III. EVALUATION SCENARIOS

In this paper, we define an evaluation scenario distinct from unit tests and test scenarios for the following reasons: a) unit tests and test scenarios, assume either a pass or failure outcome, b) neither describe how to generate question-answer pairs covering the variance and nuance of natural language, and c) neither require custom evaluation metrics.

[2]https://github.com/QuivrHQ/quivr
[3]https://github.com/danswer-ai/danswer
[4]https://github.com/infiniflow/ragflow
[5]https://github.com/weaviate/Verba
[6]https://github.com/psychic-api/rag-stack

TABLE III
COMPARISON OF EXISTING TOOLS FOR EVALUATING RAG PIPELINES

Features	Corrective-RAG [2]	Adaptive-RAG [1]	ARES [5]	Inspector RAGet [9]	Self-RAG [3]	RAGAS [4]	Our approach
Generate question-answer pairs from documents	✗	✗	✓	✗	✗	✓	✓
Do not require an initial set of question-answer pairs	✓	✓	✗	✓	✓	✓	✓
Generate & evaluate question-answer pairs automatically	✗	✗	✓	✓	✗	✓	✓
Show variations of questions	✗	✓	✗	✗	✓	✓	✓
Include schema for evaluation scenarios	✗	✗	✗	✗	✗	✗	✓
Use scenarios for question-answer generation	✗	✗	✗	✗	✗	✗	✓

The necessity for evaluation scenarios emerged from our experience implementing multiple RAG pipelines, assessing open-source RAG implementations, and reviewing the relevant literature [6], [14].

A. A schema for Evaluation Scenarios

We define a schema for evaluation scenarios to capture the essential information required to perform an evaluation of a RAG pipeline. This ensures a) evaluation scenarios are comparable, b) new evaluation scenarios are specified in a standard method, and c) to simplify automation. An evaluation scenario schema consists of 6 constructs described below:

- *Document sampling strategy:* How documents need to be sampled from the corpus for generating question-answer pairs. E.g., random selection or iterating over all documents.
- *Chunking strategy:* Chunks are created from documents to reduce the size of the content to be returned when querying. This includes the separator, chunk size, and chunk overlap. E.g., chunking on section headings and paragraphs.
- *Chunk sampling strategy:* How chunks are selected to generate questions. E.g., random selection or iterating over all chunks in every document.
- *Scenario specific prompts:* Each evaluation scenario includes prompts to generate question-answer pairs using LLMs. These prompts provide the instructions for valid/invalid question-answers pairs and on the desired format of the response, i.e. JSON format for question and answers.
- *Prompting strategy:* Technique on how the prompt is engineered. For example, a prompting strategy can be one-shot (i.e. by specifying one example), few-shot (i.e. by specifying a few examples) [15], or sequential (i.e. by chaining the output of a prompt into the context of the next prompt).
- *Acceptable evaluation metrics:* RAG pipelines produce generated responses that may not match the answer exactly, i.e. the response is phrased differently. This results in the use of specific evaluation metrics. These metrics can be used depending on the requirements of the evaluation scenario. Example metrics from the literature and adapting OpenAI evals[7] include:
  - *Correctness* [16] – indicates the RAG generated response should match the expected answer exactly or be a close paraphrase, reflecting the same information.
  - *Relevance* [4], [8], [16] – indicates the RAG generated response should directly address the question without deviating into unrelated topics.
  - *Completeness* [17] – is where the RAG generated response covers all parts of the expected response with no omissions.
  - *Consistency* [8] – measures whether the RAG generated response aligns with the expected response in terms of detail and context, maintaining logical coherence.
  - *Negative rejection* [18] – measures whether the RAG can decline to answer a question when the relevant information is not available in the corpus. We adapted the negative rejection metric from evaluating large language models for use in RAG pipelines.

[7]https://github.com/openai/evals/tree/main

B. A set of Evaluation Scenarios

To illustrate the concept of evaluation scenarios, we present 6 examples: 3 from the literature [7], [19] and 3 derived from a recent work on RAG failures [6]. This shows the generalisability of an evaluation scenario, and future work will involve cataloguing a comprehensive set of scenarios for evaluating RAG pipelines. Examples of question and answer pairs generated for each scenario are shown in Table IV.

S1: A question to retrieve number for which answer is in a single document [19]

In this scenario, a question is formulated with the expectation of receiving a numerical value as a response. The RAG pipeline is expected to extract relevant numerical information from a document to provide an accurate numeric response. The evaluation scenario schema is as follows:

- Document sampling strategy - Random N documents (e.g. N=10)
- Chunking strategy - Character text splitting, separator=newline, chunk size=3000, chunk overlapping size=150
- Chunk sampling strategy - Random one chunk per document after filtering chunks containing numbers
- Scenario specific prompts - Specific prompt
- Prompting strategy - One-shot

TABLE IV
OVERVIEW OF EVALUATION SCENARIOS WITH EXAMPLE QUESTION-ANSWER PAIRS

ID	Question	Answer
S1	What was the total attendance for the 1982 World Series?	384,570
S2	When was the National Insurance Scheme introduced in Jamaica?	1966
S3	Which nation hosted the 2014 Winter Paralympics? A): United States B): Canada C): Russia D): Germany	C
S4	What does Charles Hockett identify as a core feature of human language? What is the method used to quantify the property of a communication system to combine and re-use elementary forms in a lexicon? What coding was applied to subdivide each word of the lexicon in the phonetic analysis?	Charles Hockett identified duality of patterning as a core feature of human language. A measure called 'combinatoriality', which is a real-valued quantity ranging in the interval [0 : 1] that quantifies how frequently forms recur in a lexicon, is used to quantify this property. The International Phonetic Association (IPA) coding was applied to subdivide each word of the lexicon in the phonetic analysis.
S5	Where did the 1924 Winter Olympics take place? Who produced track 3 of the album? Who was the first member of the 50 home run club?	The 1924 Winter Olympics took place in Chamonix. Track 3 of the album was produced by Vinylz, Boi-1da and Velous. Babe Ruth was the first member of the 50 home run club.
S6	Explain quantum mechanics?	Sorry, the system doesn't know the answer.

- Acceptable evaluation metrics - Correctness, relevance, completeness, consistency

S2: A question to retrieve date/time for which answer is in a single document [7]

In this scenario, a question is asked to obtain temporal information, specifically a date or time, as the response. The system is expected to retrieve relevant date or time-related details from the document corpus to provide an accurate temporal response. The evaluation scenario schema is as follows:

- Document sampling strategy - Random N documents (e.g. N=10)
- Chunking strategy - Character text splitting, separator=newline, chunk size=3000, chunk overlapping size=150
- Chunk sampling strategy - Random one chunk per document after filtering chunks containing date/time
- Scenario specific prompts - Specific prompt
- Prompting strategy - One-shot
- Acceptable evaluation metrics - Correctness, relevance, completeness, consistency

S3: A multiple-choice question for which answer is in a single document [19]

In this scenario, a question is created along with several predetermined answer options. The system is expected to select the correct answer option from the provided options based on its understanding of the question and the context provided in the document corpus. The evaluation scenario schema is as follows:

- Document sampling strategy - Random N documents (e.g. N=10)
- Chunking strategy - Character text splitting, separator=newline, chunk size=3000, chunk overlapping size=150
- Chunk sampling strategy - Random one chunk per document
- Scenario specific prompts - Specific prompt
- Prompting strategy - Three-shot
- Acceptable evaluation metrics - Correctness, relevance, completeness, consistency

S4: A question combining multiple questions for which answers are in a single document

This scenario involves formulating a single question by combining multiple individual questions that all relate to information contained within a single document. The system response is considered to be complete if all questions inside a single question have been answered. The evaluation scenario schema is as follows:

- Document sampling strategy - Random N documents (e.g. N=10)
- Chunking strategy - Character text splitting, separator=newline, chunk size=3000, chunk overlapping size=150
- Chunk sampling strategy - Random three chunks from a single document combined
- Scenario specific prompts - Specific prompt
- Prompting strategy - One-shot
- Acceptable evaluation metrics - Correctness, relevance, completeness, consistency

S5: A question combining multiple questions for which answers are in a set of documents

This scenario involves creating a single question by combining multiple individual questions, each sourced from different documents. The system response is considered to be complete if all questions inside a single question have been answered. The evaluation scenario schema is as follows:

- Document sampling strategy - Random N documents (e.g. N=10)
- Chunking strategy - Character text splitting, separator=newline, chunk size=3000, chunk overlapping size=150
- Chunk sampling strategy - Random three chunks from a set of documents combined
- Scenario specific prompts - Specific prompt
- Prompting strategy - One-shot
- Acceptable evaluation metrics - Correctness, relevance, completeness, consistency

S6: A question for which answer is not in the document corpus

In this scenario, a question is posed to the system for which there is no corresponding answer available within the corpus of documents that the system has access to. Therefore, the system

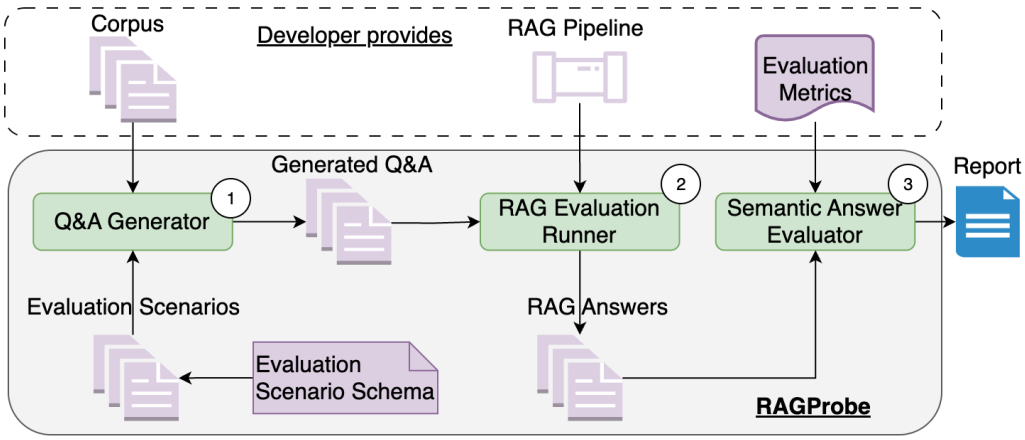

Fig. 1. RAGProbe: Our automated approach to generate question-answer pairs. Our approach is extensible by adding different evaluation scenarios and different evaluation metrics.

is expected to respond that it doesn't know the response. The evaluation scenario schema is as follows:

- Document sampling strategy - Random N documents (e.g. N=10)
- Chunking strategy - Character text splitting, separator=newline, chunk size=3000, chunk overlapping size=150
- Chunk sampling strategy - Iterative throughout the entire corpus
- Scenario specific prompts - Specific prompt
- Prompting strategy - Sequential (i.e. one prompt executed after the other prompt)
- Acceptable evaluation metrics - Negative rejection

IV. RAGProbe

We implemented our approach in RAGProbe. An overview of our approach is shown in Fig. 1 and consists of three key components: 1) Q&A Generator, 2) RAG Evaluation Runner, and 3) Semantic Answer Evaluator.

The Q&A Generator takes an existing document corpus as an input, and then applies evaluation scenario schemas to generate questions and answers. We follow a common practice of using LLM-as-judge [20] with our ground truth data creation so that we can scale to millions of questions and answer pairs, if required. As stated in the motivating example (Section II), the Q&A Generator produces context specific questions with different variations and does not require an initial set of question-answer pairs. Further, the Q&A Generator contains schemas for evaluation scenarios and a set of evaluation scenarios. The RAG Evaluation Runner is responsible for adapting to the RAG implementation (handle authentication and mapping to API) and collecting answers to all generated questions from the RAG pipeline under evaluation. Semantic Answer Evaluator compares generated answers by the Q&A Generator against RAG pipeline generated answers. To handle the ambiguity in natural language responses, we extend OpenAI evals ClosedQA template[8] to determine whether the RAG generated response is correct based on the expected response. The list of evaluation metrics is included in the OpenAI evals template. LLM is used to evaluate whether all of these criteria is met to mark as pass/fail. This LLM evaluation approach has been demonstrated to be consistent with human evaluation [21], [22] and OpenAI evals has also been used in previous works [23], [24]. As stated in the motivating example (Section II), the Semantic Answer Evaluator performs the automated evaluation of the answers produced by the RAG pipeline against the expected answer.

RAGProbe is implemented in Python and is available online[9]. We use GPT-4 [25] to generate question-answer pairs for scenarios S1-S5, enhancing the quality of the generated pairs. For scenario S6, we use GPT-3.5-Turbo to reduce costs, as S6 involves iterative sampling of chunks and repeated application of the LLM.

V. Evaluation

We evaluated our concept of evaluation scenarios by a manual inspection of the generated question-answers and a quantitative study on the effectiveness for finding defects in RAG pipelines. We also compared our approach against current state-of-the-art, RAGAS [4]. The following research questions guided our evaluation:

- **RQ1**: How reliable are the generated question-answer pairs and evaluation metrics?
- **RQ2**: How effective are the evaluation scenarios in exposing failure rates in open-source RAG pipelines?
- **RQ3**: How does our approach compare to existing state-of-the-art approaches?

A. Datasets

To answer our research questions we needed two different datasets, a) a set of corpus of documents to form the input to RAGProbe , and b) a set of RAG implementations to evaluate.

[8]https://github.com/openai/evals/blob/main/evals/registry/modelgraded/closedqa.yaml

[9]https://figshare.com/s/ba57959589fd6c462d0e

TABLE V
OVERVIEW OF THE OPEN-SOURCE RAG PIPELINES WITH DEFAULT SETTINGS. NOTE: OUR APPROACH IS A BLACK BOX TESTING APPROACH.

Repo Name	Stars	Programming Language	Large Language Model	Embedding
Quivr	31,800	Python	gpt-3.5-turbo	text-embedding-ada-002
Danswer	9,000	Python	gpt-3.5-turbo-16k-0613	intfloat/e5-base-v2
Ragflow	4,400	Python	gpt-3.5-turbo	text-embedding-ada-002
Verba	2,100	Python	gpt-4-1106-preview	text-embedding-ada-002
Rag-stack	1,400	Python	ggml-gpt4All-J-v1.3-groovy	all-MiniLM-L6-v2

1) Corpus Datasets: For a set of corpus, we use i) Qasper [26], ii) Google Natural Questions (NQ) [27], and iii) MS Marco [28]. Qasper is a dataset of 1,585 scientific research papers. Google NQ corpus is a question answering dataset with 307,373 training examples. These documents are from Wikipedia pages. MS Marco is a dataset of real web documents using the Bing search engine. MS Marco dataset contains 3.2 million documents and 8.8 million passages. From all three datasets, we used documents that could be downloaded and indexed by our selected RAG pipelines. We noticed that some URLs in MS Marco dataset were obsolete and no longer valid, so we excluded those URLs when downloading documents. Qasper dataset contains of PDF files, Google NQ dataset contains of HTML files, and MS Marco dataset contains a combination of PDF and HTML files. With respect to domains, Qasper dataset belongs to academic or scientific research domain which has scientific research papers. Both Google NQ and MS Marco datasets are open-domain, meaning they do not pertain to any specific domain. All documents were converted to plain text before ingestion.

2) Open-source RAG Repositories: We selected 5 open-source RAG repositories from GitHub (see Table V). Our selection criteria for RAG repositories was 1) contains keywords 'retrieval augmented generation' or 'rag', 2) timeline (the repository must be created in the last 5 years), 3) popularity (the repository must have at least 1000 stars), 4) activeness (the repository must have at least a commit in the last 2 years), 5) programming language (the repository doesn't necessarily have to be Python-specific), 6) functionality (the repository should support uploading documents, the repository should not test RAG pipelines, and the repository does not require code changes to upload documents), 7) purpose (the repository should not be libraries or library wrappers or API wrappers), and 8) language (the repository description should be written in English).

The search term "retrieval augmented generation" yielded 1.2k projects, while the term "RAG" returned 36.4k projects. Out of these GitHub projects, repositories with more than 1000 stars and created after 2019, resulted in 53 repositories. Out of these 53 repositories, project description was written in English for 50 repositories. We manually inspected each of these repositories to check whether 1) the repository supports uploading documents, 2) the repository does not require changing code to upload documents, 3) the repository does not evaluate RAG pipelines, and 4) the repository is not a library or a library wrapper or an API wrapper (For example, Haystack[10], RAGatouille[11] were excluded because they were libraries). Our selection criteria resulted in 6 repositories, and we manually set up all these 6 repositories for evaluation. We failed to set up one repository, superagent[12] due to complexity with indexing documents. This resulted in 5 RAG repositories, which are 1) Quivr, 2) Danswer, 3) Ragflow, 4) Verba, and 5) Rag-stack. Table V summarizes the selected RAG repositories for evaluation. We used default settings of these repositories to execute the generated questions. Differences across the five RAG pipelines include various large language models, embedding models, frameworks (e.g., Langchain, LlamaIndex), vector databases (e.g., Weaviate), and indexing strategies.

B. Method

1) RQ1: How reliable are the generated question-answer pairs and evaluation metrics?: Our approach relies on large language models for two tasks: 1) the generation of question-answer pairs, and 2) the comparison of generated answers with an answer from a RAG pipeline. To understand how LLMs impact our approach, we manually inspected the generated outputs [29], [30]. For evaluation scenarios S1-S5, we manually inspected the generated questions and answers, as we used OpenAI's existing evals for the comparison. When using OpenAI evals for S6, we got incorrect results, requiring a new evaluation metric. For S6, we manually inspected the comparison results instead. Note that all answers to S6 questions should be a refusal to answer, so the ground truth answer does not need to be generated by RAGProbe .

We randomly sampled 75 question-answer pairs generated by RAGProbe for S1-S5, with 15 question-answer pairs for each scenario. Three authors manually annotated whether the chunk includes the question and the answer. If both criteria are met (i.e, question in the chunk, answer in the chunk), then the question-answer pair is considered as reliable. An initial session was held among annotators to establish common understanding by collaboratively labelling 10 question-answer pairs. After labelling is done independently by three authors, we calculate Fleiss' kappa inter-rater reliability score [31] to measure agreement across labelling.

For S6, there are three possible outcomes from comparing an answer from a RAG pipeline with the ground truth: a) refuse to answer, b) provide an answer, or c) refuse to answer but provide an answer anyway. We customised the OpenAI evals template to handle these variations by including the negative rejection metric, a definition, and only producing a

[10]https://github.com/deepset-ai/haystack
[11]https://github.com/bclavie/RAGatouille
[12]https://github.com/superagent-ai/superagent

Yes/No response. We also modified the prompt to take only answers when making a comparison. One author evaluated 150 comparisons (30 from each of the 5 RAG pipelines).

2) RQ2: How effective are the evaluation scenarios in exposing failure rates in open-source RAG pipelines?: Unlike state-of-the-art techniques that randomly generate question-answer pairs, evaluation scenarios enable fine grain control over the types of questions to generate. We evaluated the effectiveness of the 6 evaluation scenarios we have identified by detecting failures in 5 open-source RAG pipelines. Further, we investigated how failure rates vary across the different datasets.

We randomly sampled 10 documents from each dataset of Qasper, Google NQ, and MS Marco for generation of question-answer pairs. Then, we converted the selected documents into a text file (both PDF and HTML files). We automatically generated 30 questions for each evaluation scenario by creating one question from each of the 30 documents sampled across three datasets. This resulted in 180 questions to be executed across one RAG pipeline. Then, we captured the generated responses and evaluated against the expected responses as per the evaluation metrics. Finally, we calculated the failure rate per evaluation scenario. We also analysed failure rate per evaluation scenario per RAG pipeline.

> Failure is any answer from the RAG pipeline that does not pass one or more evaluation metrics. The LLM is used to assess pass/fail for each criteria given the ground-truth and the generated-answer.

Further, we grouped the generated questions per dataset. Then, we calculated the total number of failing questions per dataset. Further, we calculated the total number of RAG pipelines, that failed to answer all 10 questions from a dataset of a given scenario.

3) RQ3: How does our approach compare to existing state-of-the-art approaches?: To answer RQ3, we compared our approach to a state-of-the-art approach, RAGAS [4]. We initially searched the literature for tools used to evaluate RAG pipelines, focusing specifically on those that only require a collection of documents. Then, we generated questions using the selected tools and executed against the selected 5 open-source RAG pipelines. We compared failure rate against our approach versus the state-of-the-art approach. From the generated questions, we performed a statistical test (the Wilcoxon Signed-Rank Test [32]) to determine if there is a significant difference between the failure rates between the two approaches. All data and results are available online[13].

C. Results

1) How reliable are the generated question-answer pairs and evaluation metrics?: Fleiss' kappa showed an overall inter-rater reliability of 0.74, indicating substantial agreement between annotators [33]. This kappa score reflects consistency in the labelling made by the three annotators across the dataset. There were 61 out of 75 question-answer pairs where all three annotators marked as reliable. Out of 75 question-answer pairs, there were 7 question-answer pairs where all three annotators marked as unreliable. Out of these, 4 were from S5, 2 from S4, and 1 from S1. Three from S5 (combining multiple questions from a set of documents) were flagged as having neither the question nor answer available in the chunk. For example, a question was generated - *"What is the role of essential part in the analysis of many phenomena?"* and an answer was generated - *"Essential parts form an integral component in the analysis of many phenomena, however, their exact role is determined by the specific phenomenon being analyzed."*, even though the chunk mentions that *"...However, they form an essential part of the analysis of many phenomena and are consequently available in the implementation..."*. This example shows that both question and answer were fabricated. Table VI shows the breakdown of annotations, where all three annotators marked question-answer pairs as unreliable.

TABLE VI
BREAKDOWN OF QUESTION-ANSWER PAIRS, WHERE ALL THREE ANNOTATORS MARKED AS UNRELIABLE. Q INDICATES QUESTION IN THE CHUNK, AND A INDICATES ANSWER IN THE CHUNK.

	Annotator 1		Annotator 2		Annotator 3	
	Q	A	Q	A	Q	A
S1 QA Pair 1	Y	N	N	N	Y	N
S4 QA Pair 1	N	N	Y	N	Y	N
S4 QA Pair 2	Y	N	Y	N	Y	N
S5 QA Pair 1	N	N	N	N	N	N
S5 QA Pair 2	N	N	N	N	N	N
S5 QA Pair 3	N	N	N	N	N	N
S5 QA Pair 4	N	N	Y	N	N	N

During manual inspection of comparisons for S6, 100% of comparisons were correct. For S6, among 150 responses from RAG pipelines: 82 answered the question, 35 did not answer the question, 3 did not answer the question but provided a response afterwards, 14 described the provided context, 10 described the provided context and also responded to the question, and 6 responded context size exceeded error. One RAG pipeline (Verba) described the provided context despite not being relevant to the question. Some of these responses included an answer to the question. An example response from this RAG pipeline was: *"The provided context does not contain a direct explanation of evolutionary psychology. The context predominantly discusses the emergence of syntactic structures, the concept of duality of patterning in language development, and computational models of language evolution. Evolutionary psychology is a field that examines psychological traits as evolved adaptations due to natural selection....."*

> **Answer to RQ1:** 81% of the generated question-answer pairs were marked as reliable by all three annotators. 100% of negative rejection comparisons were correct.

2) How effective are the evaluation scenarios in exposing failure rates in open-source RAG pipelines?: Fig. 2 shows the summary of automatic execution of automatically generated

[13]https://figshare.com/s/ba57959589fd6c462d0e

questions. In total, there were 150 generated questions per scenario, that were executed across 5 RAG pipelines. Scenario S5 has the highest failure rate at 91%, followed by S4 at 78% and S6 at 69%, indicating these scenarios are the most problematic. Scenarios S1 and S2 show moderate failure rates at 45% and 40% respectively. Scenario S3 has the lowest failure rate at 29%, suggesting it encounters the fewest number of failing questions across 5 RAG pipelines.

As shown in Table VII, Quivr and Rag-stack exhibited the highest failure rates (100%) in scenarios S4 and S5. When asking questions for which the answer is not in the corpus (S6), Danswer and Rag-stack did not generate any correct responses. During the manual inspection, we found that Danswer and Rag-stack generated responses by referring to the large language model's trained knowledge, instead of answering "The system doesn't know the answer". Ragflow had the lowest failure rate (27%) with multiple-choice questions compared to all other scenarios for this RAG pipeline. Verba had the lowest failure rate (17%) when handling multiple-choice questions compared to all other scenarios for this RAG pipeline. Compared to 5 RAG pipelines, Rag-stack had the highest failure rates in all scenarios, indicating significant challenges in handling complex questions and specific information retrieval.

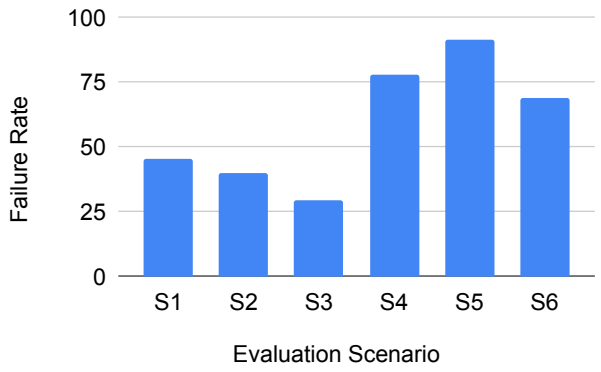

Fig. 2. Total failure rate combining all 5 RAG pipelines. The failure rate is calculated as the number of failures divided by the total number of questions.

TABLE VII
FAILURE RATES OF RAG PIPELINES ACROSS EVALUATION SCENARIOS

ID	Quivr	Danswer	Ragflow	Verba	Rag-stack
S1	17%	43%	50%	43%	73%
S2	20%	30%	43%	30%	77%
S3	17%	17%	27%	17%	70%
S4	100%	70%	80%	40%	100%
S5	100%	70%	97%	90%	100%
S6	17%	100%	80%	50%	100%

As shown in Fig. 3, MS Marco had the highest number of failing questions, whilst Google NQ had the lowest number of failing questions. 60%, 53%, and 64% failure rates were observed for Qasper, Google NQ, and MS Marco datasets respectively. The high failure rates across all datasets indicate that RAG pipelines struggle consistently with the complexity

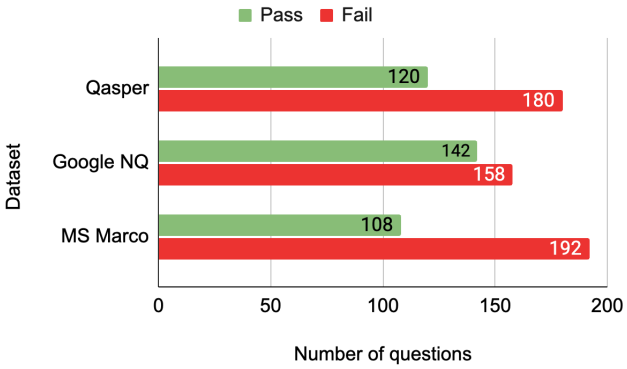

Fig. 3. Automated evaluation results per dataset.

TABLE VIII
NUMBER OF RAG PIPELINES FAILED ACROSS EVALUATION SCENARIOS AND DATASETS

ID	Qasper	Google NQ	MS Marco
S1	5/5	4/5	5/5
S2	4/5	5/5	5/5
S3	5/5	5/5	3/5
S4	5/5	5/5	5/5
S5	5/5	5/5	5/5
S6	4/5	5/5	5/5

and variability of questions in both academic and open-domain datasets.

Table VIII shows the number of RAG pipelines which have failed to produce correct responses across evaluation scenarios for each dataset. In summary, Google NQ and MS Marco have the highest number of failing scenarios, where all RAG pipelines failed in 5 scenarios. Qasper shows more variability, with one RAG pipeline passing in S2 and S6 scenarios.

Answer to RQ2: Evaluation scenarios had a failure rate of 29% - 91% across 5 RAG pipelines (30 questions each). The lowest failure rate was S3 - multiple-choice questions, and the highest was S5 - combining multiple questions from a set of documents.

3) How does our approach compare to existing state-of-the-art approaches?: By reviewing the literature, we identified two tools capable of generating question-answer pairs from documents: RAGAS [4] and ARES [5]. When setting up these tools, ARES [5] required an initial set of question-answer pairs. Therefore, we used RAGAS [4] to generate question-answer pairs for comparison with our approach. Out of the 180 questions generated by RAGAS, 24 had answers listed as "nan" (i.e., without valid answers). This indicates that RAGAS has limitations in accurately generating valid answers, affecting the utility of the generated data for downstream tasks.

Our approach, compared to the state-of-the-art approach, exposed increased failure rates across each dataset and across each RAG pipeline. The state-of-the-art approach revealed 37%, 21%, 45%, 19%, and 71% failure rates whereas our approach revealed 45%, 55%, 63%, 45%, and 87% failure rates across Quivr, Danswer, Ragflow, Verba, and Rag-stack respectively, as can be seen in Fig. 4. The state-of-the-art

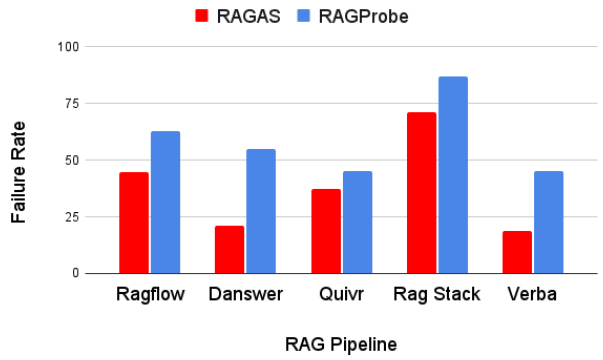

Fig. 4. Comparison of failure rates across RAG pipelines. A higher failure rate is better as it reveals more limitations of the RAG pipelines.

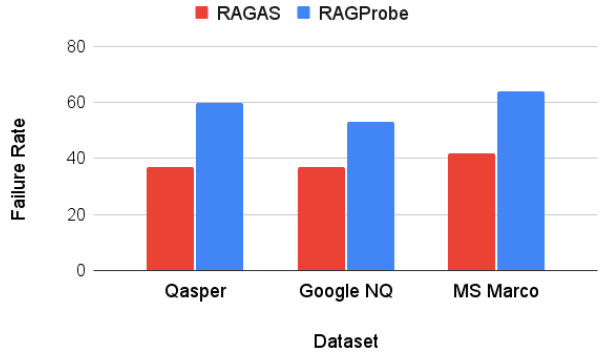

Fig. 5. Comparison of failure rates across datasets. A higher failure rate is better as it reveals more limitations of the RAG pipelines.

approach exposed 37%, 37%, and 42% failure rates, whereas our approach exposed 60%, 53%, and 64% failure rates for Qasper, Google NQ, and MS Marco datasets respectively, as shown in Fig. 5. When performed the Wilcoxon signed rank test on failure rate pairs, it resulted in p-values of 0.25, and 0.0625 per dataset and per RAG pipeline respectively. Both p-values being higher than the common significance level of 0.05, suggesting that there is no significant difference in the failure rates across different datasets and across different RAG pipelines. This implies that the observed differences in failure rates might be due to random chance rather than inherent differences in the datasets or RAG pipelines being evaluated. Consequently, our evaluation shows that the performance of the RAG pipelines is relatively consistent across different datasets and pipeline implementations.

Answer to RQ3: RAGProbe found 77% more failures on average per RAG pipeline and 53% more failures on average per dataset compared to state-of-the art RAGAS [4].

VI. Discussion

A. Evaluating RAG Pipelines

Even though automatic test case generation is widely used in software engineering field, they cannot be used in RAG evaluation due to interpolative geometrical aspect of LLMs. The software development of RAG pipelines in practice involves making many decisions that ultimately contribute towards its quality, i.e., about text splitter, chunk size, chunk overlap size, embedding model, large language model, vector store, distance metric for semantic similarity, top-k to rerank, reranker model, top-k to context, and prompt engineering. In real-world settings, these decisions are not grounded in methodologically sound evaluation practices but rather ad-hoc and driven by developers and product owners, constrained by project deadlines. Thus, robust and reliable evaluation of RAG pipelines relies on having variations of questions and by automating the entire process.

Our results from RQ1 indicate a high degree of reliability in the generated questions and answers. However, as new evaluation scenarios are implemented, a method will be needed to semi-automate the evaluation of both the quality of the question-answer pairs and evaluation metrics. As demonstrated by needing a different evaluation metric for S6 than S1-S5, evaluation scenarios are a flexible way to reuse and leverage new evaluation metrics.

Results from RQ2 reveal that the most effective evaluation scenarios with more than 50% failure rates include: 1) a question combining multiple questions for which answers are in a single document, 2) a question combining multiple questions for which answers are in a set of documents, and 3) a question for which answer is not in the document corpus. This suggests the need for careful design of evaluation scenarios for the thorough evaluation of a RAG pipeline going into production. The comparison of our approach with state-of-the-art for RQ3 highlights the importance of having diverse evaluation scenarios. For example, the RAG pipeline Quivr had a close failure rate (37% compared to 45%) with our approach. However, it's clear *that having targeted designed evaluation scenarios is more effective than randomly generating question-answer pairs.*

B. Threats to Validity

1) Internal Validity: When evaluating the RAG generated responses against the expected responses, we considered different evaluation metrics such as correctness, relevance, completeness, consistency, and negative rejection. However, other evaluation metrics such as faithfulness and bias were not included. To address this, the existing Semantic Answer Evaluator (as shown in Fig. 1) can be extended by defining additional metrics to evaluate the responses.

2) External Validity: In the current study, we have considered only one LLM (i.e., GPT) from OpenAI. There are other LLMs such as LLaMA, Alpaca, Vicuna, and Falcon. This limits the generalizability of our results. Investigating additional LLMs would provide a more comprehensive understanding and improve the external validity of our findings.

3) Construct Validity: We investigated RAG pipelines which use various language models, embedding models, and chunking strategies. Comparing results across these differing

default settings ensures that the variability in these configurations does not bias our findings. However, consistent settings with the same large language model, embedding model, and chunking strategy might have led to different results. To generate question-answer pairs, we have not considered other prompting techniques such as chain-of-thought, and graph-of-thought. These techniques could have produced different patterns in question-answer pairs. Exploring different prompting techniques is left for future work.

C. Implications

1) Researchers: Based on identified evaluation scenarios, researchers can explore additional evaluation scenarios to create more variations in question-answer pairs. Further, researchers can include additional evaluation metrics such as faithfulness and context relevance [4], [5] for evaluating answers. Commercially available large language models such as GPT-3.5 and GPT-4 were selected for the purpose of this study based on their popularity [34]–[37] (i.e. under extensive investigation in recent research). Open-source LLMs were excluded from the study as they (i) underperform [38], (ii) require extensive infrastructure to operate, and (iii) have been trained on smaller datasets [39]. However, researchers can investigate how open-source LLMs could perform when they are utilised to generate question-answer pairs.

2) Developers: As shown in Fig. 1, developers can consider the proposed approach to evaluate RAG pipelines that they are developing, and pinpointing the failure points. As can be seen in Table IX, developers can apply suggested fixes based on failing question-answer pairs. Automated evaluation for RAG pipelines offers benefits, including increased efficiency by reducing manual effort, ensuring consistent and reliable evaluation by minimizing human errors, and enabling integration into CI/CD pipelines. Understanding and fixing these failure points is essential for the development of effective and robust RAG pipelines. Our approach provides a guidance for developers for evaluating RAG pipelines in information retrieval and generation contexts.

TABLE IX
OVERVIEW OF EVALUATION SCENARIOS WITH THE RECOMMENDED FIXES BASED ON THE LITERATURE

ID	Recommendation of Fixes
S1	Upgrading embedding model [40], [41]
S2	Upgrading embedding model [40], [41]
S3	Upgrading LLM [16], Fine-tuning [16], [40], [42]
S4	Query rewriting [41], [43], [44]
S5	Query rewriting [41], [43], [44]
S6	Prompt engineering during generation [40], [43]

VII. Related Work

Prior studies have proposed reference-free evaluation such as RAGAS [4] and ARES [5], which similarly use LLM-generated data to evaluate RAG pipelines. While they focus on evaluation metrics such as contextual relevance, faithfulness, and answer relevance, our approach evaluates RAG responses for correctness, relevance, completeness, consistency, and negative rejection. Unlike InspectorRAGet [9], which does not generate questions, our method generates question-answer pairs from scratch without relying on initial data, avoiding potential error propagation. Codium-AI [45] generates unit tests using LLMs and requires an initial set of unit tests from developers, whereas our approach offers six evaluation scenarios and does not require an initial set of questions. Since Codium-AI generates unit tests based on an initial set, there is a risk of propagating errors if the initial unit tests are flawed.

The effectiveness of RAG pipelines is assessed through various methods, including evaluating individual components and their integration, using benchmarks, and applying different evaluation frameworks. For instance, PipeRAG [46] system integrates pipeline parallelism, flexible retrieval intervals, and a performance model to reduce generation latency and enhance quality. The eRAG method [13] evaluates document relevance by utilizing each document individually and comparing the output against downstream task ground truth labels. ClapNQ [47] provides a benchmark dataset focusing on long-form question answering to highlight areas for improvement in RAG pipelines. Another research [48] evaluates information retrieval component of RAG pipelines and found that having irrelevant documents in the corpus can unexpectedly improve performance of a RAG pipeline.

VIII. Conclusion and Future Work

In this paper, we present a novel approach, evaluation scenarios, used to assist in the evaluation of RAG pipelines. We implemented our approach in RAGProbe and carried out an evaluation against 5 open-source RAG implementations, utilising six different evaluation scenarios. Our approach generates context-specific questions and answers from a corpus of documents for evaluating a RAG pipeline without needing an initial question-answer set and enabling automated answer evaluation. 81% of question-answer pairs generated by RAGProbe were reliable, and the *evaluation scenarios* related to multiple questions posed in a single set (S4 & S5) resulted in the highest failure rates (91% and 78%). Additionally, our approach outperformed the existing state-of-the-art by having 77% higher failure rate on average per RAG pipeline. This addresses the critical requirement of robust evaluation of RAG pipelines by introducing challenging instances that mirror real-world complexities.

Future work includes cataloguing high impact evaluation scenarios, assessing flakiness introduced by using LLMs, using a jury of LLMs instead of a single LLM to evaluate expected responses and RAG pipeline generated responses, and providing suggested fixes (mined from the literature and empirically validated). We also plan to do empirical investigation into the impact of chosen configurations (e.g. large language model, embedding model, chunk size) for RAG pipelines. This work will provide an empirical basis for the design of robust and reliable RAG pipelines. The exact behavior of a RAG pipeline is unknowable without empirical evaluation. Therefore, irrespective of internal configurations, identifying evaluation scenarios and generating associated question-answer pairs plays a vital role in evaluating RAG pipelines.

REFERENCES

[1] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity," *arXiv preprint arXiv:2403.14403*, 2024.

[2] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective retrieval augmented generation," *arXiv preprint arXiv:2401.15884*, 2024.

[3] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," *arXiv preprint arXiv:2310.11511*, 2023.

[4] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," *arXiv preprint arXiv:2309.15217*, 2023.

[5] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, "Ares: An automated evaluation framework for retrieval-augmented generation systems," *arXiv preprint arXiv:2311.09476*, 2023.

[6] S. Barnett, S. Kurniawan, S. Thudumu, Z. Brannelly, and M. Abdelrazek, "Seven failure points when engineering a retrieval augmented generation system," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, 2024, pp. 194–199.

[7] K. Wu, E. Wu, and J. Zou, "How faithful are rag models? quantifying the tug-of-war between rag and llms' internal prior," *arXiv preprint arXiv:2404.10198*, 2024.

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

[9] K. Fadnis, S. S. Patel, O. Boni, Y. Katsis, S. Rosenthal, B. Sznajder, and M. Danilevsky, "Inspectorraget: An introspection platform for rag evaluation," *arXiv preprint arXiv:2404.17347*, 2024.

[10] Y. Mao, P. He, X. Liu, Y. Shen, J. Gao, J. Han, and W. Chen, "Generation-augmented retrieval for open-domain question answering," *arXiv preprint arXiv:2009.08553*, 2020.

[11] Z. Feng, X. Feng, D. Zhao, M. Yang, and B. Qin, "Retrieval-generation synergy augmented large language models," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 661–11 665.

[12] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, "Raft: Adapting language model to domain specific rag," *arXiv preprint arXiv:2403.10131*, 2024.

[13] A. Salemi and H. Zamani, "Evaluating retrieval quality in retrieval-augmented generation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 2395–2400.

[14] Z. Rasool, S. Barnett, D. Willie, S. Kurniawan, S. Balugo, S. Thudumu, and M. Abdelrazek, "Llms for test input generation for semantic applications," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, 2024, pp. 160–165.

[15] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[16] A. Balaguer, V. Benara, R. L. de Freitas Cunha, R. d. M. Estevão Filho, T. Hendry, D. Holstein, J. Marsman, N. Mecklenburg, S. Malvar, L. O. Nunes *et al.*, "Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture," *arXiv e-prints*, pp. arXiv–2401, 2024.

[17] K. Zhu, Y. Luo, D. Xu, R. Wang, S. Yu, S. Wang, Y. Yan, Z. Liu, X. Han, Z. Liu *et al.*, "Rageval: Scenario specific rag evaluation dataset generation framework," *arXiv preprint arXiv:2408.01262*, 2024.

[18] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 754–17 762.

[19] Z. Rasool, S. Kurniawan, S. Balugo, S. Barnett, R. Vasa, C. Chesser, B. M. Hampstead, S. Belleville, K. Mouzakis, and A. Bahar-Fuchs, "Evaluating llms on document-based qa: Exact answer selection and numerical extraction using cogtale dataset," *Natural Language Processing Journal*, p. 100083, 2024.

[20] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in Neural Information Processing Systems*, vol. 36, pp. 46 595–46 623, 2023.

[21] C.-H. Chiang and H.-y. Lee, "Can large language models be an alternative to human evaluations?" *arXiv preprint arXiv:2305.01937*, 2023.

[22] A. Sottana, B. Liang, K. Zou, and Z. Yuan, "Evaluation metrics in the era of gpt-4: reliably evaluating large language models on sequence to sequence tasks," *arXiv preprint arXiv:2310.13800*, 2023.

[23] A. Myrzakhan, S. M. Bsharat, and Z. Shen, "Open-llm-leaderboard: From multi-choice to open-style questions for llms evaluation, benchmark, and arena," *arXiv preprint arXiv:2406.07545*, 2024.

[24] K. Okuda and S. Amarasinghe, "Askit: Unified programming interface for programming with large language models," in *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2024, pp. 41–54.

[25] OpenAI, "Gpt-4 technical report," 2023.

[26] P. Dasigi, K. Lo, I. Beltagy, A. Cohan, N. A. Smith, and M. Gardner, "A dataset of information-seeking questions and answers anchored in research papers," *arXiv preprint arXiv:2105.03011*, 2021.

[27] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[28] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "Ms marco: A human-generated machine reading comprehension dataset," 2016.

[29] E. Oro, F. M. Granata, A. Lanza, A. Bachir, L. De Grandis, and M. Ruffolo, "Evaluating retrieval-augmented generation for question answering with large language models," 2024.

[30] X. Yu, H. Cheng, X. Liu, D. Roth, and J. Gao, "Reeval: Automatic hallucination evaluation for retrieval-augmented large language models via transferable adversarial attacks," in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 1333–1351.

[31] J. L. Fleiss, B. Levin, M. C. Paik *et al.*, "The measurement of interrater agreement," *Statistical methods for rates and proportions*, vol. 2, no. 212-236, pp. 22–23, 1981.

[32] R. F. Woolson, "Wilcoxon signed-rank test," *Encyclopedia of Biostatistics*, vol. 8, 2005.

[33] T. R. Nichols, P. M. Wisner, G. Cripe, and L. Gulabchand, "Putting the kappa statistic to use," *The Quality Assurance Journal*, vol. 13, no. 3-4, pp. 57–61, 2010.

[34] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, "Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design," in *Generative AI for Effective Software Development*. Springer, 2024, pp. 71–108.

[35] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[36] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *ACM Transactions on Software Engineering and Methodology*, 2023.

[37] Z. Zhu, Y. Xue, X. Chen, D. Zhou, J. Tang, D. Schuurmans, and H. Dai, "Large language models can learn rules," *arXiv preprint arXiv:2310.07064*, 2023.

[38] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[39] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[40] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[41] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, "Query rewriting for retrieval-augmented large language models," *arXiv preprint arXiv:2305.14283*, 2023.

[42] S. Siriwardhana, R. Weerasekera, E. Wen, and S. Nanayakkara, "Fine-tune the entire rag architecture (including dpr retriever) for question-answering," *arXiv preprint arXiv:2106.11517*, 2021.

[43] X. Li, J. Jin, Y. Zhou, Y. Zhang, P. Zhang, Y. Zhu, and Z. Dou, "From matching to generation: A survey on generative information retrieval," *arXiv preprint arXiv:2404.14851*, 2024.

[44] J. Tan, Z. Dou, Y. Zhu, P. Guo, K. Fang, and J.-R. Wen, "Small models, big insights: Leveraging slim proxy models to decide when and what to retrieve for llms," *arXiv preprint arXiv:2402.12052*, 2024.

[45] N. Alshahwan, J. Chheda, A. Finogenova, B. Gokkaya, M. Harman, I. Harper, A. Marginean, S. Sengupta, and E. Wang, "Automated unit test improvement using large language models at meta," in *Companion*

Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, 2024, pp. 185–196.

[46] W. Jiang, S. Zhang, B. Han, J. Wang, B. Wang, and T. Kraska, "Piperag: Fast retrieval-augmented generation via algorithm-system co-design," *arXiv preprint arXiv:2403.05676*, 2024.

[47] S. Rosenthal, A. Sil, R. Florian, and S. Roukos, "Clapnq: Cohesive long-form answers from passages in natural questions for rag systems," *arXiv preprint arXiv:2404.02103*, 2024.

[48] F. Cuconasu, G. Trappolini, F. Siciliano, S. Filice, C. Campagnano, Y. Maarek, N. Tonellotto, and F. Silvestri, "The power of noise: Redefining retrieval for rag systems," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 719–729.