
ITA Assignment 04

RAG-based Question-Answering

Ibrahim Ayoubi | Mahnoor Adeel(best) | Kisa Fatima

TABLE OF CONTENTS

I. Platform Details.....	3
II. Data Details.....	3
III. Algorithms, Models, and Retrieval Methods.....	4
Retrieval Methods.....	5
Models and Embedding Approaches.....	5
Chunking Strategy.....	5
Additional Techniques and Observations.....	6
Final Insights.....	6
IV. Performance Metrics.....	7
Relevance Score Strategy.....	7
Challenges with Computing Faithfulness.....	8
V. Best RAG Pipeline.....	9
Model Configuration.....	11
Evaluation Metrics.....	11

I. Platform Details

The experimentation was conducted on Kaggle completely.

II. Data Details

The dataset was created by scraping [Overleaf's Learn documentation](#) to create a structured dataset of LaTeX guides using Python (BeautifulSoup & Requests). The goal was to extract guide content (titles, text, lists, and code blocks) and store it in a structured JSON file.

1. Fetching Guide Links

The script first accessed the /learn page and collected all internal links (/learn/...) using **BeautifulSoup**. These links were stored in a list for further processing.

2. Scraping Each Guide

For each guide URL, the script performed detailed parsing of the HTML structure:

- **Titles** were captured from **<h1>** tags
- **Section headings** were extracted from both **<h2>** and **<h3>** elements
- **Paragraph content** came from **<p>** tags
- **Lists** (both bulleted and numbered) were taken from **** and **** elements
- **Code examples** were preserved from **<pre>** blocks

The system automatically filtered out empty elements to maintain data integrity.

3. Storing Data

All extracted content was organized into a **hierarchical JSON structure** for each guide, maintaining clear relationships between different content types. The final dataset of **151** complete guides was saved as **overleaf_guides.json**

4. Stored Data Structure

Each guide in the JSON file includes:

- **"title"** (from <h1>).
- **"url"** (original guide link).
- **"content"** (a list of sections with "type" and "data" fields for headings, text, lists, and code).

This approach efficiently captured LaTeX documentation in a machine-readable format for further analysis or archival.

III. Algorithms, Models, and Retrieval Methods

In our assignment, we evaluated various retrieval methods, chunking strategies, embedding models, and LLMs to determine how different configurations impact the performance of a Retrieval-Augmented Generation (RAG) system. The objective was to find the most effective pipeline that enhances answer relevance, context awareness, and completeness.

Retrieval Methods

We used three retrieval methods: **FAISS**, **TF-IDF**, and **BM25**. FAISS used vector embeddings to find semantically relevant chunks and gave clear answers when paired with good models. TF-IDF and BM25 relied on keywords. TF-IDF gave stable results with overlapping chunks, while BM25 performed better with query rephrasing and quality embeddings. Overall, keyword methods improved when combined with modern embeddings.

Models and Embedding Approaches

We used sentence-transformer models like **all-MiniLM-L6-v2** and **nomic-embed-text-v1.5** to generate document embeddings. These helped us match user questions with the most relevant parts of the text. For generating answers, we used multiple LLMs such as **Mistral-7B**, **Gemma-3B**, **Zephyr-7B**, and **Qwen-3B**. Larger models like Mistral-7B and Zephyr-7B gave better and more complete answers, especially when used with good retrieval and chunking. Smaller models like Gemma-3B struggled to generate detailed answers even with relevant context.

Chunking Strategy

We experimented with different chunk sizes and overlap methods. For example, we used 512 tokens with 50-token overlap and also tried 200-word chunks with a 30-word overlap. Overlapping chunks helped maintain the flow of information across segments. This improved the quality of retrieved content and reduced the chances of missing important context.

Additional Techniques and Observations

- We tested different values of **k** (top retrieved chunks), ranging from **3 to 8**.
- **FAISS (semantic search)** with **k = 3 to 7** gave high relevance scores, especially with models like **Zephyr-7B** and **Qwen-3B**.
- **BM25** worked better when **queries were rephrased** and **embeddings were accurate**.
- **Without RAG**, model answers were often **vague or incomplete**.
- **With RAG**, especially using **semantic search and overlapping chunks**, answers became **more detailed** and **closely matched the question**.

Final Insights

In conclusion, combining **semantic search, overlapping chunking, query rephrasing**, and a **powerful LLM** gave the best results. Models like Mistral-7B and Zephyr-7B performed well when supported by good embeddings and chunking. BM25 also worked well when configured correctly. Smaller models like Gemma-3B were less effective, even with good context.

IV. Performance Metrics

Relevance Score Strategy

To evaluate the relevance of an answer in response to a specific query, we implemented a **semantic similarity-based strategy** powered by **large language models (LLMs)**. Rather than relying on keyword overlap or surface-level heuristics, this method captures the **underlying meaning** of both the query and the response.

The idea is to **reverse-engineer the query** from the answer. If a model can generate meaningful questions based on the answer that are **semantically similar** to the original query, it indicates that the answer is contextually relevant.

Following function was used to compute relevance score:

```
calculate_answer_relevancy_from_query_and_response(query, response)
```

Step-by-step Breakdown:

1. Generating Artificial Questions from the Response

Using a LLM to **generate artificial questions** based on the response generated from the **RAG pipeline**. This step simulates **reverse-engineering** what kinds of questions the given answer might belong to.

2. Computing Semantic Embeddings

The original query and each of the generated questions are converted into **vector representations (embeddings)**. **Mean of the model's output logits** as a proxy for semantic representation. These logits are the raw output scores before generating the final text, and by **averaging** them across all tokens, a single, dense vector is obtained that reflects the overall **semantic intent** of the input.

3. Computing Relevancy Score

Cosine similarity is used to measure the **semantic closeness** between the query embedding and each generated question embedding. This metric assesses the angle between high-dimensional vectors, serving as a proxy for

semantic similarity. The final relevancy score is computed as the **average** of these similarity values. A higher score indicates that the query closely aligns with the types of questions the response naturally addresses, reflecting a more contextually appropriate answer.

While questions can also be generated using external LLMs like ChatGPT or DeepSeek(faster and easier approach), the same model that generated the response was employed to generate artificial questions to ensure fairness and consistency.

Challenges with Computing Faithfulness

Computing faithfulness remains a complex task, as it requires verifying whether a response is not only relevant but also factually grounded in the source context or knowledge base. Unlike relevance, which can be approximated using mathematical formulas like cosine similarity, faithfulness cannot be measured directly. It requires identifying and extracting factual claims from the response and checking them against the source or ground truth—a process that involves natural language understanding, information retrieval, and even reasoning. Automated methods frequently struggle to detect subtle hallucinations or factual shifts, while human evaluation, though more accurate, is not easily scalable.

To complement this, we wrote our own evaluation understanding for each run in the attached Excel sheet.

V. Best RAG Pipeline

After experimenting with multiple configurations of our **Retrieval-Augmented Generation (RAG)** pipeline, the best-performing setup is summarized below:

Query

“How to add an image in latex?”

Retrieved Context

1

Title: Fixing and preventing compile timeouts

URL: https://www.overleaf.com/learn/how-to/Why_do_I_keep_getting_the_compile_timeout_error_message%3F

Text:

2 but not in version 3. The use of EPS or SVG images can require extra processing to convert them to PDF format. This extra processing will add to the time needed to compile your project. While the latex compiler will support EPS images directly, the pdfLaTeX compiler does not support EPS images, so an extra step is required to convert these to pdf images when that compiler is used. The processing is handled by the `stopdfpackage` which uses Ghostscript to convert the EPS files to PDF. This conversion ...

2

Title: Fixing and preventing compile timeouts

URL: https://www.overleaf.com/learn/how-to/Debugging_Compilation_timeout_errors

Text:

“2 but not in version 3. The use of EPS or SVG images can require extra processing to convert them to PDF format. This extra processing will add to the time needed to compile your project. While the latex compiler will support EPS images directly, the pdfLaTeX compiler does not support EPS images, so an extra step is required to convert these to pdf images when that compiler is used. The processing is handled by the `stopdfpackage` which uses Ghostscript to convert the EPS files to PDF. This conversion ...”

3

Title: Inserting Images

URL: https://www.overleaf.com/learn/latex/Inserting_Images

Text:

“more complete article about image positioning see [Positioning images and tables](#) Open an images example in Overleaf Captioning, labelling and referencing Captioning images to add a brief description and labelling them for further reference are two important tools when working on a lengthy text. Captions Let's start with a caption example:
`\begin{figure}[h]\caption{Example of a parametric plot ($\sin(x)$, $\cos(x)$, x)}\centering\includegraphics[width=0.5\textwidth]{spiral}\end{figure}` It's really eas...”

4

Title: Fixing and preventing compile timeouts

URL: https://www.overleaf.com/learn/how-to/Debugging_Compilation_timeout_errors

Text:

“can correct your code. Several common errors are explained on this page, and some LaTeX debugging suggestions are listed on this page. Please also see this section in this article that lists some common fatal compile errors that cause compile timeouts.

Step 2: Try compiling inFast [draft]mode Including many or large image files in your project can significantly add to the time required to compile your project. One method that can help find out if image processing is the cause of the timeout is to try c..."

5

Title: Fixing and preventing compile timeouts

URL: https://www.overleaf.com/learn/how-to/Why_do_I_keep_getting_the_compile_timeout_error_message%3F

Text:

"can correct your code. Several common errors are explained onthis page, and some LaTeX debugging suggestions are listed onthis page. Please also see thesection in this articlethat lists some commonfatal compile errorsthat cause compile timeouts.

Step 2: Try compiling inFast [draft]mode Including many or large image files in your project can significantly add to the time required to compile your project. One method that can help find out if image processing is the cause of the timeout is to try c..."

6

Title: Articles

URL: <https://www.overleaf.com/learn/latex/Articles>

Text:

"Lift the Lid of TeX Boxes How TeX Calculates Glue Settings in an \hbox Boxes and Glue: A Brief, but Visual, Introduction Using LuaTeX TeX tables TeX Tables: How TeX Calculates Spanned Column Widths Mathematical typesetting OpenType-based math typesetting: An introduction to the STIX2 OpenType fonts Miscellaneous topics The Stoic Resilience of PDF Within a Digital Ecosystem How to use latexmkrc with Overleaf: examples and techniques How to create a multilingual, customisable CD disk jewel case ca.."

Generated Response

"To add an image in LaTeX, follow these steps:

1. Save the image in a format that LaTeX can read, such as PDF, PNG, or JPEG.
2. Insert the following command in your LaTeX document, replacing `image.pdf` with the name of your image file:

```
```latex
\includegraphics[width=0.5\textwidth]{image}
```
```

This will insert the image with a width of 50% of the text width. You can adjust the width by changing the value in the square brackets.

3. Compile your LaTeX document to see the image.

Note: If you're using the `pdfLaTeX` compiler, you may need to install the `graphicx` package to be able to use this command. You can do this by adding the following line to the preamble of your document:

```
```latex
\usepackage{graphicx}
```
```

If you're using the `latex` compiler, you may need to convert your image to EPS format and use the `epsfig` package instead of `graphicx`. Here's an example:

```
```latex
\usepackage{epsfig}
\epsfig{file=image, width=0.5\textwidth}
```
```

I hope this helps! Let me know if you have any other questions.

”

Answer:

To add an image in LaTeX, follow these steps:

1. Save the image in a format that LaTeX can read, such as PDF, PNG, or JPEG.
2. Insert the following command in your LaTeX document, replacing `image.pdf` with the name of your image file:

```
```latex
\includegraphics[width=0.5\textwidth]{image}
```
```

This will insert the image with a width of 50% of the text width. You can adjust the width by changing the value in the square brackets.

3. Compile your LaTeX document to see the image.

Note: If you're using the `pdfLaTeX` compiler, you may need to install the `graphicx` package to be able to use this command. You can do this by adding the following line to the preamble of your document:

```
```latex
\usepackage{graphicx}
```
```

If you're using the `latex` compiler, you may need to convert your image to EPS format and use the `epsfig` package instead of `graphicx`. Here's an example:

```
```latex
\usepackage{epsfig}
\epsfig{file=image, width=0.5\textwidth}
```
```

I hope this helps! Let me know if you have any other questions.

Model Configuration

- **Retrieval Model:** BM25
- **Retriever Parameters:**
 - **Top-K Documents Retrieved:** $k = 6$
- **Generator Model:** Zephyr-7B-Beta
- **Prompt:**

Evaluation Metrics

- **Relevance Score:** 0.944071963489725

The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.

Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.

The attention mask is not set and cannot be inferred from input because pad token is same as eos token. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.

Answer Relevancy Score: 0.9440719634897254