Document augmented QA using Vector Store Index

Github link:
https://github.com/kalyani-Goda/Question-Answering-using-Retrieval-Augmented-Generation.git

1. Overview

This project demonstrates different methodologies for answering questions using a subset of the SQuAD dataset. The project focuses on three key approaches:

1. Simple Question Answering (QA) using LangChain without retrieval augmentation.
2. Retrieval Augmented Generation (RAG) using LangChain and FAISS.
3. RAG without using LangChain by leveraging Sentence Transformers and FAISS.

2. Dataset Description

The SQuAD (Stanford Question Answering Dataset) is a reading comprehension dataset consisting of questions posed on a set of Wikipedia articles, where the answer to every question is a segment of text from the corresponding passage. For this project, a subset of the SQuAD dataset is used.

- **Training Set (questions.csv):** Contains a list of questions that the model needs to answer.
- **Passages Set (passages.csv):** Contains corresponding passages from which answers can be extracted.
- **Validation Set (val_questions.csv):** Contains additional questions used to evaluate the model's performance.

3. Methodologies

a) Simple QA using LangChain

This approach uses the LangChain framework to perform basic question answering without retrieval augmentation. The model used is google/flan-t5-large, hosted on Hugging Face, and it generates answers directly based on the input question.

b) RAG with LangChain

This method augments question answering by retrieving relevant passages from a corpus before generating an answer. LangChain handles the retrieval using FAISS and embeddings generated by HuggingFaceEmbeddings. The retrieved passages provide additional context to the model, improving answer quality.

c) RAG without LangChain

In this approach, retrieval and answer generation are done independently of LangChain. Sentence Transformers generate embeddings for the passages and questions, and FAISS performs the retrieval. The same google/flan-t5-large model is used for generating answers.

4. Experimental Setup

Installation and Setup

Before running the code, install the necessary dependencies:

```
pip install --quiet -U langchain-community
```

```
pip install -r requirements.txt
```

Set up the Hugging Face token:

```
export hf_token="your_hugging_face_token_here"
```

Running the Script

To run the script, use one of the following commands:

Simple QA without RAG:
```
python main.py --questions ./data/val_questions.csv --output val_no_rag.csv
```

RAG using LangChain:

```
python main.py --questions ./data/val_questions.csv --rag --langchain --passages ./data/passages.csv --output val_rag_langchain.csv
```

RAG without LangChain:

```
python main.py --questions ./data/val_questions.csv --rag --passages ./data/passages.csv --output val_rag.csv
```

Streamlit Interface To use the Streamlit interface for real-time QA: `streamlit run main.py`

5. Results

Sample Outputs

Here are some sample results from running the three different approaches:

1. **Simple QA without RAG:**
   - **Question:** "What is the capital of France?"
   - **Answer:** "Paris"
2. **RAG with LangChain:**
   - **Question:** "What is the capital of France?"
   - **Answer:** "Paris"
   - **Relevant Documents:** [Passage 1, Passage 2, ...]
3. **RAG without LangChain:**
   - **Question:** "What is the capital of France?"
   - **Answer:** "Paris"
   - **Relevant Documents:** [Passage 1, Passage 2, ...]

Comparison of Methods

- **Accuracy:** Both RAG methods provide more contextually accurate answers compared to simple QA, especially for complex questions.
- **Performance:** RAG with LangChain leverages pre-built components for efficiency, while RAG without LangChain offers more control over the pipeline.

6. Conclusion

The RAG approach, particularly when integrated with LangChain, significantly improves the accuracy of answers by providing the model with additional context. This is particularly useful for complex queries that require understanding from multiple sources.

7. Future Work

- **Scaling the Dataset:** Expanding the dataset to cover more diverse topics can help in understanding the limitations of the current models.
- **Model Fine-tuning:** Fine-tuning the models on specific domains could further improve the performance of the QA system.
- **Integration with Other Tools:** Incorporating other retrieval tools and comparing their performance with FAISS could be beneficial.