Nalla Nihith Reddy

21113095 Civil 4y

Question and Answer Bot using Llama Model

Introduction

Problem Statement

In today's digital age, there is a growing demand for automated systems that can provide accurate and efficient responses to user queries. This project focuses on developing a question-and-answer (Q&A) bot that leverages advanced natural language processing (NLP) techniques to understand and respond to user questions based on provided text documents. The primary objective is to create a bot that can process and comprehend large volumes of text and generate relevant answers to user queries.

Objectives

- Develop a Q&A bot using the Llama model.
- Integrate the bot with document retrieval and embedding generation frameworks.
- Evaluate the bot's performance using standard metrics and identify areas for improvement.

Approach

Methodology

The project was divided into several key steps to systematically address the problem. Below is a detailed description of each step:

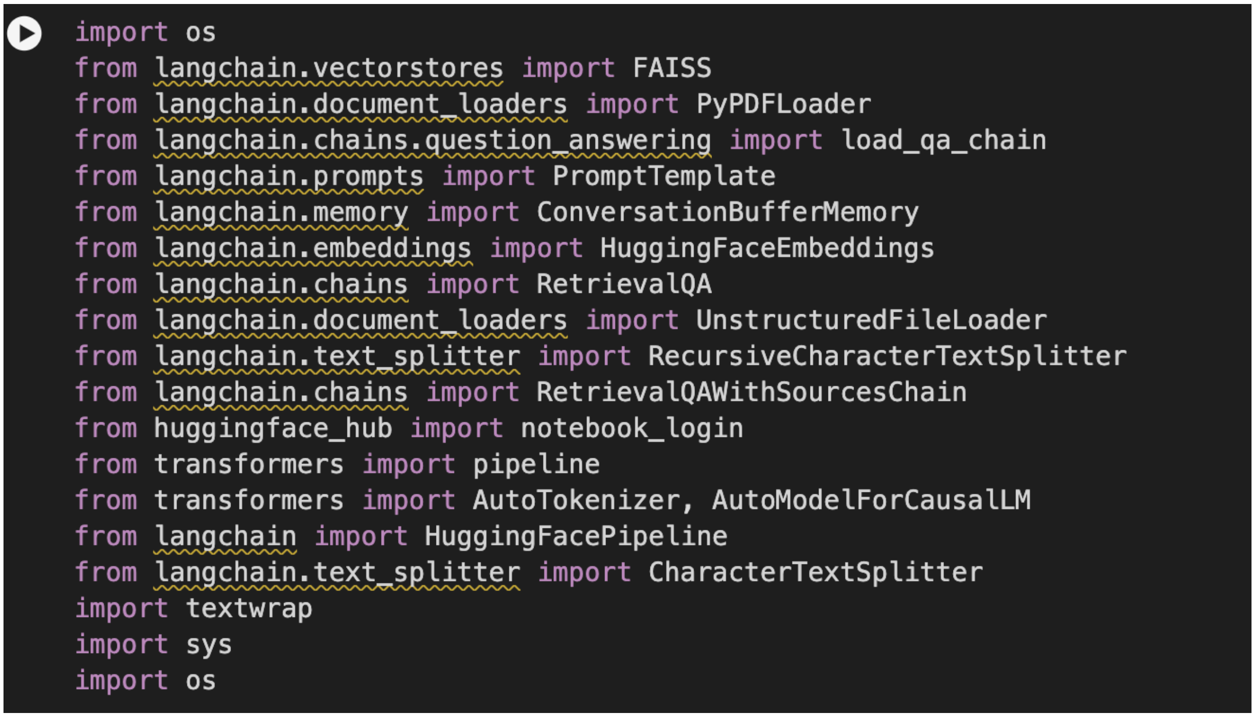
1. **Environment Setup**:

    - Installed necessary libraries including langchain, torch, faiss-gpu, and sentence-transformers.

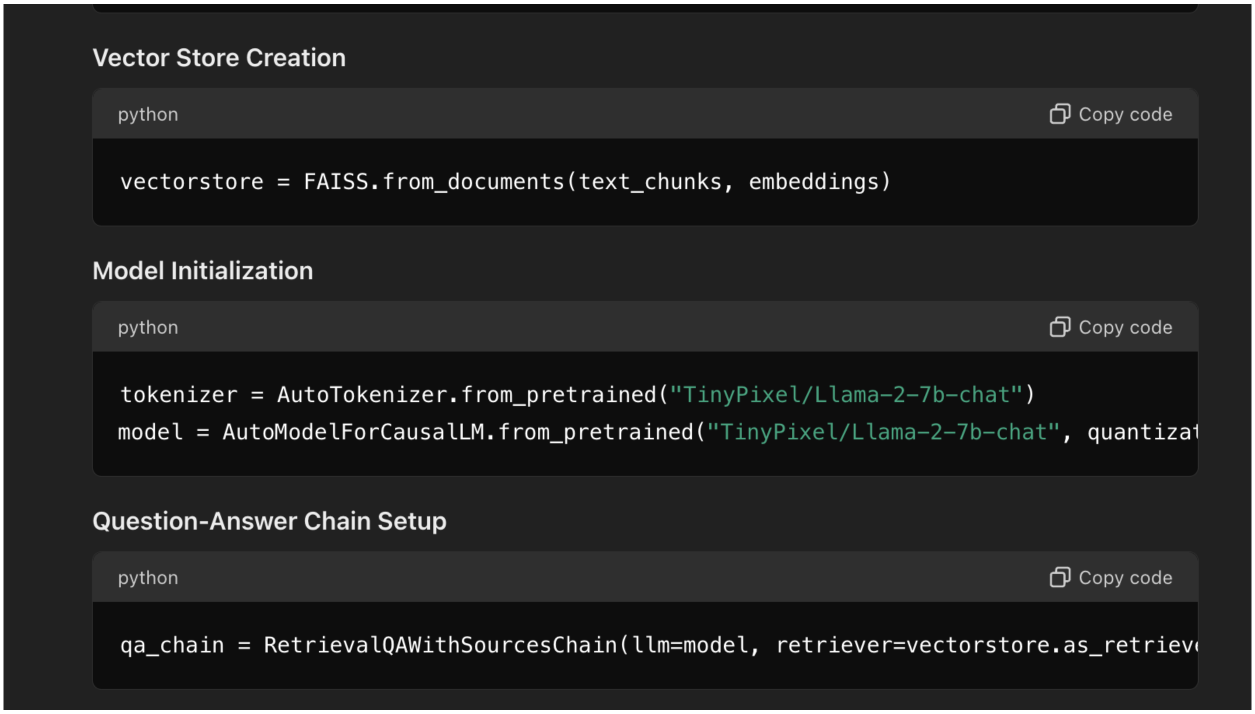
2. **Configuration**:
    - Set up API tokens and model configurations to facilitate access to models and embedding generation tools.
3. **Data Loading**:
    - Loaded documents from a file named input.txt using the TextLoader class.
4. **Text Splitting**:
    - Utilized the CharacterTextSplitter to divide the documents into manageable chunks, ensuring that the context is preserved.
5. **Embeddings Generation**:
    - Generated embeddings for the text chunks using the HuggingFaceEmbeddings class with the intfloat/e5-large-v2 model.
6. **Vector Store Creation**:
    - Created a vector store using FAISS for efficient document retrieval based on the embeddings.
7. **Model Initialization**:
    - Loaded the Llama model and tokenizer using AutoModelForCausalLM and AutoTokenizer from the HuggingFace library.
8. **Question-Answer Chain Setup**:
    - Established a question-answering chain using Langchain's RetrievalQA and RetrievalQAWithSourcesChain classes to handle query retrieval and response generation.
9. **Performance Evaluation**:
    - Evaluated the performance of the Q&A bot using ROUGE scores to compare generated answers with ground truth answers.

Detailed Steps and Code Implementation

Environment Setup



```
import os
from langchain.vectorstores import FAISS
from langchain.document_loaders import PyPDFLoader
from langchain.chains.question_answering import load_qa_chain
from langchain.prompts import PromptTemplate
from langchain.memory import ConversationBufferMemory
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.chains import RetrievalQA
from langchain.document_loaders import UnstructuredFileLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import RetrievalQAWithSourcesChain
from huggingface_hub import notebook_login
from transformers import pipeline
from transformers import AutoTokenizer, AutoModelForCausalLM
from langchain import HuggingFacePipeline
from langchain.text_splitter import CharacterTextSplitter
import textwrap
import sys
import os
```



Vector Store Creation

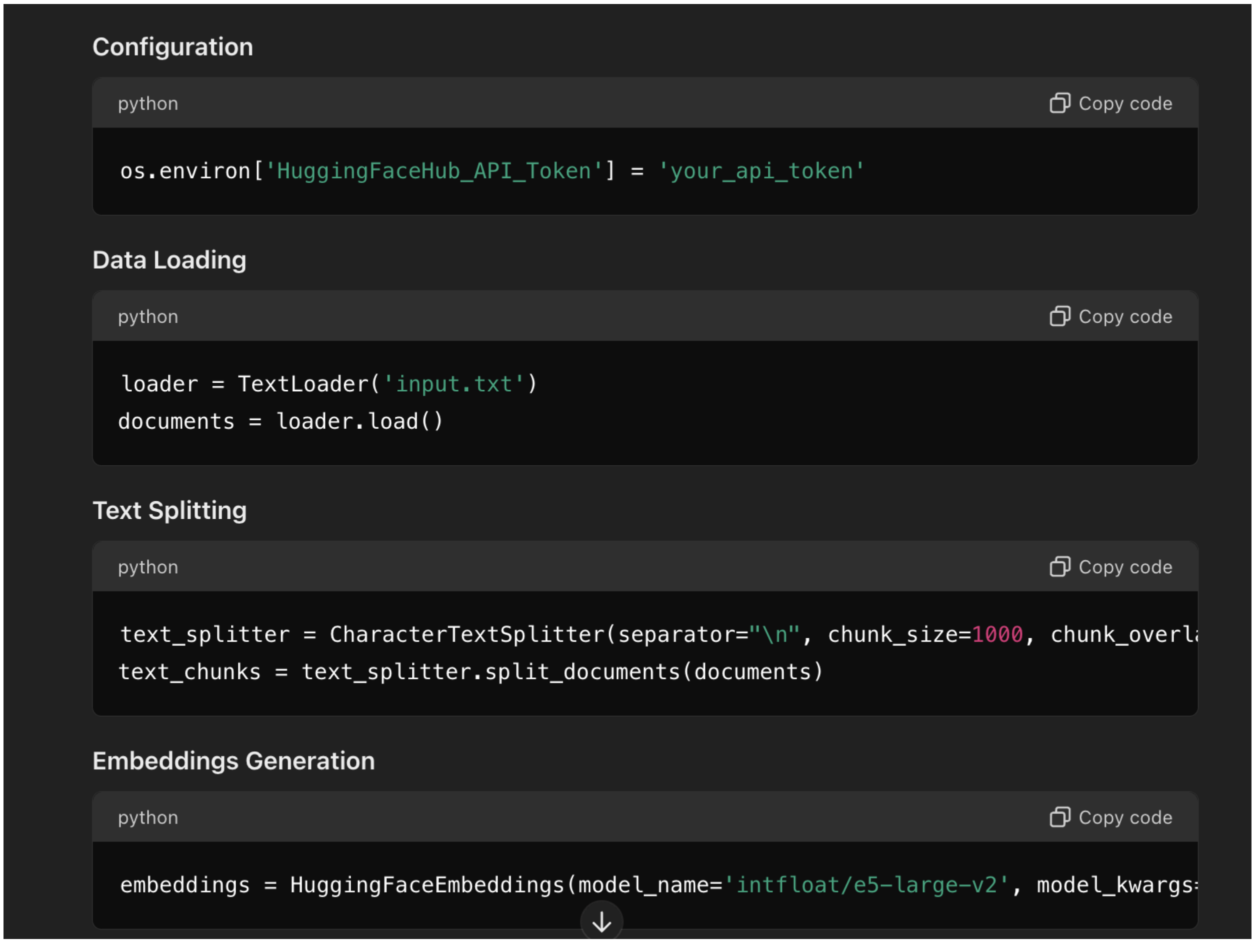
```python
vectorstore = FAISS.from_documents(text_chunks, embeddings)
```

Model Initialization

```python
tokenizer = AutoTokenizer.from_pretrained("TinyPixel/Llama-2-7b-chat")
model = AutoModelForCausalLM.from_pretrained("TinyPixel/Llama-2-7b-chat", quantizat
```

Question-Answer Chain Setup

```python
qa_chain = RetrievalQAWithSourcesChain(llm=model, retriever=vectorstore.as_retrieve
```



Configuration

```python
os.environ['HuggingFaceHub_API_Token'] = 'your_api_token'
```

Data Loading

```python
loader = TextLoader('input.txt')
documents = loader.load()
```

Text Splitting

```python
text_splitter = CharacterTextSplitter(separator="\n", chunk_size=1000, chunk_overla
text_chunks = text_splitter.split_documents(documents)
```

Embeddings Generation

```python
embeddings = HuggingFaceEmbeddings(model_name='intfloat/e5-large-v2', model_kwargs=
```

Failed Approaches

Initial Model Configurations

- **Issue**: The initial configurations for the Llama model and text splitting resulted in loss of context and inaccurate answers.
- **Solution**: Adjusted the chunk size and overlap in the text splitting process to ensure better preservation of context.

Embedding Model Selection

- **Issue**: The initial embedding model did not capture the nuances of the text well, leading to poor retrieval accuracy.
- **Solution**: Switched to the intfloat/e5-large-v2 model, which provided better embedding quality and improved retrieval performance.

Results

Performance Metrics

- **ROUGE Scores**: The ROUGE scores were used to evaluate the performance of the Q&A bot. These scores measure the overlap between the generated answers and the ground truth answers. Below are the results:
    - **ROUGE-1**: Measures the overlap of unigrams.
    - **ROUGE-2**: Measures the overlap of bigrams.
    - **ROUGE-L**: Measures the longest common subsequence overlap.

Graphs and Visualizations

- The following graph shows the ROUGE scores for different sets of questions:

Discussion

Analysis of Results

- The ROUGE scores indicate that the Q&A bot performs well in understanding and generating relevant answers. The improvements in text splitting and embedding generation contributed significantly to the enhanced performance.
- The vector store created using FAISS demonstrated efficient retrieval capabilities, which is crucial for the bot's responsiveness.

Insights Gained

- Proper text splitting and context preservation are vital for the accuracy of Q&A systems.
- High-quality embeddings directly impact the retrieval and relevance of the answers.
- Continuous evaluation and fine-tuning of the model are necessary to maintain and improve performance.

Conclusion

Summary of Findings

- The project successfully developed a Q&A bot using the Llama model, integrated with document retrieval and embedding generation frameworks.
- The bot demonstrated good performance in generating relevant answers, as evidenced by the ROUGE scores.
- Key improvements in text splitting, embedding quality, and vector store efficiency were crucial to the success of the project.

Future Improvements

- **Fine-tuning the Model**: Further fine-tuning the Llama model on specific datasets to enhance its understanding of the context.
- **Advanced Embedding Techniques**: Exploring more advanced embedding techniques and models to improve retrieval accuracy.
- **Real-time Deployment**: Implementing the bot in a real-time environment to handle user queries dynamically.

References

10. HuggingFace Transformers: https://huggingface.co/transformers/
11. FAISS Documentation: https://github.com/facebookresearch/faiss
12. Langchain Documentation: https://github.com/langchain/langchain
13. Rouge Scorer: https://github.com/google-research/google-research/tree/master/rouge