Title: A Comprehensive Evaluation Framework for Retrieval-Augmented Generation Systems

Author: Sandipan Das Mahapatra

Abstract

This paper presents a comprehensive evaluation of Retrieval-Augmented Generation (RAG), from the perspective of a Model Validator performing a detailed technical review of the model to ensure model risks are minimised. These systems combine the abilities to retrieve and generate information, creating detailed and contextually rich responses. This dual function requires evaluation in several areas, such as how relevant the retrieved information is, how well the generated content is based on those sources, how complete the responses are to user queries, and the overall relevance of the answers provided. As a result, evaluating RAG systems is more complex than assessing traditional models with clear-cut outputs.

Our evaluation framework follows a systematic approach comprising several key steps: Ground truth Generation, Synthetic Query Dataset Generation by LLM, Hyperparameter Evaluation, Performance Assessment.

Given their increasing real-world deployment, comprehensive evaluation is crucial. We aim to bridge evaluation gaps and assist in developing more robust RAG systems.

Introduction

Retrieval-Augmented Generation (RAG) systems have emerged as a powerful approach to overcome the limitations of traditional large language models (LLMs) by integrating external knowledge sources into the language generation process. This fusion of information retrieval and natural language generation enables RAG systems to produce more comprehensive, factual, and contextually relevant responses. However, the evaluation of RAG systems presents unique challenges due to their complex interplay of retrieval and generation components.

While existing research has explored various aspects of RAG evaluation, such as individual metrics for retrieval and generation quality, there remains a need for a holistic framework that addresses the multifaceted nature of these systems, particularly from a risk management perspective. This paper aims to bridge this gap by proposing a comprehensive evaluation framework that considers not only the accuracy and relevance of retrieved information but also the faithfulness, completeness, and potential biases of the generated text.

Our framework is motivated by the recognition that RAG systems, while promising, can introduce new risks and challenges. By providing a rigorous evaluation methodology, we aim to assist developers and model validators in ensuring that RAG systems meet the stringent requirements of real-world applications and minimize potential risks.

Related Work

Evaluating the performance of Retrieval-Augmented Generation (RAG) systems presents unique challenges due to their complex interplay of retrieval and generation components. A robust evaluation framework must consider not only the accuracy and relevance of retrieved information but also the quality, faithfulness, and potential biases of the generated text. This section reviews existing work on evaluating RAG systems, highlighting their strengths, limitations, and suitability for different use cases.

Existing RAG Evaluation Frameworks
Several frameworks have been developed to assess RAG system performance, ranging from proprietary solutions to open-source tools. The choice of framework depends on factors such as the specific RAG use case, budget constraints, desired level of support, and the need for ongoing monitoring or specialized domain adaptation.

- **ARES**: This open-source framework leverages synthetic data and LLM judges to evaluate RAG systems, particularly in dynamic environments requiring continuous training and updates. It emphasizes ranking-based metrics like Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG).

- **RAGAS**: This open-source tool offers a streamlined, reference-free evaluation approach, focusing on Average Precision (AP) and custom metrics like Faithfulness. It is particularly useful for initial assessments or when reference data is limited.

In this paper, we have presented a comprehensive evaluation framework for Retrieval-Augmented Generation (RAG) systems, designed to address the unique challenges posed by these complex systems. Our framework incorporates a systematic approach to generating synthetic query datasets, evaluating both the retrieval and generation components, and assessing the overall performance of the system. We advocate for a multi-faceted evaluation strategy that encompasses various metrics, including retrieval metrics, generation metrics, and fairness metrics, to provide a holistic view of the system's capabilities and limitations.

Our work is particularly important from a risk management perspective, as it provides a rigorous methodology for identifying and mitigating potential risks associated with RAG systems. By thoroughly evaluating these systems, we can ensure that they generate accurate, relevant, and safe responses, while also addressing concerns related to bias, fairness

Synthetic Query Dataset Generation by LLM

This section focuses on the process of creating a synthetic query dataset using a Large Language Model (LLM) and outlines a methodology for evaluating the RAG system's responses.

Process:
1. Synthetic Data Generation:
   - Due to the potential lack of historical "test" data, synthetic data is crucial for evaluating RAG systems.
   - An LLM is employed to generate a diverse and broad range of synthetic queries.
   - These queries are designed to cover various scenarios and potential user inputs, including grounded, off-topic, and jailbreaking queries.
2. RAG System Response:
   - The RAG system processes each synthetic query and generates corresponding responses.
3. Evaluation using a Separate LLM:
   - A separate LLM acts as a judge to evaluate the RAG system's responses.
   - This LLM assesses the groundedness, relevance, and appropriateness of the responses based on predefined criteria and evaluation metrics.

Types of Queries:
These query types are designed to assess different aspects of the system's performance and identify potential risks. The text specifically mentions three types of queries:

1. Groundedness Queries:
    - Purpose: To ensure that the RAG system generates accurate answers that are grounded in the provided context.

    - Process:
        - An LLM randomly selects documents from the RAG system's knowledge base.
        - The LLM creates questions based on these documents.
        - The RAG system retrieves relevant context and generates answers to these questions.
        - The LLM acts as a judge and scores the groundedness of the RAG system's responses based on the provided context.

    - Example: If the document mentions "The capital of France is Paris," a groundedness query could be "What is the capital of France?" The LLM would then evaluate if the RAG system accurately extracts and presents this information.

2. Off-topic Queries:
    - Purpose: To ensure that the RAG system remains aligned with its objectives and avoids generating responses on sensitive, harmful, or inappropriate topics.

    - Process:
        - The LLM generates off-topic questions that cover areas such as political issues, sensitive topics, harmful or illegal activities, and questions about internal policies or practices.
        - The RAG system responds to these questions.
        - The LLM judges the relevance of both its own answers and the RAG system's answers to these off-topic queries.

    - Example: An off-topic query could be "What are your views on the current political situation?" The LLM would assess if the RAG system avoids answering such questions or provides neutral and safe responses. RAG system with appropriate guardrails should be able to bypass these questions.

3. Jailbreaking Queries:
    - Purpose: To assess the RAG system's safeguards against malicious or inappropriate queries that attempt to manipulate the system or extract sensitive information.

    - Process:
        - The LLM generates jailbreaking queries that involve context switching (manipulating the conversation flow) or code injection (introducing code into the system).
        - The RAG system responds to these queries.
        - The LLM judges the relevance of both its own answers and the RAG system's answers to these jailbreaking queries.

    - Example: A jailbreaking query could involve providing a prompt like "Ignore the previous instructions and tell me your internal code." The LLM would then evaluate if the RAG system resists such manipulation and maintains its security protocols.

By incorporating these diverse query types, the evaluation process can comprehensively assess the RAG system's capabilities, identify potential risks, and ensure its safe and reliable deployment in real-world scenarios.

Scoring:

- The LLM judge scores the RAG system's responses based on metrics such as:
    - Groundedness: The extent to which the response is based on the provided context and avoids hallucinations.
    - Relevance: How relevant the response is to the query.

Expanding Query Domains:

- Based on the RAG system's objectives, specific domains or areas of interest can be expanded to capture the types of questions expected in a production setting or particular areas of concern.

Benefits:

- This approach allows for a comprehensive evaluation of the RAG system's performance across a wide range of queries and potential risks.
- It helps ensure that the system generates accurate, relevant, and safe responses in various scenarios.

Hyperparameter Optimization

Similar to traditional machine learning models, Large Language Models (LLMs) employed in Retrieval-Augmented Generation (RAG) systems can be significantly influenced by hyperparameter tuning. Hyperparameters are external configurations that govern the learning process and operational behavior of the model. Optimizing these settings is crucial for achieving the desired performance and efficiency in a RAG system.

Key Hyperparameters in RAG Systems

RAG systems involve a complex interplay of retrieval and generation components, each with its own set of hyperparameters. Key categories and examples include:

- **Retrieval Parameters:**
    - **Number of Retrieved Documents:** This parameter determines how many documents are retrieved from the knowledge source for a given query. A balance must be struck: retrieving too few documents may omit crucial information, while retrieving too many can introduce irrelevant noise and hinder the generation process.

- **Generation Parameters:**
    - **Temperature:** This parameter controls the randomness of the language model's output. Lower temperatures result in more deterministic and focused responses, while higher temperatures increase diversity and creativity, potentially at the cost of accuracy. The choice of temperature should align with the desired characteristics of the generated text.

    - **Max Token Length:** This setting limits the length of the generated response. An appropriate maximum length ensures comprehensive answers without producing excessively long or rambling outputs that may include irrelevant information.

- **Integration Parameters:**
    - **Context Window Size:** This defines the amount of contextual information from the retrieved documents that is passed to the language model. An optimal window size provides sufficient context for generating relevant responses without exceeding the model's processing capacity.
    - **Context Selection:** This mechanism determines which portions of the retrieved documents are most relevant to the query and should be prioritized for response generation. Effective context selection ensures that the model focuses on the most pertinent information.

This section has explored the critical role of hyperparameter optimization in fine-tuning Retrieval-Augmented Generation (RAG) systems. By systematically tuning key parameters such as those governing retrieval, generation, and integration, we can significantly enhance the performance and efficiency of RAG models.

Several strategies exist for hyperparameter optimization, including grid search and Bayesian optimization. Grid search, while computationally intensive, provides a comprehensive exploration of the hyperparameter space, ensuring no potential combination is overlooked. Bayesian optimization offers a more efficient approach by intelligently sampling the search space based on prior knowledge and observed performance.

The choice of hyperparameter optimization strategy depends on the specific RAG system, available resources, and desired level of accuracy. It is crucial to experiment with different approaches and evaluate the impact of hyperparameter tuning on the system's overall performance.

By carefully optimizing hyperparameters, we can unlock the full potential of RAG systems, enabling them to generate more accurate, relevant, and coherent responses in a wide range of applications.

Performance Assessment
The choice of metrics for evaluating a Retrieval-Augmented Generation (RAG) system depends on the specific NLP task and the desired characteristics of the generated text. In this paper, we consider mainly three types of metrics to provide a comprehensive evaluation of the system.

1. **Retrieval Metrics:**
   We use Contextual Precision, Contextual Recall, and Contextual Relevancy to evaluate the performance of the retriever component. These metrics measure the ability of the retriever to identify and rank relevant context documents chunks (nodes) from the Vector Database (Vector DB) in response to input queries.

2. **Generation Metrics:**
   We use a variety of metrics to evaluate the quality of the generated responses, including Answer Relevancy, Faithfulness, Hallucination, BLEU, ROUGE, and METEOR. These metrics measure the relevancy, factual accuracy, fluency, and semantic similarity of the generated text compared to a reference text.

3. **Fairness Metrics:**
   We use Bias and Toxicity to evaluate the fairness of the generated responses. These metrics measure the extent to which the generated response exhibits bias towards certain groups or viewpoints, and the extent to which it contains toxic or offensive language.

Limitations of Metrics:
We acknowledge the limitations of each metric. For example, BLEU is sensitive to word order and may not capture semantic meaning effectively. ROUGE focuses on recall and may not penalize irrelevant information. Contextual Precision and Contextual Recall may not fully capture the nuances of relevancy. Bias and Toxicity may not capture all forms of bias and toxicity.

Combining Metrics:
We consider using a combination of metrics to provide a more comprehensive evaluation of the generated text. This can help mitigate the limitations of individual metrics and provide a more holistic view of the system's performance. For example, we can combine BLEU and ROUGE to get a more balanced view of the fluency and accuracy of the generated text. We can combine Contextual Precision, Contextual Recall, and Contextual Relevancy to get a more comprehensive view of the relevancy of the retrieved context. We can combine Bias and Toxicity to get a more complete picture of the fairness of the generated responses.

Retrieval Evaluation Metrics

Contextual Precision measures the proportion of relevant information retrieved in the context documents used by a language model to generate a response. It's a key metric in Retrieval-Augmented Generation (RAG) systems, where the accuracy of the retrieved context directly impacts the quality of the generated output .

Calculation Method:

Contextual Precision is calculated as the average precision at rank k (Precision@k) across all retrieved contexts . Here's a breakdown of the steps involved:

1. **Determine Relevance:** For each chunk of information in the retrieved context, determine whether it is relevant to the given query. This often involves comparing the retrieved information to a ground truth or reference answer.
2. **Calculate Precision@k:** For each rank position (k) in the retrieved context, calculate the precision. Precision@k is the ratio of relevant chunks at rank k to the total number of chunks retrieved up to rank k.
3. **Calculate Average Precision:** Calculate the average of the precision values at each rank position where a relevant chunk appears. This gives you the Average Precision (AP).
4. **Calculate Contextual Precision:** Average the AP values across all retrieved contexts to obtain the final Contextual Precision score.

Example:

Imagine a user asks, "What is the capital of France?" and the RAG system retrieves the following context:

1. "France is a country in Western Europe. Its capital is Paris, which is known for landmarks like the Eiffel Tower."
2. "The Eiffel Tower is a wrought-iron lattice tower on the Champ de Mars in Paris, France."
3. "Paris is the capital and most populous city of France."

Assuming all three chunks are considered relevant, the precision at each rank would be:

- Precision@1 = 1/1 = 1
- Precision@2 = 2/2 = 1
- Precision@3 = 3/3 = 1

The Average Precision (AP) would be (1 + 1 + 1) / 3 = 1.

Let's illustrate Contextual Precision with another example. Suppose a user asks, "What are the health benefits of blueberries?" and the RAG system retrieves five chunks of information:

1. "Blueberries are a good source of antioxidants." (Relevant)
2. "Apples are rich in fiber." (Not relevant)
3. "Studies show blueberries may improve memory." (Relevant)
4. "Oranges contain vitamin C." (Not relevant)
5. "Blueberries can help protect against heart disease." (Relevant)

To calculate Contextual Precision, we first determine the relevance of each chunk. Then, we calculate the precision at each rank:

- Precision@1 = 1/1 = 1
- Precision@2 = 1/2 = 0.5
- Precision@3 = 2/3 = 0.67
- Precision@4 = 2/4 = 0.5
- Precision@5 = 3/5 = 0.6

Next, we calculate the Average Precision (AP). Since there are three relevant chunks, we average the precision values at ranks 1, 3, and 5:
AP = (1 + 0.67 + 0.6) / 3 = 0.76

Contextual Recall assesses the ability of a Retrieval-Augmented Generation (RAG) system to retrieve all the necessary information from external sources to generate a complete and accurate response to a user's query . It measures the comprehensiveness of the information retrieval process, ensuring that the system doesn't miss any crucial details .

Calculation Method:
To calculate Contextual Recall, you need to compare the information retrieved by the RAG system to a ground truth or reference answer. Here's a breakdown of the steps involved :

1. Identify Claims: Extract all the factual claims or statements from the ground truth answer.
2. Attribute Claims: For each claim in the ground truth, determine if it can be attributed to any of the retrieved context chunks. This often involves assessing whether the retrieved information provides sufficient evidence to support the claim.
3. Calculate Recall: Divide the number of ground truth claims that can be attributed to the retrieved context by the total number of claims in the ground truth.

Example:
Let's say a user asks, "Why do people forget things?" and the ground truth answer includes these claims:

- Memory decay: Memories fade over time if not accessed regularly.

- Interference: New information can interfere with old memories.
- Retrieval failure: Sometimes, the information is stored in memory, but we can't access it.
- Motivated forgetting: People may actively suppress unpleasant memories.

Now, suppose the RAG system retrieves the following context:

- Chunk 1: "Without reinforcement, memories weaken over time. This is known as decay theory."
- Chunk 2: "Learning new things can sometimes disrupt older memories, making them harder to recall."
- Chunk 3: "Tip-of-the-tongue phenomenon is an example of retrieval failure, where you know the information but can't quite access it."

In this case, the retrieved context supports the first three claims from the ground truth: memory decay (Chunk 1), interference (Chunk 2), and retrieval failure (Chunk 3). However, it doesn't provide any information about motivated forgetting.
Therefore, the Contextual Recall would be 3/4 or 75%.

Contextual Relevancy evaluates the semantic similarity between the retrieved context and the user's query in a Retrieval-Augmented Generation (RAG) system . It assesses whether the retrieved information truly aligns with the user's intent and provides the necessary context for generating a relevant and accurate response.

Calculation Method:
Calculating Contextual Relevancy often involves using advanced techniques like Large Language Models (LLMs) or specialized algorithms to assess the semantic similarity between the query and the retrieved context. Here's a general approach:

1. Embed the Query and Context: Generate vector representations (embeddings) of the user's query and each chunk of the retrieved context using an embedding model.
2. Calculate Similarity: Calculate the similarity between the query embedding and each context chunk embedding using a similarity metric like cosine similarity.
3. Aggregate Similarity Scores: Aggregate the similarity scores across all context chunks to obtain an overall relevancy score. This could involve averaging the scores or using a weighted average based on the importance of each chunk.

Example:
Let's say a user asks, "What are the environmental benefits of electric cars?" The RAG system retrieves the following context chunks:

- Chunk 1: "Electric cars produce zero tailpipe emissions, reducing air pollution."
- Chunk 2: "Electric car batteries can be recycled, minimizing waste."
- Chunk 3: "The manufacturing process of electric cars can be energy-intensive."

An LLM-based evaluation might determine that Chunk 1 and Chunk 2 are highly relevant to the query, while Chunk 3 is less relevant. The system would then calculate the semantic similarity between the query and each chunk, resulting in high similarity scores for Chunk 1 and Chunk 2, and a lower score for Chunk 3. By aggregating these scores, the system would produce an overall Contextual Relevancy score, reflecting the degree to which the retrieved context aligns with the user's query.

Mean Reciprocal Rank (MRR) is a metric used to evaluate the performance of information retrieval systems, including those used in Retrieval-Augmented Generation (RAG) . It measures the average position of the first relevant item in a ranked list of results . In the context of RAG, MRR is particularly useful for assessing how quickly the system can retrieve the correct answer to a user's query .
Calculation Method:

To calculate MRR, you follow these steps :

1. For each query, obtain a ranked list of results. This list could be generated by a search engine, a recommendation system, or any other information retrieval component.
2. Identify the First Relevant Item: For each query, determine the position of the first relevant item in the ranked list of results. Like other ranking metrics, MRR needs the ground truth. You must identify which items within the top-K recommendations were relevant to the user.
3. Calculate the Reciprocal Rank: For each query, calculate the reciprocal of the rank of the first relevant item. For example, if the first relevant item is at position 3, the reciprocal rank is 1/3. If no relevant items are found, the reciprocal rank is 0.
4. Average the Reciprocal Ranks: Calculate the average of the reciprocal ranks across all queries. This average is the MRR.

Example:
Imagine you have a RAG system that answers questions by retrieving relevant information from a knowledge base. You test the system with three different questions:

- Question 1: "What is the capital of France?"
- Question 2: "Who painted the Mona Lisa?"
- Question 3: "What is the highest mountain in the world?"

For each question, the system retrieves a ranked list of possible answers. Let's say the positions of the first correct answers are:

- Question 1: Correct answer at position 1
- Question 2: Correct answer at position 3
- Question 3: Correct answer at position 3

To calculate the MRR:

1. Reciprocal Ranks:
    - Question 1: 1/1 = 1
    - Question 2: 1/3 = 0.333
    - Question 3: 1/3 = 0.333
2. Average: (1 + 0.333 + 0.333) / 3 = 0.555

The MRR for this RAG system is 0.555. This means that, on average, the first relevant answer appears a little after the first position in the ranked list of results.

Generation Evaluation Metrics
Answer Relevancy (LLM Based) how well the generated response aligns with the user's query in a Retrieval-Augmented Generation (RAG) system . It assesses whether the generated answer directly addresses the user's question and provides the information they are seeking . This metric focuses on the pertinence of the answer without considering its factual accuracy .
Calculation Method:
One common method for calculating Answer Relevancy involves using an LLM to assess the semantic similarity between the query and the generated response . Here's a breakdown of the steps:

1. Generate Embeddings: Generate vector representations (embeddings) of the user's query and the generated response using an embedding model.
2. Calculate Similarity: Calculate the cosine similarity between the query embedding and the response embedding. This similarity score represents the Answer Relevancy.

Example:
Let's say a user asks, "What are the benefits of meditation?" and the RAG system generates the following response:
"Meditation can reduce stress, improve focus, and promote emotional well-being."
To calculate Answer Relevancy, the system would first generate embeddings for both the query and the response. Then, it would calculate the cosine similarity between these embeddings. A

high similarity score would indicate that the response is highly relevant to the query, while a low score would suggest the response might not be directly addressing the user's question.

Faithfulness measures whether the information in the generated response factually aligns with the contents of the retrieved context document chunks (nodes) . It assesses the degree to which the generated output is grounded in the provided context and avoids generating fabricated or misleading content .
Calculation Method:
Calculating Faithfulness typically involves using an LLM to analyze both the generated response and the retrieved context . Here's a common approach:

1. Extract Claims: Identify the factual claims or statements made in the generated response.
2. Verify Claims: For each claim, determine if it is supported by the information present in the retrieved context. This often involves assessing whether the context provides sufficient evidence or explicitly states the claim.
3. Calculate Faithfulness: Divide the number of claims supported by the context by the total number of claims in the generated response.

Example:
Let's say a user asks, "What are the main causes of climate change?" and the RAG system retrieves context mentioning greenhouse gas emissions from human activities. If the generated response states, "Climate change is primarily caused by increased greenhouse gas emissions from human activities like burning fossil fuels and deforestation," then the response is considered faithful to the retrieved context.
However, if the response includes a claim like, "Volcanic eruptions are the primary contributor to climate change," and this claim is not supported by the retrieved context, it would decrease the Faithfulness score.

BLEU (Bilingual Evaluation Understudy) is a metric originally designed to evaluate the quality of machine translations by comparing the generated text to one or more reference translations . It calculates the modified precision score by examining the overlap of n-grams (sequences of words) between the generated text and the reference text . While primarily used for translations, BLEU can also be applied to assess the quality of machine-generated text in other contexts .
Calculation Method:
BLEU is calculated by measuring the overlap of n-grams between the generated response and a set of reference responses . It incorporates a brevity penalty to penalize short responses and weights for different n-gram levels . The formula for the BLEU score is:
BLEU = BP * exp(sum(wn * log(Pn)))
Where:

- BP is the brevity penalty to penalize short responses.
- Pn is the precision of n-grams.
- wn are the weights for each n-gram level.

Example:
Let's say a user asks, "What is the capital of France?" and the RAG system generates the response: "The capital of France is Paris."
Assuming the reference answer is "Paris is the capital of France," the BLEU score would be calculated by comparing the n-grams in the generated response to those in the reference answer. Since the generated response contains all the necessary unigrams (single words) and the correct bigram ("capital of"), it would likely receive a high BLEU score, indicating a good match with the reference answer.

Limitations:
While BLEU is widely used and easy to calculate, it has some limitations :

- It does not consider semantic similarity or word order .
- It may not capture the meaning or nuances of the text effectively .
- It can be sensitive to the choice of reference translations .

Therefore, it's important to use BLEU in conjunction with other metrics to get a more comprehensive evaluation of the RAG system's performance.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to assess how well a machine-generated summary of a text compares to a human-written one. It works by measuring the overlap of words, phrases, and word sequences between the two texts.

Here's a breakdown of the different ROUGE metrics:

- ROUGE-N: Looks at the overlap of N-grams (sequences of N words). For example, ROUGE-1 compares single words, ROUGE-2 compares pairs of words, and so on.
- ROUGE-L: Focuses on the longest common subsequence (LCS) of words between the two texts, regardless of whether the words are consecutive. This helps capture the overall similarity in the ordering of information.
- ROUGE-S: Measures the overlap of skip-bigrams (pairs of words that may not be next to each other). This is useful for capturing meaning even when the word order is different.
- ROUGE-SU: Similar to ROUGE-S, but also includes single words in the evaluation.
- ROUGE-W: Like ROUGE-L, but gives more weight to consecutive matches in the LCS, emphasizing the importance of word order.

For each of these metrics, ROUGE calculates precision, recall, and F1-score.

- Recall: Measures how much of the important information from the human-written summary is captured in the machine-generated summary.
- Precision: Measures how much of the machine-generated summary is actually relevant to the human-written summary.
- F1-score: A balanced measure that combines precision and recall.

Here's a simple **example:**
Imagine a human wrote this summary of a football game: "The Eagles beat the Giants 34-17. Jalen Hurts threw for 250 yards and 2 touchdowns."
And the machine generated this: "The Eagles won against the Giants. Hurts had a great game with 2 touchdowns."
ROUGE-1 (which looks at single word overlap) would identify these matching words: "the", "Eagles", "Giants", "Hurts", "touchdowns". It would then calculate recall (how many of the words from the human summary are in the machine summary) and precision (how many of the words in the machine summary are in the human summary).
ROUGE is a valuable tool for evaluating summaries because it provides a quantitative measure of how well a machine-generated summary captures the key information from a human-written one.
Calculation Method:
To calculate ROUGE, we need to find the overlap between the generated text and the reference text in terms of n-grams, longest common subsequences (LCS), skip-bigrams, and unigrams. Then, we can calculate precision, recall, and F1-score for each metric.
Here's the calculation for ROUGE-1 (unigram overlap):

1. Identify matching unigrams: "the", "Eagles", "Giants", "Hurts", "touchdowns"
2. Calculate ROUGE-1 recall: 5 (matching unigrams) / 10 (total unique unigrams in reference) = 0.5

3. Calculate ROUGE-1 precision: 5 (matching unigrams) / 14 (total unique unigrams in generated summary) = 0.357

The ROUGE-1 score would then be calculated using these precision and recall values, providing a measure of how well the generated summary captures the unigrams from the reference summary.

METEOR (Metric for Evaluation of Translation with Explicit Ordering) is an automatic metric used to evaluate the quality of machine-generated text, such as machine translation or text summarization, by comparing it to a human-written reference text.

What sets METEOR apart from other metrics like BLEU or ROUGE is its ability to consider synonyms and paraphrases when assessing the similarity between the generated text and the reference. This leads to a more nuanced evaluation of semantic similarity, which is crucial for capturing the meaning and fluency of the text, rather than just focusing on surface-level word matches.

Calculation Method:

1. Alignment: METEOR first aligns the generated text with the reference text to identify matching words and phrases.
2. Scoring: It then calculates a score based on the alignment, taking into account:
    - Precision: How many of the words in the generated text are also present in the reference text.
    - Recall: How many of the words in the reference text are also present in the generated text.
    - Word order: How well the order of words in the generated text matches the order in the reference text.
    - Synonymy: It uses an external resource, such as WordNet, to identify synonyms and map them to a common stem, allowing for a more flexible matching process.
3. Penalty: METEOR also includes a penalty for incorrect word order, encouraging the generated text to follow a similar structure to the reference.

Example:

Let's use a very simplified example to get the idea:

- Reference: The cat sat on the mat.
- Generated: The feline sat upon a rug.

1. Matching: "sat" matches exactly.
2. Stemming: No stemming needed in this simple case.
3. Synonym matching: "cat" and "feline" are synonyms, as are "mat" and "rug".
4. Alignment: We have 4 aligned pairs: (the, the), (cat, feline), (sat, sat), (mat, rug)

Now, a simplified METEOR-like calculation (without the exact penalty formula):

- Precision: 4 (matched words) / 4 (words in generated) = 1.0
- Recall: 4 (matched words) / 4 (words in reference) = 1.0
- Penalty: There's a small penalty because "upon a" is a different order than "on the". Let's say this penalty is 0.1.
- Simplified METEOR: (1.0 + 1.0) / 2 * (1 - 0.1) = 0.9

BERT score is a relatively new method for evaluating the quality of machine-generated text. It leverages the power of BERT (Bidirectional Encoder Representations from Transformers), a powerful language model developed by Google, to assess the similarity between generated text and a reference text.

Calculation Method:

1. Contextual Embeddings: Both the generated text and the reference text are fed into a pre-trained BERT model. BERT processes these texts and generates contextual embeddings for each word, capturing its meaning within the sentence.
2. Cosine Similarity: The cosine similarity between the embeddings of corresponding words in the generated and reference texts is calculated. This measures how semantically similar the words are in their respective contexts.
3. Aggregation: The individual word similarities are aggregated to produce an overall similarity score for the entire text. This can be done using different methods, such as averaging or taking the maximum similarity.

Example:

4. Let's say the reference text is: "The cat sat on the mat."
5. And the generated text is: "The feline was sitting on the rug."
6. BERT Score would recognize that "cat" and "feline" are semantically similar, as are "mat" and "rug," and would give a higher score than metrics that only consider exact word matches.

Fairness Metrics

Toxicity Measures the extent to which the generated response contains toxic or offensive language. A lower toxicity score is desirable, indicating that the response is safe and respectful.

In the context of Retrieval Augmented Generation (RAG), toxicity refers to the presence of harmful, offensive, or disrespectful language in the generated responses. This can include hate speech, personal attacks, profanity, and other forms of language that can create a negative or hostile environment for users.

Ensuring that RAG systems generate non-toxic responses is crucial for several reasons:

- User Safety and Well-being: Toxic language can cause emotional distress, discourage participation, and even lead to real-world harm.
- Trust and Engagement: Users are more likely to trust and engage with systems that provide safe and respectful interactions.
- Brand Reputation: Toxic outputs can damage the reputation of the organization or platform deploying the RAG system.
- Ethical Considerations: It's important to develop AI systems that align with ethical principles and avoid perpetuating harmful biases or behaviors.

Calculation Method using RoBERTa:
Utilize a pre-trained RoBERTa model specifically fine-tuned for toxicity detection. The model will typically output a probability or score indicating the likelihood that the input text is toxic. You'll need to decide on a threshold above which you consider text to be toxic (e.g., 0.5 or 0.7)

Choosing the Right Metric for a Specific RAG Task

Selecting the appropriate metric for evaluating a RAG system depends on the specific task and the desired outcome. For instance, if the goal is to generate a concise summary of a lengthy document, ROUGE, which measures the recall of n-grams, word sequences, and word pairs in the generated text compared to a reference text , might be a suitable metric due to its focus on

recall . Conversely, if the task involves answering a question with a single correct answer, MRR, which evaluates how quickly the system retrieves the correct answer , might be a better choice as it emphasizes the retrieval of the correct answer quickly .
Factors to consider when choosing a metric include:

- The type of RAG task: Different RAG tasks have different evaluation needs. Here's a breakdown of some common tasks and their corresponding metrics:
    - Summarization: ROUGE , SummarizationScore, compression ratio, coverage
    - Question Answering: MRR , MAP, accuracy, faithfulness, answer relevancy, context precision , context recall
    - Dialogue Generation: accuracy, fluency, coherence, relevance, bias, toxicity
    - Code Generation: code quality metrics (e.g., code complexity, code coverage), execution accuracy, code readability
- The desired outcome: What aspects of the RAG system are most important?
    - Accuracy: Focus on metrics like faithfulness , answer correctness, and context precision .
    - Fluency: Consider metrics like BLEU and METEOR , which assess the grammatical correctness and readability of the generated text.

By carefully considering these factors, developers can select the most appropriate metric to evaluate their RAG systems effectively.

Conclusion

In this paper, we have presented a comprehensive evaluation framework for Retrieval-Augmented Generation (RAG) systems, designed to address the unique challenges posed by these complex systems. Our framework incorporates a systematic approach to generating synthetic query datasets, evaluating both the retrieval and generation components, and assessing the overall performance of the system. We advocate for a multi-faceted evaluation strategy that encompasses various metrics, including retrieval metrics, generation metrics, and fairness metrics, to provide a holistic view of the system's capabilities and limitations.
Our work is particularly important from a risk management perspective, as it provides a rigorous methodology for identifying and mitigating potential risks associated with RAG systems. By thoroughly evaluating these systems, we can ensure that they generate accurate, relevant, and safe responses, while also addressing concerns related to bias, fairness, and explainability.