RAG-based Question Answering System (Indic Language Focus - Culture Domain)

This document provides a comprehensive overview of a Retrieval-Augmented Generation (RAG) based question answering system, tailored for high performance on Indic language data sources, specifically focusing on the "culture" domain within Wikipedia. This system is designed to fulfill the following requirements:

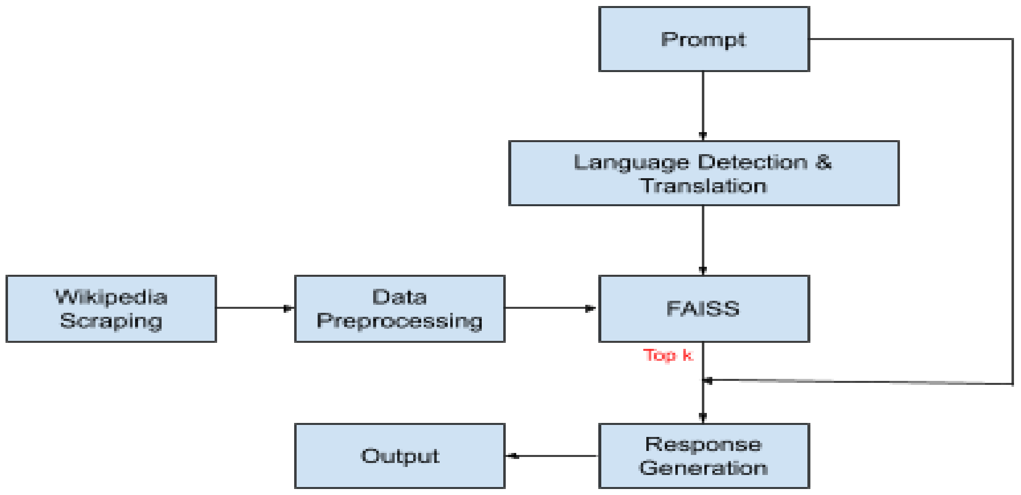
- **High-Performance RAG Pipeline:** Build an efficient RAG pipeline.
- **Indic Language Data Source:** Utilize Wikipedia as the primary data source, indexing approximately 5000 pages.
- **Multilingual Queries:** Support queries in both English and Indic languages.
- **Comparative Reporting:** Enable the creation of comparative reports on models and optimization techniques.
- **Domain Concentration:** Focus on a specific domain, such as "culture"

1. Purpose

The system aims to create a robust RAG pipeline that can effectively answer user queries related to sports, drawing information from a large, indexed Indic language Wikipedia dataset. This pipeline is optimized for performance and allows for comparative analysis of different language models and optimization strategies.

2. System Architecture

The system follows a RAG architecture, comprising the following components:



- **Wikipedia Scraping Module:** Retrieves approximately 5000 Wikipedia pages related to culture.
- **Data Preprocessing Module:** Cleans and prepares the scraped data for indexing.
- **Indexing Module (FAISS):** Creates an index of the preprocessed data for efficient retrieval.
- **Language Detection and Translation Module:** Handles language detection (different Indic language) and translation of user queries.
- **Retrieval Module:** Retrieves relevant documents from the index based on the user query.
- **Response Generation Module (T5):** Generates a response based on the retrieved information and the user query.

3. Code Breakdown

3.1 Configuration Variables:

- `MODEL_NAME`:
  - Specifies the T5 model to use for response generation.
  - Options: `"t5-small"`, `"t5-base"`, `"t5-large"`.
  - Uncomment the desired model for use.
- `LANGUAGE`:
  - Specifies the language for Wikipedia scraping and query input.
  - Options: `"hi"` (Hindi), `"en"` (English), `"bn"` (Bengali), `"mr"` (Marathi), `"ta"` (Tamil), `"te"` (Telugu).
  - Uncomment the desired language for use.
- `QUERY`:
  - Defines the query to be answered.
  - The query is defined conditionally based on the selected language, allowing for multilingual support.
  - Uncomment the desired query for use. Current active query is "Cultural Query 3: World Dance Forms"
- `TOP_K`:
  - Specifies the number of top-k documents to retrieve from the FAISS index.

3.2 Language Code Mapping and Topic Mapping:

- `LANGUAGE_MAPPING`:
  - Maps language codes to Wikipedia language codes.
  - Used to ensure correct Wikipedia language retrieval.
- `TOPIC_MAPPING`:

    - Maps language codes to relevant topics in Wikipedia.
    - Currently set to "Culture", but can be changed to "Sports" to concentrate on sports related articles.

3.3 Wikipedia Scraping Module (`scrape_wikipedia`):

- **Function:** `scrape_wikipedia(pages=5000, language=LANGUAGE)`
- **Purpose:** Retrieves approximately 5000 Wikipedia pages related to the specified topic and language.
- **Implementation:**
    - Uses `wikipediaapi` to interact with the Wikipedia API.
    - Retrieves the main page for the specified topic.
    - Iterates through links on the main page and retrieves content from linked pages.
    - Includes error handling and prevents duplicate scraping.
- **Parameters:**
    - `pages`: The number of pages to retrieve.
    - `language`: The language for Wikipedia retrieval.
- **Returns:** A list of strings, each representing the text of a Wikipedia article.

3.4 Data Preprocessing Module (`preprocess_text`):

- **Function:** `preprocess_text(text)`
- **Purpose:** Cleans and prepares text data.
- **Implementation:** Removes extra spaces and references.
- **Parameters:** `text`: The text to preprocess.
- **Returns:** The preprocessed text.

3.5 Indexing Module (FAISS) (`Indexing` class):

- **Class:** `Indexing`
- **Purpose:** Creates and manages a FAISS index for efficient similarity search.
- **Implementation:**
    - Uses Sentence Transformers to generate document embeddings.
    - Builds a FAISS index for the embeddings.
- **Methods:**
    - `__init__`: Initializes the class and creates the FAISS index.
    - `retrieve`: Retrieves the top-k most similar documents to a query.

3.6 Language Detection and Translation Module (`detect_and_translate`):

- **Function:** `detect_and_translate(query, target_lang="en")`
- **Purpose:** Detects the language of a query and translates it to English if necessary.

- **Implementation:** Uses `langdetect` for language detection and `deep-translator` for translation.
- **Parameters:**
    - `query`: The query string.
    - `target_lang`: The target language for translation.
- **Returns:** The translated query and the original language.

3.7 Response Generation Module (T5) (`generate_response`):

- **Function:** `generate_response(query, retrieved_docs, language, model_name=MODEL_NAME, max_response_length=200)`
- **Purpose:** Generates a response to a query based on retrieved documents and a T5 model.
- **Implementation:**
    - Uses the `transformers` library to load and use the T5 model.
    - Constructs a prompt by combining the query and retrieved context.
    - Generates a response using the T5 model.
    - Translates the response back to the original language if necessary.
    - Includes memory management.
- **Parameters:**
    - `query`: The translated query.
    - `retrieved_docs`: The retrieved documents.
    - `language`: The original language of the query.
    - `model_name`: The T5 model to use.
    - `max_response_length`: The maximum length of the generated response.
- **Returns:** The generated response.

3.8 Main Execution Block:

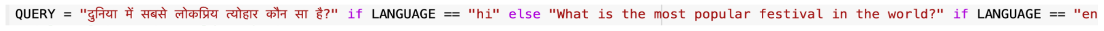
- **Purpose:** Orchestrates the entire RAG pipeline.
- **Implementation:**
    - Scrapes Wikipedia using `scrape_wikipedia`.
    - Creates a FAISS index using the `Indexing` class.
    - Detects the language and translates the query using `detect_and_translate`.
    - Retrieves relevant documents using the `retrieve` method.
    - Generates a response using `generate_response`.
    - Prints the detected language, translated query, and the response.

4. Comparative Optimization

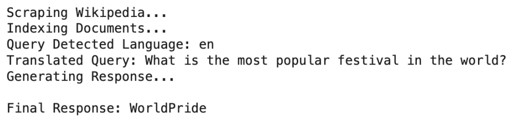
- **Model Benchmarking:**
  - Compare the performance of `"t5-small"`, `"t5-base"`, and `"t5-large"` needs to be done.
  - Record response time, accuracy, and resource usage for each model is remaining.
- **Optimization Techniques:**
  - **FAISS Index Tuning:** Experiment with different FAISS index configurations.
  - **Prompt Engineering:** Refine the prompt for better response quality. This might resolve issues related to single word output.
  - **Context Window Optimization:** Adjust `max_context_length`.
  - **Quantization:** Explore model quantization.
  - **Hardware Acceleration:** Utilize GPUs.

5. Result & Conclusion

- It's working quite well for english Queries

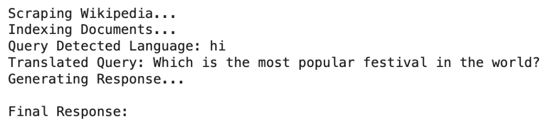


```
QUERY = "दुनिया में सबसे लोकप्रिय त्योहार कौन सा है?" if LANGUAGE == "hi" else "What is the most popular festival in the world?" if LANGUAGE == "en
```

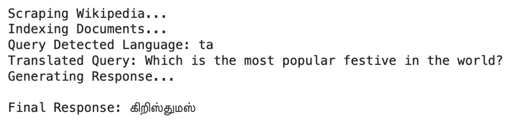


```
Scraping Wikipedia...
Indexing Documents...
Query Detected Language: en
Translated Query: What is the most popular festival in the world?
Generating Response...

Final Response: WorldPride
```

- It's failing for other Indic languages like hi **(debugged this and found out wikipedia scraping is not happening properly)**



```
Scraping Wikipedia...
Indexing Documents...
Query Detected Language: hi
Translated Query: Which is the most popular festival in the world?
Generating Response...

Final Response:
```



```
Scraping Wikipedia...
Indexing Documents...
Query Detected Language: ta
Translated Query: Which is the most popular festive in the world?
Generating Response...

Final Response: கிறிஸ்துமஸ்
```

- Another issue is resource limitation because of which I haven't used **t5-large** while t5-base is working but t5-small is failing badly.
  - `MODEL_NAME = "t5-small"`
  - `MODEL_NAME = "t5-base"`
  - `MODEL_NAME = "t5-large"`

- I can have a sparse embedding **(tf-idf,bm25)** approach that will be a faster retriever for lesser token document storage.
- Experiment with other generative models like BERT, llama, etc. is still remaining

Usage Instructions

1. Install dependencies.
2. Configure MODEL_NAME, LANGUAGE, QUERY, and TOP_K.
3. Run the script.
4. Record benchmarking data.

Dependencies

- wikipedia api
- faiss-cpu
- numpy
- sentence-transformers
- langdetect
- transformers
- deep-translator
- torch