Monday, 18 November 2024

Bulk Text Analyzer

A Streamlit Based Tool for Analyzing Large Text Data

- Analyzing bulk data and extracting meaningful insights from it is critical, especially when dealing with text data.
- In this Bulk Text Analyzer project, which leverages the power of OpenAI, LangChain, and NLTK libraries to extract key terms and display word frequencies from large text files.
- This web app is built using Streamlit and provides an easy-to-use interface for uploading XML files and processing the text data to extract useful insights.
- **Bulk Text Analyzer** is a web application built using Streamlit that allows users to upload large XML files, preprocess the text data, extract key terms using OpenAI's language models, and visualize word frequencies.
- The app leverages several libraries and tools, including NLTK for text processing, LangChain for text chunking and embedding, FAISS for vector storage, and OpenAI for language modeling.

• Overview :

- The Bulk Text Analyser perform the following tasks:

1. **Preprocesses the uploaded text:** It removes punctuation, stop words, and irrelevant characters.
2. **Extract key terms:** Using OpenAI's language models, the app extracts key related words from the text.
3. **Displays word Frequencies:** The extracted words are displayed in a bar chart, showing their frequency of occurrence in the text.

- The user-friendly interface allows the user to upload a file, extract the key terms, and visualize the results instantly.
- Key Steps involved in the project are as:-

1. **Text Preprocessing:** Using the nltk library.
2. **Python 3.7+ installed on your system.**
3. **An OpenAI API Key.**

4. **app.py:** The main Streamlit application
5. u**tils.py:** Helper functions for text processing and analysis.
6. **requirements.txt:** List of dependencies.
7. **README.md:** Project documentation.

- **Importing libraries(**utils.py**)**
- **Text Preprocessing(**utils.py**)**
- **Text Chunking and Embedding(**utils.py**)**
- **Vector Store Creation(**utils.py**)**
- **Conversational retrieval chain(**utils.py**)**
- **Streamlit Frontend(**app.py**)**
- **Detailed Explanation:**

1. **Importing Libraries:** We import necessary libraries in utils.py. These include nltk for text processing, langchain for handling text chunking and embeddings, faiss for vector storage, and openai for accessing language models.
2. **Text Preprocessing:** The preprocess text function removes numbers, short words (1-2 characters), punctuation, and stopwords from the text. This cleans the data and prepares it for analysis.
3. **Text Chunking and Embedding:** split_text_into_chunks breaks the text into manageable chunks using CharacterTextSplitter from LangChain. This helps in efficiently processing large texts.
4. **Vector Store Creation:** create_vector_store_from_text_chunks creates vector embeddings of the text chunks using OpenAI embeddings and stores them in a FAISS vector store for efficient similarity searches.
5. **Conversational Retrieval Chain:** get_conversation_chain initializes a conversational retrieval chain using LangChain and OpenAI's gpt-4-turbo model. This chain facilitates interactive queries over the text data.
6. **Streamlit Frontend:** In app.py, we build the Streamlit interface:

- **Sidebar:** Contains inputs for the OpenAI API key and file uploader.
- **Main Area:** Displays the title, description, and results.
- **Processing Flow:**

1. **Upload XML File:** Users upload their XML files.
2. **Enter API Key:** Users provide their OpenAI API key.
3. **Analyse:** On clicking the "Analyze" button, the app processes the text:

- Parses the XML to extract text.
- Preprocesses the text.
- Splits text into chunks.
- Creates a vector store.
- Initializes the conversation chain.
- Extracts key terms using OpenAI.
- Counts word frequencies.
- Visualizes the top 20 words in a bar chart.
- **Conclusion:**

You've now built a **Bulk Text Analyzer** using Streamlit, complete with text preprocessing, key term extraction using OpenAI's language models, and interactive visualizations. The application is user-friendly and can handle large XML files efficiently. Deploying the app ensures it's accessible to your target audience. Feel free to further enhance the app by adding more features, improving the UI, or optimizing performance.