|  |
| --- |
| STAR WARS: BROTHERHOOD RAG |

Table of Contents

[Introduction 2](#_Toc195800242)

[Platform Details 2](#_Toc195800243)

[Data Details 2](#_Toc195800244)

[Algorithms, Models, and Retrieval Methods 3](#_Toc195800245)

[Retrieval Methods 3](#_Toc195800246)

[Models 3](#_Toc195800247)

## Introduction

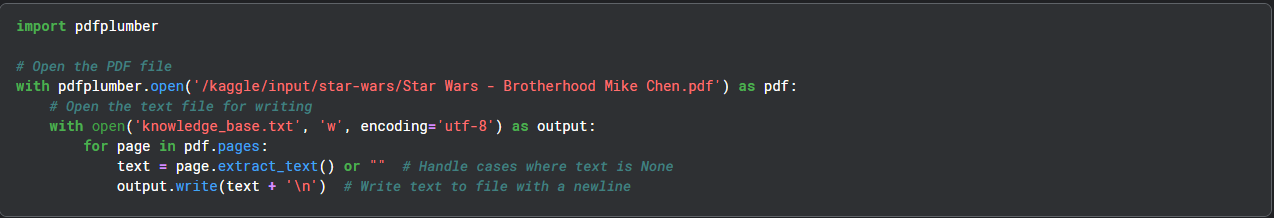
This document outlines the development of a Retrieval-Augmented Generation (RAG) based question-answering system designed to answer natural language queries about the novel *Star Wars: Brotherhood*. The system addresses the challenge of efficiently retrieving and generating relevant information from a specific domain corpus—in this case, a single PDF of the novel. By leveraging advanced retrieval and generation techniques, the RAG system ensures accurate and context-aware responses. The use case for this RAG system lies in providing an interactive tool for readers, researchers, or fans who seek detailed insights or quick answers related to the novel's content, thereby enhancing their engagement and understanding of the material.

## Platform Details

For all stages of the project, including PDF parsing, data cleaning, experimentation, and model execution, the Kaggle platform was utilized. Kaggle's environment provided the necessary computational resources, such as GPUs, which facilitated efficient processing and experimentation. All processes were conducted within Kaggle notebooks, ensuring consistency and ease of resource management. No other platforms were used during the development and testing of the system.

## Data Details

The data used for this project consisted of a single PDF document of the book ***Star Wars: Brotherhood*.** The source of the document was LibGen, a repository containing thousands of PDFs of various books. The PDF is approximately 352 pages long and served as the sole corpus for the system. The pdf was parsed through the pdfPlumber module.



## Algorithms, Models, and Retrieval Methods

### Retrieval Methods

The retrieval phase of the system utilized a hybrid approach, combining the strengths of both semantic search and keyword-based search using dense embeddings retriever and BM-25 techniques to ensure high-quality document retrieval.

1. **BM25 Function:** BM25, a probabilistic retrieval model, was used to rank documents based on the frequency of query terms adjusted by term saturation and document length normalization. It excelled at capturing keyword-based relevance, making it effective for queries requiring exact term matches.
2. **Dense Embeddings Retriever:** Dense retrieval was implemented using pre-trained transformer-based models to generate semantic embeddings (BAAI/bge-small-en, GPT4AllEmbedgins). This approach allowed for understanding the contextual relationships between queries and documents, improving retrieval for natural language queries.
3. **Reciprocal Rank Fusion (RRF):** The results from BM25 and the dense embeddings retriever were combined using RRF. This method prioritized documents ranked highly by both retrieval strategies, selecting the top n documents from the fused results. This hybrid mechanism ensured precision and semantic relevance in retrieval.

**Justification for the Approach**

Combining BM25 with dense embeddings retrieval provided a balance between precision and recall. BM25 addressed keyword-matching scenarios, while dense embeddings captured semantic nuances. RRF integration leveraged the strengths of both methods, resulting in a robust and versatile retrieval system.

Standalone BM25 performed well for exact term queries but struggled with semantic comprehension. Dense retrieval captured context effectively but occasionally prioritized less relevant documents due to limited reliance on keyword matches. The hybrid approach with RRF resolved these limitations, delivering improved overall performance for diverse query types.

### Models

For the generation phase of the system, multiple state-of-the-art large language models (LLMs) were employed to evaluate their effectiveness in answering questions based on the retrieved documents. The specific models used were

* **Falcon-3B-Instruct**
* **Falcon-7B-Instruct**
* **Qwen/Qwen2.5-3B-Instruct**
* **Qwen/Qwen2.5-7B-Instruct**
* **Mistral/Mistral-3B-Instruct**
* **Mistral/Mistral-7B-Instruct**

**Explanation for Choices**

* **Falcon Models:** Chosen for their efficient architecture and strong performance in generating coherent and contextually relevant responses. The instruct fine-tuned versions enhanced alignment with user queries.
* **Qwen Models:** Known for their structured understanding and contextual reasoning. The 2.5 versions provided improved capabilities over earlier iterations.
* **Mistral Models:** Selected for their robust generalization and logical inference abilities. The instruct-tuned variants ensured high-quality outputs across diverse query types.

Using both 3B and 7B variants of Falcon, Qwen, and Mistral models ensured that the system could cater to a broad range of scenarios, from lightweight tasks to those requiring extensive reasoning. This approach facilitated a comprehensive evaluation and optimization of the system for diverse user requirements.

Employing multiple LLMs allowed for comparative analysis across various parameters, such as:

* **Performance:** To identify the best-performing model for different query types.
* **Efficiency:** To evaluate the trade-offs between latency and answer quality.