**Detailed Report: QA System with RAG Approach and Wikipedia Integration**

**Level2**

**1. Introduction**

This report documents the development of a Question-Answering (QA) system using FAISS (Facebook AI Similarity Search) for document retrieval, Sentence Transformers for embedding, and OpenAI GPT-4 for answer generation. The system integrates Wikipedia articles and a 20 Newsgroups dataset(sports set ) to retrieve contextually relevant information. When retrieval is unsuccessful, it falls back to GPT for generating responses. Chainlit is used for the user interface, providing an interactive way to ask questions and receive answers.

Documentation Report: QA System with FAISS, LangChain, Chainlit, OpenAI GPT, and Wikipedia Integration

1. Introduction

This report documents the development of a Question-Answering (QA) system using FAISS (Facebook AI Similarity Search) for document retrieval, Sentence Transformers for embedding, and OpenAI GPT-4 for answer generation. The system integrates Wikipedia articles and a 20 Newsgroups dataset to retrieve contextually relevant information. When retrieval is unsuccessful, it falls back to GPT for generating responses. Chainlit is used for the user interface, providing an interactive way to ask questions and receive answers.

1. **System Overview**

**Core Features:**

**Document Retrieval with FAISS:**

Retrieves relevant text from embedded documents based on cosine similarity.

Uses embeddings generated by SentenceTransformer (all-MiniLM-L6-v2).

**Answer Generation:**

If relevant document chunks are found, OpenAI GPT uses them as context to generate an answer.

If no relevant context is retrieved, GPT generates an answer directly.

**Wikipedia Integration:**

Wikipedia articles are fetched for predefined topics and combined with local datasets for enhanced document retrieval.

**Chainlit UI:**

A user-friendly interface for interacting with the QA system.

1. **System Components and Architecture**

**Architecture Breakdown**

**Wikipedia Data Fetching**

wikipediaapi is used to fetch Wikipedia articles. The summaries of articles for topics such as "Artificial Intelligence," "Machine Learning," "Data Science," and "Cloud Computing" are extracted.

Articles are preprocessed (removal of extra spaces) before being embedded.

**Sentence Embedding**

A pre-trained SentenceTransformer (all-MiniLM-L6-v2) is used to convert the documents (Wikipedia + local dataset) into dense vector representations.

These embeddings are normalized for cosine similarity search.

**FAISS for Document Retrieval**

FAISS is used to index the document embeddings.

Cosine similarity is used as the metric for retrieval by normalizing the embeddings before performing the search.

**OpenAI GPT-4 Integration**

OpenAI GPT-4 generates answers using relevant retrieved chunks (context). If no chunks are retrieved, GPT-4 provides a fallback answer based solely on the query.

The system supports retries in case of RateLimitError from OpenAI.

**Chainlit UI**

Chainlit provides a simple user interface where users can ask questions, and the system provides answers in real time.

**Conclusion**

This system integrates several powerful technologies (FAISS, OpenAI GPT, Wikipedia API, Chainlit) to create an intelligent question-answering platform. By combining document retrieval and GPT's generative capabilities, it produces accurate and context-aware responses. The modular design allows for easy extension and scaling.

**How does Chainlit App looks :**

  
