RAG Pipeline Report - The Castle of Otranto

The main goal of this project was to build a Retrieval-Augmented Generation (RAG) system using one of the books from the Gutenberg electronic library. The RAG system was designed to answer questions about the selected book using a combination of document retrieval and large language model (LLM) generation.

At the start of the project, I focused on preparing the data and building a pipeline where the text is tokenized and passed through an LLM alongside relevant context to generate accurate answers to knowledge-based questions.

For the data preparation phase, I used the Gutenberg metadata API to list and fetch book content. A script was developed to automate this retrieval. I then cross-referenced the list of books with the NarrativeQA dataset to identify Gutenberg books with available QA test sets. These were stored in a CSV file, each with a unique ID. I checked whether each ID had a corresponding test set in NarrativeQA. Among those, I filtered for books available in '-0.txt' format, and ultimately selected 'The Castle of Otranto' as the final candidate.

Once selected, the book underwent a cleaning process: removing start and end markers, chapter headings, excessive whitespace, and unnecessary punctuation. This resulted in a cleaner version of the text.

Following the cleaning step, I extracted the question-answer pairs associated with this book from the NarrativeQA dataset.

I then moved to the chunking stage. I experimented with several methods to create text chunks. I used the BGE-small model and MiniLM to generate embeddings, and finally opted for the E5 embedding model using a parent-child chunking structure. To further improve performance, I also applied a semantic splitting step before implementing the parent-child structure, generating a new embedding structure with enhanced context relevance.

All generated embeddings were stored in a Qdrant vector database. I ran Qdrant locally using Docker and connected it to the RAG system. After setting up the retrieval mechanism, I designed several prompt templates and tested different approaches to maximize generation quality.

To evaluate the performance, I compared two setups: one using the RAG pipeline and another without retrieval (baseline). Evaluation was conducted using ROUGE, BLEU, and cosine similarity metrics.

RAG Pipeline Report

All scripts, models, and results are included in the GitHub repository associated with this project.