

Table of Contents

[Introduction 3](#_Toc182760856)

[**Significance of the Project** 3](#_Toc182760857)

[Domain Overview: Travel and Tourism Industry of Pakistan 4](#_Toc182760858)

[Corpus Collection, Cleaning, and Preprocessing 5](#_Toc182760859)

[Platform Selection: Google Colab 5](#_Toc182760860)

[Selection of Language Models and Retrieval Methods: 5](#_Toc182760861)

[Integration with Hugging Face Library 6](#_Toc182760862)

[Initial Setup and Preprocessing 6](#_Toc182760863)

[Combinations of Retrieval Techniques and Embedding Models Used for Efficient RAG Pipelines with LLMs 7](#_Toc182760864)

[TF-IDF Retrieval for Efficient Document Ranking and NLP 7](#_Toc182760865)

[BM25 Retriever with LLM Response Generation (Chunk Sizes: 1000 and 500) 7](#_Toc182760866)

[Maximum Marginal Relevance (MMR) with Sentence Transformer Embedding (Chunk Sizes: 500 and 1000) 7](#_Toc182760867)

[Basic Similarity-Based Retrieval with Sentence Transformer Embedding 7](#_Toc182760868)

[FAISS with Sentence Transformer Embedding for Efficient Retrieval 8](#_Toc182760869)

[FAISS with FastText Embedding for Efficient Document Retrieval 8](#_Toc182760870)

[FAISS with GloVe Embedding for Semantic Document Search 8](#_Toc182760871)

[Evaluation Criteria for LLM Performance 8](#_Toc182760872)

[Gemma 2B LLM Overview 9](#_Toc182760873)

[LLaMA Model Overview 10](#_Toc182760874)

[Mistral Model Overview 11](#_Toc182760875)

[Qwen Model Overview 13](#_Toc182760876)

[Best LLM: Qwen 2.5-Coder (32B Instruct) 14](#_Toc182760877)

[Conclusion and Challenges 14](#_Toc182760878)

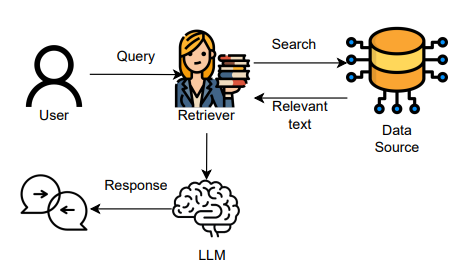
[References: 17](#_Toc182760879)

Introduction

In recent years, there has been significant progress in natural language processing (NLP), particularly with the development of advanced language models capable of understanding and generating human-like text. However, despite these advances, large language models (LLMs) like GPT-3 and GPT-4 can struggle with specific, domain-related queries, as their training data may not encompass the latest or niche information. To address this limitation, the concept of **Retrieval-Augmented Generation (RAG)** has emerged as a powerful approach for enhancing question-answering systems by integrating information retrieval with generative capabilities.

**What is Retrieval-Augmented Generation (RAG)?**

Retrieval-Augmented Generation is an advanced framework that enhances the performance of LLMs by integrating a **retrieval module** that searches a specific corpus for relevant information before generating an answer. The RAG framework works in two main stages:

* **Retrieval Stage**: The system searches for relevant documents or pieces of information from a large corpus based on the query. This can be achieved using techniques such as semantic search or keyword-based search, which help identify the most relevant content to the question.
* **Generation**: Once relevant documents are retrieved, a language generation model (such as GPT or BERT-based transformers) synthesizes an answer using the context provided by the retrieved documents. This step allows the system to generate responses that are not only factually accurate but also coherent and contextually relevant

By augmenting the LLM with retrieved context, RAG systems can:

* **Improve Accuracy**: By grounding responses in up-to-date and domain-specific documents, RAG reduces the risk of hallucinations (where the model generates incorrect or fabricated information).
* **Enhance Relevance**: The retrieval component ensures that the LLM focuses on information pertinent to the query, even if the LLM was not explicitly trained on such specific data.
* **Increase Efficiency**: Instead of fine-tuning large models on extensive domain-specific data, RAG leverages retrieval to dynamically access relevant knowledge, making it a more efficient approach.

**Significance of the Project**

* The implementation of a RAG-based question-answering system is particularly useful for specialized domains where accuracy and relevance are paramount. Traditional question-answering systems relying solely on LLMs may struggle with specific queries, especially when the training data does not cover certain niche areas. By integrating a retrieval mechanism, the RAG model can tap into external, curated sources of information, providing more accurate and context-aware responses.
* In this project, the aim was to demonstrate the effectiveness of the RAG framework by building a robust question-answering system tailored to a specific domain corpus. The system will not only retrieve relevant information efficiently but also generate coherent and contextually appropriate answers, showcasing the practical benefits of this approach in real-world scenarios.

Domain Overview: Travel and Tourism Industry of Pakistan

For this project, T**ravel and Tourism industry of Pakistan** was chosen as the domain due to its relevance, growing interest, and the wealth of available information across various subtopics. The aim was to build a question-answering system that could effectively provide insights and information across multiple facets of the tourism industry. The primary focus was on gathering a broad and diverse corpus that would enable the Retrieval-Augmented Generation (RAG) model to handle queries on a wide range of tourism-related topics.

**Scope of the Domain and Key Subtopics**

The research covered the following subtopics, aimed at providing a holistic view of Pakistan’s tourism industry:

1. **Planning and Destination Insights**: Information on travel planning and popular tourist sites across Pakistan.
2. **Adventure and Eco-Tourism**: Opportunities for adventure activities like trekking and eco-tourism experiences in natural reserves.
3. **Cultural and Religious Tourism**: Overview of historical landmarks, cultural festivals, and religious pilgrimage sites.
4. **Economic Impact of Tourism**: Analysis of tourism’s contribution to Pakistan’s GDP and economic growth.
5. **Tourism Policies and Government Initiatives**: Government efforts and policies aimed at boosting tourism.
6. **Environmental Impact and Sustainable Tourism**: The ecological effects of tourism and the importance of sustainable practices.
7. **Cultural Norms and Social Etiquette**: Guidance on local customs, etiquette, and social practices for travellers.
8. **Role of social media in Promoting Tourism**: Influence of digital platforms and travel influencers in shaping tourism trends.
9. **Comparison with Other Countries**: Comparative analysis of Pakistan’s tourism industry with neighbouring countries.
10. **Female Travel**: Insights into female solo travel experiences and safety considerations.
11. **Local Cuisine and Food Tourism**: Exploration of Pakistan’s culinary traditions and their appeal to food tourists.

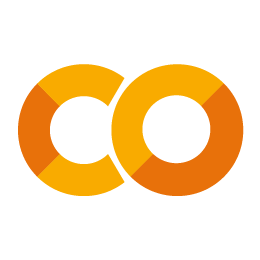
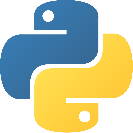
Corpus Collection, Cleaning, and Preprocessing

The corpus consists of approximately **1100 pages** of text data, gathered from a variety of sources, including **articles, research papers, blogs, official tourism websites, and Wikipedia entries**. The data collection focused on compiling relevant and authoritative content, which was primarily in the form of **PDFs and Word documents**. All the gathered files were organized, compressed, and uploaded as a zip file to a shared Google Drive for easy access.

During the preprocessing stage, following steps were undertaken:

1. **Data Cleaning**: firstly images, advertisements, and any non-text elements were removed from the collected documents to ensure the quality and consistency of the text data.
2. **Standardization**: The text content was standardized by converting all files into plain text format, removing unnecessary formatting, and combining the cleaned text into a single dataset.
3. **Consolidation**: The cleaned and standardized text data was then combined to form a comprehensive corpus, ready for use in the information retrieval and response generation stages of the RAG model.

*Details of the sources and the corpus content are provided in the* ***References*** *section of this report.*



Platform Selection: Google Colab

For the development of the Retrieval-Augmented Generation (RAG) model, **Google Colab was** utilized throughout the project. Colab was chosen for its significant advantages over local environments like Jupyter Notebook, particularly its free access to GPU and TPU resources, which facilitated faster training and processing of large datasets. Being cloud-based, Colab allowed seamless real-time collaboration among team members without the hardware limitations typically encountered on local machines. It also offered a comprehensive set of pre-installed libraries and direct integration with Google Drive, simplifying data handling and reducing setup time. Additionally, automatic cloud storage ensured continuous backup of the work, providing a reliable and efficient platform for experimentation and development needs.

Selection of Language Models and Retrieval Methods:

In the Retrieval-Augmented Generation (RAG) system, four diverse large language models (LLMs) were integrated: Llama, Gemma, Qwen, and Mistral. Each model was chosen for its unique strengths. Llama provided robust generative capabilities with strong contextual understanding, while Gemma was effective in integrating external knowledge for improved response accuracy. Qwen handled structured queries efficiently, and Mistral demonstrated versatility across a range of complex language tasks, making it a valuable addition to the model ensemble.

For retrieval, a hybrid approach was employed, combining both keyword-based and semantic search techniques. For keyword-based retrieval, BM25 and TF-IDF were used, which ranked documents based on term frequency and relevance. In semantic search, FAISS (Facebook AI Similarity Search), Maximal Marginal Relevance (MMR), and a basic similarity model were applied to capture deeper contextual relationships between queries and documents. These semantic retrieval methods leveraged embeddings such as Sentence Transformer Embedding, GloVe Embedding, and FastText Embedding, allowing us to effectively encode and compare semantic meanings.

Integration with Hugging Face Library

**Hugging Face** applied platform extensively for importing and deploying the chosen language models. Hugging Face offers a vast repository of pre-trained models, including **Llama**, **Gemma**, **Qwen**, and **Mistral**, providing easy access and integration within the RAG system. Its robust API support and streamlined library functions allowed us to efficiently load, fine-tune, and experiment with different LLMs, significantly reducing the time required for model setup and enhancing the overall development process.

Initial Setup and Preprocessing

To begin the project, a comprehensive setup on Google Colab was performed, leveraging its GPU capabilities for efficient computation. All necessary libraries were installed, including **Bitsandbytes**, **Transformers**, **LangChain**, **Sentence Transformers**, and **Chromadb**, to facilitate model access, embedding creation, and document retrieval. Additionally, **Hugging Face Hub** was integrated for seamless access to pre-trained models using an API token.

Google Drive was mounted to access the dataset and utilized **PyMuPDF** and **LangChain's** document loaders to read the PDF files stored in the corpus directory. The corpus, consisting of PDFs and DOC files, was cleaned and preprocessed by removing images and irrelevant content. Using the **CharacterTextSplitter** from LangChain, the documents were split into manageable text chunks (1,000 characters each) to enable effective processing in subsequent steps. This preprocessing ensured that the data was prepared and ready for efficient retrieval and input into the LLMs.

Questions used for querying the LLMs:

1. What are the most popular destinations in Pakistan?
2. Which historical landmarks should I visit in Lahore?
3. What is the best time to visit northern Pakistan?
4. Tell me some must-visit beaches in Pakistan.
5. Can you suggest a 7-day travel itinerary for exploring Gilgit-Baltistan?
6. Which cities in Pakistan have the best street food?
7. What safety tips should female solo travelers keep in mind when visiting Pakistan?
8. What is the temperature range in Naran during the summer season?

Combinations of Retrieval Techniques and Embedding Models Used for Efficient RAG Pipelines with LLMs

**TF-IDF Retrieval for Efficient Document Ranking and NLP**

In this approach, TF-IDF was employed to identify key documents by assessing the importance of terms within the documents using cosine similarity. After performing keyword-based retrieval, the top documents were selected based on relevance, and then used in the natural language generation pipeline. This method ensured that only the most relevant content, based on keyword significance, was passed to the LLM for further processing, resulting in highly focused answers. This approach was tested for travel-related queries, such as identifying popular destinations in Pakistan, to provide accurate and contextually rich responses.

**BM25 Retriever with LLM Response Generation (Chunk Sizes: 1000 and 500)**

Using the BM25 retrieval method from LangChain, documents were ranked based on term frequency, document length, and keyword matches. Two chunk sizes (1000 and 500) were experimented to evaluate how granular document splitting affected the retrieval quality. BM25 efficiently retrieved the top 4 documents, focusing on keyword matches within the text. These relevant documents were then passed through the LLM pipeline, allowing the model to generate answers based on the selected context. The combination of BM25 with LLMs aimed to provide more efficient and relevant responses by leveraging keyword-focused retrieval.

**Maximum Marginal Relevance (MMR) with Sentence Transformer Embedding (Chunk Sizes: 500 and 1000)**

MMR was used with Sentence Transformer embeddings (all-MiniLM-L6-v2) to rank documents based on both diversity and relevance. This method first converts the text into semantic vector embeddings and then ranks the documents based on their relevance to the query, while also considering the diversity of the content. The retrieval was configured with chunk sizes of 500 and 1000 to test the impact of document granularity on performance. MMR with Sentence Transformer Embedding retrieved the top 4 most relevant documents from a batch of 20 documents and passed them to the LLM for generating contextually accurate responses. This configuration was tested across different LLMs to understand how different models processed the varied retrieval configurations.

**Basic Similarity-Based Retrieval with Sentence Transformer Embedding**

In this method, basic similarity-based retrieval using Sentence Transformer embeddings was employed to retrieve the most relevant documents from a Chroma vector store. The embeddings were used to convert the text into high-dimensional vectors, and cosine similarity was then applied to identify the most similar documents to the query. The context from these documents was passed through the LLM pipeline, enabling the model to generate answers based on the retrieved content. This straightforward, similarity-driven approach was tested with multiple LLMs, ensuring that the context provided was directly relevant to the queries.

**FAISS with Sentence Transformer Embedding for Efficient Retrieval**

FAISS was used for efficient nearest neighbor search on embedded document vectors. By employing Sentence Transformer embeddings, the top 5 most relevant documents were able to be retrieved for a given query. These documents were selected based on semantic similarity to the query. With FAISS handling the retrieval, the pipeline ensured fast and accurate retrieval of relevant documents, which were then passed to the LLM for answer generation. This setup was tested with two chunk sizes (500 and 1000) to evaluate how different chunk sizes impacted retrieval performance and LLM response generation.

**FAISS with FastText Embedding for Efficient Document Retrieval**

For this configuration, FastText embeddings were generated for the document corpus by averaging the word vectors of each document. The pre-trained FastText model (cc.en.300.bin) was used to obtain semantic word-level embeddings. These embeddings were aggregated to form document vectors, which were then indexed using FAISS to allow efficient retrieval of the top 5 most relevant documents for any given query. After retrieval, the documents were processed by the LLM to generate answers. This method was particularly effective for retrieving semantically rich content and was tested with the query "What are the most popular destinations in Pakistan?"

**FAISS with GloVe Embedding for Semantic Document Search**

This setup utilized GloVe embeddings, specifically the 100-dimensional model (glove.6B.100d.txt), for document representation. The GloVe model was loaded, and a custom embedding function was created to average the word vectors of each document. NLTK was used for tokenization, and the average vector of all the words in a document was calculated. These embeddings were indexed with FAISS, allowing for fast retrieval of the top 5 most relevant documents. The retrieved documents were passed to the LLM for response generation. This method was tested to evaluate how well the GloVe embeddings and FAISS worked in combination to provide relevant and accurate answers.

Evaluation Criteria for LLM Performance

The overall performance of all the LLMs and their models were assessed using a comprehensive scoring system based on seven key parameters:

1. **Relevancy**: How well the generated response aligns with the context of the retrieved documents.
2. **Accuracy**: The factual correctness of the response content.
3. **Completeness**: The extent to which the answer covers all aspects of the query.
4. **Length**: Appropriateness of the response length, ensuring it's neither too brief nor overly detailed.
5. **Coherence**: Logical flow and connectivity within the generated response.
6. **Fluency**: Grammatical correctness and natural language usage in the response.
7. **Time Efficiency:** Evaluated the response speed of each LLM, factoring in model size, retrieval method, and processing power (CPU vs. GPU). This metric helped measure the overall latency and promptness of the generated answers, contributing to the assessment of usability in real-time scenarios.

Each response was rated on a scale of 1 to 5 for these parameters, and the scores were averaged to determine the overall model score.

Gemma 2B LLM Overview

In this project, the **Gemma 2B** model served as the primary LLM, chosen for its strong generative capabilities and compatibility with retrieval-based architectures. Gemma 2B was integrated into the Retrieval-Augmented Generation (RAG) pipeline, acting as the response generation layer for all retrieval configurations, including TF-IDF, BM25, MMR, and FAISS. It was particularly selected for its ability to process large amounts of travel-related information and generate highly accurate and context-aware responses.

Gemma 2B effectively combined with various embeddings (like Sentence Transformer, FastText, and GloVe) and retrieval methods, handling both large (chunk size 1000) and smaller document chunks (chunk size 500). The model was tested across different scenarios, focusing on providing comprehensive answers to detailed travel queries, ensuring the relevance of the retrieved content, and adapting dynamically to the input context.

*Refer to* ***Table 1.1*** *for a detailed overview of the performance metrics across all retrieval methods evaluated for Gemma.*

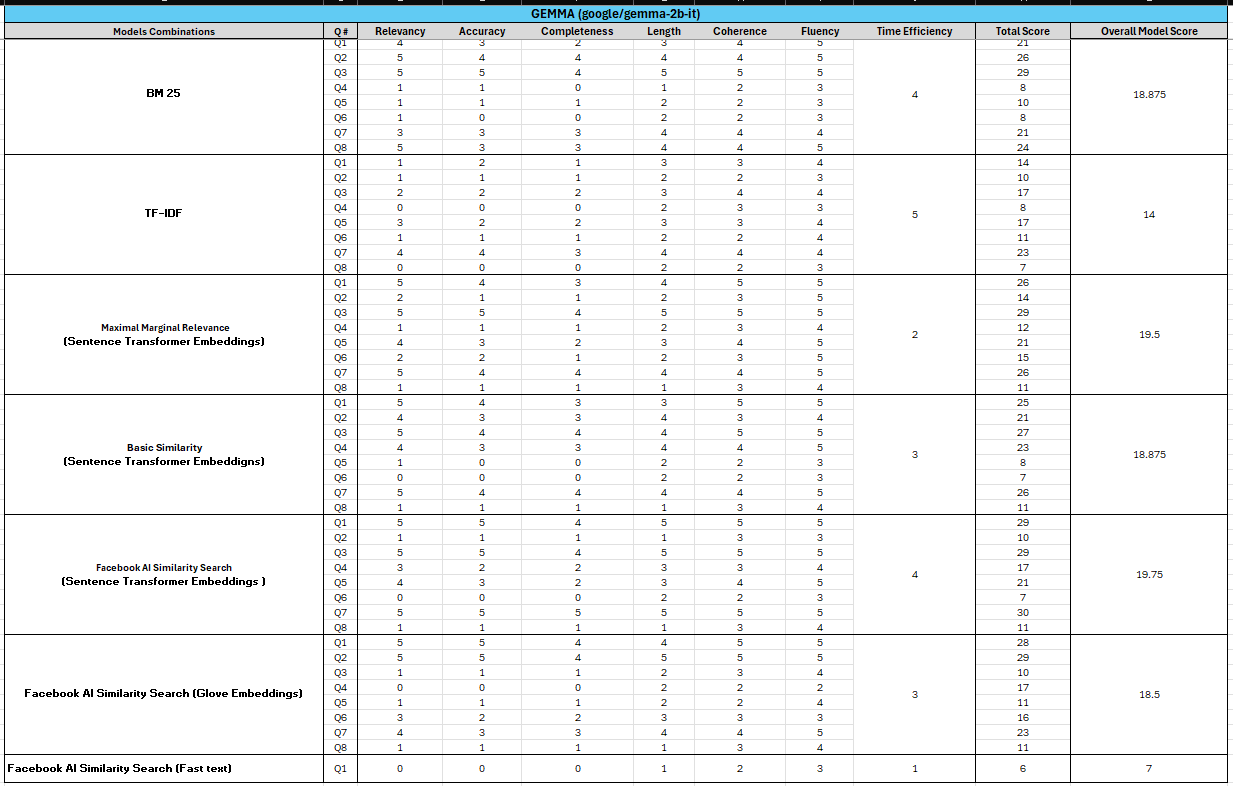
**Best Result: FAISS with Sentence Transformer Embeddings**

The highest performance score for Gemma 2B was achieved using the **FAISS with Sentence Transformer Embeddings**, yielding an average score of **19.75**. This setup utilized the **all-MiniLM-L6-v2** embedding model, combined with FAISS for efficient vector search. The method's success can be attributed to the strong semantic representation of the Sentence Transformer embeddings and the precise similarity search capabilities of FAISS, which together enabled accurate and contextually relevant retrievals. Gemma 2B effectively utilized this retrieved context to generate detailed and coherent responses, excelling across all evaluation parameters, particularly in **Relevancy**, **Completeness**, and **Fluency**.

**Worst Result: TF-IDF Retrieval**

The lowest performance score for Gemma 2B was observed with the **TF-IDF Retrieval**, which resulted in an overall score of **14**. The TF-IDF approach, based purely on keyword frequency, lacked the semantic depth required for understanding complex queries and often retrieved irrelevant documents. This led to lower scores in **Relevancy** and **Accuracy**, as the context provided was frequently misaligned with the query intent. Additionally, the generated responses were less coherent and complete, highlighting the limitations of relying solely on keyword-based retrieval in scenarios requiring nuanced understanding and deep semantic relationships.

Table 1.1



LLaMA Model Overview

The second LLM employed in the project was **LLaMA 3.2-1B Instruct**, accessed through a Hugging Face endpoint. This model, developed by Meta, is designed to be highly efficient in understanding and generating natural language responses. It utilizes an instruction-tuned version of the LLaMA model, optimized for tasks requiring precise comprehension and user-guided outputs. The model's configuration was set with a low temperature (0.1) to ensure deterministic and focused responses, reducing variability while maximizing accuracy and coherence.

*Refer to* ***Table 1.2*** *for a detailed overview of the performance metrics across all retrieval methods evaluated for LLaMA.*

**Best Result: FAISS with Sentence Transformer Embeddings**

The best performance for LLaMA was achieved using **FAISS with Sentence Transformer Embeddings**, which scored **19**. The combination of FAISS for efficient vector search and the powerful semantic encoding from the **all-MiniLM-L6-v2** Sentence Transformer enabled precise document retrieval. This led to high-quality context being provided to LLaMA, resulting in responses that excelled in **Relevancy**, **Completeness**, and **Fluency**, particularly for complex and nuanced queries.

**Worst Result: Facebook AI Similarity Search (Fast text)**

Interestingly, the worst score for LLaMA was also observed with **Facebook AI Similarity Search (Fast text)**, yielding a low score of **5**. This significant drop in performance highlights the variability in how well the retrieved context aligned with different queries. In cases where the retrieved documents were not closely related to the query, LLaMA struggled to generate accurate and coherent responses, leading to lower scores across the evaluation metrics, especially in **Relevancy** and **Coherence**. This underscores the importance of effective retrieval alignment in enhancing LLM outputs.

Table 1.2

A screenshot of a document

Description automatically generated

Mistral Model Overview

The third LLM used in the project was **Mistral 7B v0.1**, executed on a GPU for optimized performance. This model, developed by Mistral AI, is known for its large parameter size (7 billion), providing enhanced capabilities in handling complex language tasks. The Hugging Face Endpoint setup included GPU acceleration (device="cuda") and specific configurations such as a low temperature (0.1) and a maximum token length of 500 to ensure controlled, focused responses. By leveraging GPU computation, the model's processing speed was significantly improved, allowing efficient handling of larger context inputs.

However, during testing, **Mistral exhibited a notable tendency for hallucination**, frequently generating irrelevant information that was not present in the retrieved documents. Additionally, it often **repeated parts of its answers, along with empty spaces instead of answers**, diminishing the quality and coherence of the responses. This issue was observed across different retrieval strategies, highlighting a limitation of the model in maintaining context alignment and avoiding redundant information.

*Refer to* ***Table 1.3*** *for a detailed overview of the performance metrics across all retrieval methods evaluated for Mistral.*

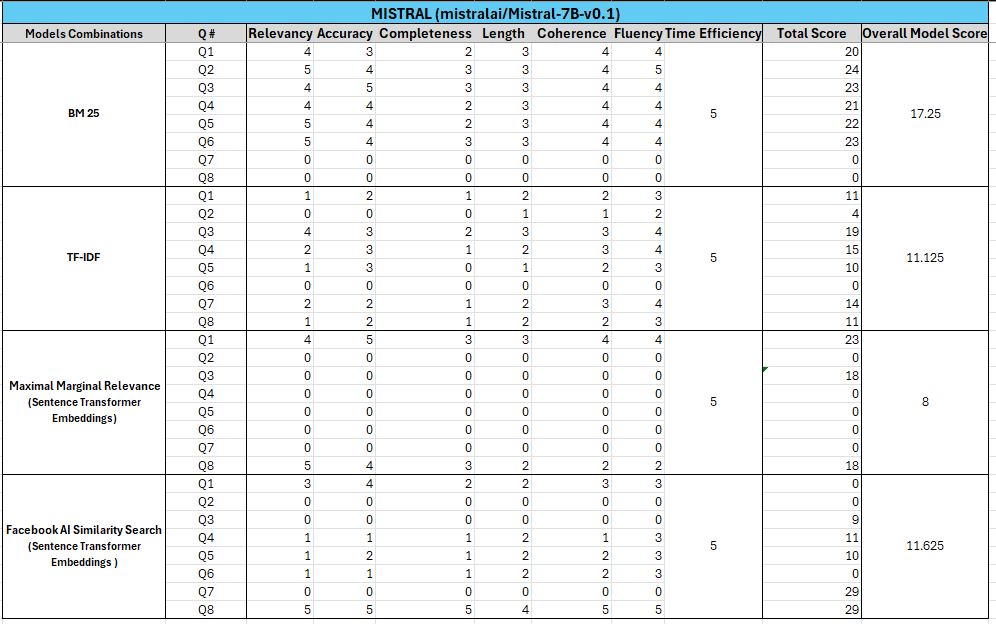
**Best Result: BM25 Retriever**

The best score for Mistral was achieved with the **BM25 Retriever**, which scored **17.25**. BM25, known for its keyword-based ranking, effectively matched query terms with relevant documents, providing concise and accurate context. This retrieval method excelled in selecting high-quality passages, contributing to strong scores in **Relevancy**, **Completeness**, and **Accuracy**. The performance boost demonstrates how traditional retrieval methods like BM25 can complement large-scale LLMs by focusing on precise term matching.

**Worst Result: Maximal Marginal Relevance (MMR) with Sentence Transformer Embeddings**

The worst result for Mistral was obtained using **Maximal Marginal Relevance (MMR) with Sentence Transformer Embeddings**, scoring only **8**. Despite the sophisticated semantic ranking capability of MMR, the retrieval quality suffered due to suboptimal alignment of the retrieved context with the input queries. This mismatch led to poor scores in **Coherence** and **Relevancy**, indicating that the semantic diversity emphasized by MMR did not effectively support the specific needs of Mistral's generation process in this setup. This highlights the challenges of balancing diverse retrieval strategies with the specific requirements of different LLMs.

Table 1.3



Qwen Model Overview

The fourth LLM integrated in this project was **Qwen 2.5-Coder (32B Instruct)**, a highly advanced model designed for complex natural language understanding and code-related tasks. With a substantial 32 billion parameters, Qwen 2.5 stands out for its exceptional capacity to handle intricate prompts and deliver nuanced responses. The model was configured through a Hugging Face Endpoint with GPU acceleration (device="cuda") to leverage the powerful processing capabilities, ensuring efficient response times and improved performance for detailed queries. Throughout testing, Qwen demonstrated exceptional performance across all retrieval methods, delivering precise and contextually relevant answers. However, it did exhibit a tendency for hallucinations, often introducing unrelated information or repeating segments of the response. Despite this, Qwen consistently provided accurate, to-the-point answers. Notably, its answers were occasionally truncated, especially when using certain retrieval methods, limiting the completeness of the response.

Qwen was particularly chosen for its strength in code comprehension and instruction-following abilities. The temperature setting of 0.1 aimed to maintain deterministic output, while max\_length and max\_new\_tokens parameters were set to prevent excessively verbose answers and enhance focus.

*Refer to* ***Table 1.4*** *for a detailed overview of the performance metrics across all retrieval methods evaluated for Qwen.*

**Best Result:** The highest score achieved by Qwen was **27.714**, recorded with **Facebook AI Similarity Search (Sentence Transformer Embeddings)**. This result demonstrated Qwen's ability to effectively utilize semantic-based embeddings for document retrieval, ensuring that the most contextually relevant information was included in its responses. The model maintained a high degree of accuracy, fluency, and coherence, providing highly relevant answers while staying concise and direct.

**Worst Result:** The lowest score recorded for Qwen was **12.375** with **TF-IDF** retrieval. This result highlighted the limitations of TF-IDF in capturing deeper semantic relationships between words, which led to less relevant document retrieval. Consequently, the model's responses lacked the depth and accuracy typically seen in other configurations, impacting the completeness and overall quality of the answers. Despite these challenges, Qwen still provided coherent answers, though the information was less aligned with the query's intent.

Table 1.4

A screenshot of a computer

Description automatically generated

Best LLM: Qwen 2.5-Coder (32B Instruct)

**Qwen 2.5-Coder (32B Instruct)** stands out as the best-performing model due to its consistent excellence across key parameters like **relevancy**, **accuracy**, **coherence**, and **fluency**. It provided concise, to-the-point answers, consistently delivering high-quality responses.

**Why Qwen Outperformed All Others:**

1. **Precision & Relevance:** Qwen’s responses were highly relevant and accurate, staying on-topic and addressing queries effectively.
2. **Fluency & Coherence:** It maintained natural, coherent answers without repetition or irrelevant content, ensuring readability.
3. **Superior Retrieval:** The integration of **Facebook AI Similarity Search (Sentence Transformer Embeddings)** enhanced its ability to retrieve contextually accurate information.
4. **Reliable Across Retrieval Methods:** Qwen showed consistent performance across all retrieval techniques, outperforming other models in terms of both quality and consistency.

Conclusion and Challenges

This project explored the performance of four large language models (LLMs)—**Qwen**, **Mistral**, **Llama**, and **Gemma**—integrated with various retrieval methods like **TF-IDF**, **BM25**, **Facebook AI Similarity Search**, and **FastText Embeddings**. Among all the models, **Qwen 2.5-Coder (32B Instruct)** proved to be the best, offering the most concise, relevant, and fluent answers across all retrieval methods. It excelled in terms of relevancy, coherence, and fluency, setting it apart from other models in the comparison.

However, the journey wasn't without challenges. The performance of the retrieval methods varied significantly depending on the LLM used, with **TF-IDF** consistently underperforming across all models, delivering poor results. Additionally, handling multiple embeddings was unmanageable and required careful handling to ensure consistent processing across various chunk sizes. Some embeddings, such as **FastText**, took significantly longer to download, which restricted their use in some cases. This added to the complexity of the process, limiting experimentation with certain configurations.

The **sequence of models** ranked from best to worst was **Qwen** > **Mistral** > **Llama** > **Gemma**. While **Qwen** provided stable and efficient results, some models—particularly **Gemma** running on a CPU—took excessively long to generate answers and frequently encountered runtime errors. This forced us to repeatedly run queries to obtain all results, especially when the model reported being "too busy."

Handling **chunk sizes** was another challenge, as different retrieval methods required fine-tuning of the chunk sizes to ensure the model received relevant context. Given the four embeddings used, it was important to process the embeddings consistently and adapt them to the specific needs of each query and model. Deciding on the most optimal chunk size for each LLM and retrieval method added complexity to the workflow.

Additionally, the task of **choosing the best model** proved difficult. Since each model performed differently based on the embeddings used and the nature of the questions, making comparisons was challenging. Some LLMs showed better results with specific retrieval methods, while others struggled, making the selection of a clear winner more nuanced.

**Evaluation Challenges:**

* **Tokenization and Preprocessing:** Different models required different tokenization strategies, and ensuring consistency across tokenization methods was vital to maintain accurate processing.
* **Chunking:** Dividing the document corpus into chunks that aligned with the model’s context window size while still maintaining the meaningful structure of the information was tricky.
* **Batch Processing:** Ensuring proper batch processing without overloading the models, especially with large chunks of data, posed an issue in some cases.
* **Corpus Collection:** The corpus needed to be highly relevant, comprehensive, and well-structured to support effective retrieval and response generation. Ensuring the quality of the data and keeping it updated was a challenge.

**Query Formulation and Context:**

Formulating the right questions to ensure that the Retrieval-Augmented Generation (RAG) pipeline could return contextually relevant responses was an ongoing challenge. It required balancing the need for specificity in questions with the ability of the model to retrieve and process answers effectively. Additionally, gauging whether the generated response was coming from the **provided corpus** or had been learned by the model during training was difficult. This often-required manual validation and comparisons against the original documents.

In conclusion, while the project presented a clear winner in **Qwen 2.5-Coder**, it also revealed the complexities involved in integrating LLMs with different retrieval methods and embeddings. The variability in LLM performance, challenges related to chunking and embedding processing, as well as the difficulty in ensuring consistent and meaningful results, highlighted the need for careful model selection and parameter tuning in retrieval-augmented tasks.

References:

Blogs and Webpages

* <https://www.researchgate.net/publication/363472659_Adoption_of_Digital_Marketing_Strategies_in_Tourism_Industry_A_Case_Study_of_Pakistan>
* <https://medium.com/@mohammed97ashraf/building-a-retrieval-augmented-generation-rag-model-with-gemma-and-langchain-a-step-by-step-f917fc6f753f>
* <https://www.pakistantravelblog.com/>
* <https://pakistantourntravel.com/>
* <https://www.paradigmshift.com.pk/tourism-of-pakistan/>
* [https://tdap.gov.pk/wp-content/uploads/2022/04/Updated\_Research-Report-on-Tourism-converted.pdf](https://tdap.gov.pk/wp-content/uploads/2022/04/Updated_Research-Report-on-Tourism-converted.pdf%20)
* <https://www.weforum.org/stories/2022/05/tourism-not-just-about-travel-also-peace/>

Corpus Reference

* <https://drive.google.com/drive/folders/19yEVDKDa-RrdPwRIF4jbTGOMTGGHKrXz>