



Enhanced RAG Pipeline with SPLADE Retrieval and Chain-of-Thought Reasoning

AUTHORS

Titouan Guerin & Yacine Chettab

February 2, 2026

1 Introduction

Large Language Models (LLMs) are currently the state of the art for text generation and are widely used for question answering tasks, with publicly available models such as GPT by OpenAI being used daily by hundreds of millions of people. Despite their powerful generative capabilities, LLMs have a noticeable limitation: their knowledge is restricted to the data they were trained on. As a result, they may not be able to generate quality information in specialized or niche domains, which can negatively impact the quality and reliability of their answers.

To address this limitation, Retrieval-Augmented Generation (RAG) has been widely adopted as an effective solution. RAG enhances LLMs by allowing them to retrieve relevant information from an external document collection at inference time. By grounding generation in available information, RAG systems can produce more accurate, informative, and context-aware answers compared to standalone (“vanilla”) language models.

Recent advances in information retrieval, such as sparse neural retrievers like SPLADE, have further improved the effectiveness of RAG pipelines by enabling better matching between queries and relevant documents. On top of that, new prompting and reasoning strategies such as Chain-of-Thought (CoT) reasoning, have shown that allowing a model to generate text through intermediate reasoning steps can significantly improve their performance on complex question answering tasks.

This project explores Retrieval Augmented Generation, SPLADE, and explicit reasoning mechanisms to study how they can improve the performance of LLMs on question answering tasks. The goal is to analyze how integrating retrieval and reasoning leads to more robust and reliable answers compared to traditional LLM-based approaches.

2 Structure and Content of the Work

In this project, we designed and implemented a functional interactive question answering system based on RAG combined with Chain-of-Thought reasoning. The primary objective of this work is not only to improve the quality of generated answers, but also to analyze how the model arrives at its decisions by explicitly exposing intermediate reasoning and evaluation steps throughout the pipeline.

To this end, we developed a complete conversational system that allows a user to pose natural language questions and receive answers grounded in an external document collection. The system follows a multi-stage process involving query reformulation, document retrieval, reasoning over retrieved evidence, answer generation, and answer verification. This design enables us to observe and evaluate the impact of retrieval and reasoning mechanisms on the final output produced by the language model. Thus the topics we studied are:

- Using an LLM to rewrite user queries before retrieval (CoT-style decomposition)
- Implement iterative RAG (checking with an LLM the quality of an answer)

All of our work is open sourced and available [here](#).

2.1 Dataset

For our experiments, we relied on the dataset `irds:lotte/technology/dev/search`, which is part of the LOTTE benchmark collection for information retrieval research. This dataset consists of a large corpus of technology-related documents covering topics such as computer hardware, software systems, networking, and general computing concepts. We also have access to a set of search queries designed to reflect realistic user information needs in the technology domain, as well as relevance judgments (qrels) that associate queries with documents deemed relevant, enabling principled evaluation of retrieval performance.

The dataset is particularly well-suited for Retrieval-Augmented Generation, as it provides a rich and specialized knowledge base that may not be fully covered by the pretraining data of a language model. We suppose that giving access to this database to the generative model will increase the quality of its outputs.

In order to properly illustrate the effectiveness of RAG, we deliberately focused on questions that are closely related to the dataset’s content. For example, queries such as “*What is RAM in computers?*” or “*How does computer memory differ from storage?*” directly align with the technological topics present in the corpus. This choice ensures that relevant documents exist in the collection and allows the benefits of retrieval-based grounding to be clearly observed. We will not be asking the model Geography questions for example.

2.2 Overview of the RAG Architecture

Our RAG system integrates three key components:

1. A modern sparse retrieval model based on SPLADE to identify relevant documents.
2. Chain-of-Thought reasoning to explicitly analyze retrieved evidence and guide answer generation.
3. An iterative RAG strategy that evaluates answer quality and generates alternative responses when necessary.

Together, these components form a robust pipeline that goes beyond a single-pass retrieval and generation approach. In the following sections, we describe each of these components in detail, starting with the SPLADE-based retrieval mechanism.

2.3 SPLADE-based Retrieval

To power the retrieval component of our RAG pipeline, we rely on *SPLADE*, a sparse neural retrieval model that combines the interpretability of traditional lexical methods with the expressive power of neural representations. SPLADE learns to expand queries and documents into high-dimensional sparse vectors, enabling more effective term matching than purely frequency-based approaches while remaining compatible with inverted index retrieval.

As a baseline, we use *BM25*, a classical information retrieval method that relies on term frequency and inverse document frequency statistics. BM25 is not a learned or parameterized model and therefore serves as a strong and widely adopted benchmark for evaluating the effectiveness of neural retrieval approaches. We compared three retrieval models:

- BM25 as a non-learned baseline,
- a SPLADE model trained during the practical sessions using synthetic data from the LOTTE dataset as well as training triplets,
- a pre-trained SPLADE model loaded from Hugging Face (`naver/splade-cocondenser-ensembledistil`).

The trained SPLADE model consistently outperforms BM25 across all evaluated metrics, as shown in Table 1. However, the improvements remain relatively modest, with small gains in recall and reciprocal rank. We hypothesize that this limited improvement may be due to the quality and size of the training data, as well as potential overfitting effects when training on synthetic or limited triplet data.

In contrast, the pre-trained SPLADE model loaded from Hugging Face significantly outperforms both BM25 and the locally trained SPLADE model, as shown in Tables 2 and 3. Notably, it achieves substantial gains across all metrics, with improvements of up to +0.0795 in R@1, +0.1447 in R@5, and +0.1480 in reciprocal rank compared to BM25. These results highlight the effectiveness of large-scale pretraining and distillation strategies employed in state-of-the-art retrieval models.

Based on these observations, we chose to use the `naver/splade-cocondenser-ensembledistil` model as the retriever in our RAG pipeline. This choice ensures stronger and more reliable document retrieval, which is critical for grounding the generation and reasoning steps of the system.

Finally, it is worth noting that our project repository includes multiple scripts dedicated to SPLADE experimentation, including training, evaluation, and comparison with traditional baselines. These scripts were initially developed during the practical sessions and subsequently reused and extended for this project.

Table 1: Comparison of Retrieval Performance Across BM25 and SPLADE Variants

Metric	BM25	SPLADE Trained	SPLADE Loaded
R@1	0.0075	0.0158	0.0870
R@5	0.0333	0.0483	0.1780
RR	0.0434	0.0568	0.1914
R@10	0.0625	0.0650	0.1867

2.4 Chain-of-Thought Reasoning

Chain-of-Thought (CoT) reasoning is used in this work to encourage the language model to explicitly reason about intermediate steps rather than directly producing an answer. This approach improves transparency and reliability by making the decision process more interpretable and reducing the likelihood of shallow or unsupported responses.

In our system, Chain-of-Thought reasoning is integrated at multiple stages of the RAG pipeline. First, it is used during query rewriting, where the model decomposes the original question into its key concepts and generates alternative formulations that may improve document retrieval. For each query, the model fetches the top 3 most relevant documents (using SPLADE). For example, here is the output of the model rewriting a user query:

Original: how does the blockchain work?
 Rewrite 1: 1. What is the process of blockchain technology?
 Rewrite 2: 2. How does the blockchain system function?

Second, CoT is applied to analyze the retrieved documents, allowing the model to reason about their relevance, identify useful information, and highlight potential gaps in the retrieved context. Finally, CoT is employed during answer evaluation, where the model critically assesses the generated response in terms of correctness, completeness, relevance, and clarity. The output of the analysis of the retrieved documents by the model shortened to avoid taking too much space, is available in the appendix section 5.1.

Overall, the main logic of Chain-of-Thought reasoning in our implementation is to separate reasoning from decision-making. The model is prompted to articulate its analysis at each stage—query reformulation, retrieval inspection, and answer evaluation—while the final selection of the answer is based on explicit quality scores rather than subjective preference. The LLM has to grade out of 5 each suggested answer. An answer that is not of good quality triggers the LLM to try and improve its answer, which we will discuss in the following section. Having and observing this behavior makes the model more certain about the answer that it gives, and also ensures the user will be more satisfied with the answer it receives.

2.5 Iterative Retrieval-Augmented Generation

While the previous section focused on how Chain-of-Thought reasoning enables the model to analyze and interpret the retrieved documents, we further extend the capabilities of the LLM by allowing it to explicitly judge its own answers. This approach, referred to as Iterative RAG, introduces a self-evaluation loop in which the model assesses the quality of its generated response and decides whether further refinement is necessary.

After producing an initial answer, the model is prompted to score its own output according to predefined qualitative criteria such as correctness, completeness, relevance, and clarity. If the resulting quality score is deemed unsatisfactory, the model is instructed to return to the reasoning phase, re-examine the retrieved documents, and generate an improved alternative answer. This process is repeated until the model considers its response to be of sufficient quality or until a predefined stopping condition is met.

In practice, due to the relatively simple and well-structured nature of the documents in the dataset, we observe that the model generally requires at most one rejection and re-evaluation cycle before producing an acceptable answer. This

suggests that iterative self-critique, when combined with retrieval grounding and explicit reasoning, can significantly improve answer quality with limited additional computational cost.

To illustrate this process, the following example shows the internal evaluation, answer rejection, refinement, and final selection performed by the model during a single iterative RAG cycle. Each output is shortened, since they are only here to illustrate the process the model goes through to generate an answer, and is found in the appendix section 5.2.

3 Observations and Analysis

Overall, we observed that providing the language model with access to an external database of relevant documents significantly improves the overall quality of generated answers.

We ran a short evaluation of our pipeline on 50 test queries from our dataset across four dimensions: answer quality (scored 1-5 according to the LLM), retrieval relevance (whether retrieved documents matched ground-truth relevant sets), answer consistency (quality differences between initial and alternative answers), and pipeline efficiency (CoT reasoning steps performed). The results are available in the file *src/evaluation/evaluation_results.json*. All retrieved documents were relevant to the queries, and final answers averaged 3.68/5 with a median of 4.0 and low variance, indicating consistent performance. Notably, alternatives were generated for 98% of queries and showed meaningful quality improvements (average difference of 0.82 points), suggesting the comparison step effectively catches and improves weaker answers. The pipeline reliably performed all reasoning steps without sacrificing quality, showing that query rewriting, retrieval analysis, and iterative comparison provide real value despite the extra computation.

However, we also observed that the benefits of RAG are not uniform across all queries and situations. One limitation lies in the inherent subjectivity of answer evaluation. Depending on the phrasing and complexity of a question, the model may apply stricter or looser quality criteria when judging its own responses. This can lead to inconsistencies in scoring and answer refinement, even when the retrieved context is similar.

We also noticed that our pipeline almost always triggers the CoT improvement of the answer. This is somewhat surprising since some queries that we tested were rather straightforward. We suppose that the prompts given to the model implicitly bias it towards reevaluating its answers. According to our observations, many originally generated answers would have been enough for most users.

Another observed limitation is that the model sometimes relies on its built-in knowledge rather than fully utilizing the retrieved documents. For simple questions (practically yes or no at times), the model may already possess a correct answer and therefore produce a response without consulting the retrieved context. In such cases, the RAG mechanism does not necessarily improve the answer and may even introduce irrelevant or noisy information if the retrieved documents are imperfectly aligned with the query.

Overall, these observations reflect well-known limitations of RAG systems. While RAG substantially enhances performance in many cases, it does not fully eliminate challenges related to retrieval noise, subjective evaluation, and the interaction between parametric and retrieved knowledge.

4 Conclusion

In this project, we developed a functional and interactive RAG question-answering pipeline enhanced with CoT reasoning and iterative self-evaluation. Overall, we are satisfied with the performance and robustness of our system, which is essentially user-ready: it allows direct interaction with the model, supporting a full user to model loop for real time question answering.

While the current evaluation focused on one dataset, future work could explore other document collections to observe how the pipeline adapts to different types of knowledge and retrieval challenges. Additionally, controlled experiments comparing RAG-enhanced models against standard LLMs without retrieval could provide further insight into the benefits and limitations of our approach.

5 Appendix

5.1 CoT Reasoning output

```
Variant: 'how does the blockchain work?'
1. Relevance: These documents are relevant to the query "how does the blockchain work?" as they
→ provide explanations and details about the concept, features, and processes involved in blockchain
→ technology.

- Doc 1 discusses the block selection mode in a Windows environment.
- Doc 2 explains [...]
```

2. Useful information:

```
- Doc 1 provides step-by-step instructions [...]
```

3. Gaps in context:

```
- The retrieved documents
```

Variant: '1. What is the process of blockchain technology?'

```
1. How relevant are these documents to the query? [...]
```

5.2 Iterative RAG output

```
=====
STEP 5: Quality Evaluation (with reasoning)
=====
Quality Score: 2/5
Needs Improvement: True

Reasoning:
Quality score: 2

Reasoning: The answer is correct in that it explains the basic functioning of the blockchain,
→ including the process of verifying transactions [...]
Suggestions for improvement:

- Provide more detail about the role of miners in [...]
```

```
=====
STEP 6: Generating Alternative Answer
=====
The Blockchain is a decentralized, distributed, and secure digital ledger technology that enables the
→ recording and verification of transactions across a network of computers without the need for a
→ central authority or intermediary. Here's how it works:

1. Each node on the network [...]
```

```
=====
STEP 7: Evaluating Alternative Answer
=====
Quality Score: 3/5

Reasoning:
```

Quality Score: 3/5

Reasoning: The answer provides a brief overview of how the blockchain works [...]

Quality Score: 3/5

Reasoning: The answer is partially correct but lacks completeness and relevance to the question [...]

=====

STEP 8: Comparing Answers and Selecting Best

=====

Comparison Analysis:

Answer 1 (Quality Score: 2/5):

Answer 1 provides a brief overview of how the blockchain works, but it lacks in-depth analysis [...]

Answer 2 (Quality Score: 3/5):

Answer 2 offers a more comprehensive explanation of how the blockchain works, covering key aspects

→ such as the decentralized nature, [...]

Comparing the two answers, Answer 2 (Quality Score: 3/5) has a higher quality score due to its more

→ comprehensive and detailed explanation of the blockchain's inner workings. It covers more

→ essential points and provides clearer examples, [...]

--- Selection Based on Quality Scores ---

Answer 1 Quality: 2/5

Answer 2 Quality: 3/5

Selected: Answer 2 (Score: 3/5)

=====

FINAL ANSWER:

=====

The Blockchain is a decentralized, distributed, and secure digital ledger technology that enables the

→ recording and verification of transactions across a network of computers without the need for a

→ central authority or intermediary. Here's how it works:

1. Each node on the network is responsible for maintaining a copy of the Blockchain, which is

→ essentially a public, tamper-proof ledger. [...]

Quality Score: 3/5