Llama2 Project for MetaData Generation using FAISS and RAGs Project Overview

Overview
Metadata refers to descriptive information about data that provides context, meaning, and structure. In simpler terms, metadata is "data about data." It plays a crucial role in organizing, managing, and understanding various types of information.

Examples of Metadata:
Text Document: Author, Title, Word Count, File Format
Web Page: URL, Last Modified, Meta Tags

This descriptive information aids in efficient search and retrieval, allowing users to find relevant data quickly. Metadata ensures proper interpretation of data, supporting decision-making processes. In diverse fields like libraries, archives, and digital asset management, metadata enhances workflow efficiency and fosters interoperability. Generally, metadata is created by manually documenting relevant details associated with the data.

Manual metadata creation can be time-intensive, demanding meticulous effort and attention to detail. The process involves identifying, cataloging, and inputting relevant information about a dataset, such as its source, context, and characteristics. This often consumes much of a user's time, especially when dealing with extensive datasets. Large Language Models (LLMs) offer a transformative solution to streamline this workflow. By leveraging advanced natural language processing capabilities, LLMs can automate metadata generation, drastically reducing the time and effort required for manual entry. These models excel at understanding and extracting meaningful information from text, making them highly efficient in annotating datasets with accurate and comprehensive metadata. Integrating LLMs into metadata creation workflows not only accelerates the process but also enhances the overall accuracy and consistency of metadata, allowing users to allocate their time and resources more effectively to other aspects of their projects.

In this project, we are implementing a metadata generation system using Large Language Models (LLMs) and a vector database. The system leverages Retrieval-Augmented Generation (RAG) techniques, RAG combines information retrieval with language generation, allowing the model to retrieve relevant information from the vector database before generating coherent and contextually appropriate responses, thereby improving the overall quality and relevance of the generated metadata.

Prerequisite Project: Kindly ensure the completion of the LLM Project for building and fine-tuning a large language model before proceeding with this project.

Note: Utilizing AWS services for this project may result in charges; it is essential to thoroughly review the AWS documentation to understand the pricing structure and potential costs associated with different resources and usage patterns.

Aim

The project aims to streamline metadata generation using large language models, specifically leveraging Llama2, RAGs, and AWS. It focuses on automating metadata creation processes to enhance workflow efficiency and reduce manual efforts.

Data Description

The dataset includes project documentation text files and code files. These files represent project-related documents containing various information, including project details, code implementations, and associated documentation. Leveraging this dataset, the project aims to automate and enhance metadata creation for various machine-learning projects.

Tech Stack

- ➔ Language: Python 3.10.4
- ➔ Libraries: langchain, torch, Textract, Tika, HuggingFace Transformers, Faiss
- ➔ Model: Llama2 (13 billion parameters, Quantized)
- ➔ Cloud Platform: Amazon Web Services (AWS)

Approach

- Project Setup:
    - Creating and Configuring AWS EC2 instances for GPU support
    - Set up the Python environment and dependencies, ensuring compatibility with required libraries and frameworks.

- Data Extraction:

    - Identify and parse project documentation, text files, and code files for metadata generation.
    - Utilize Textract for PDF files and Tika for specific formats.
    - Use langchain to parse and extract information from various file formats, creating a unified, large text dataset.
- Embedding and Vectorization:
    - Employ the HuggingFace Transformers library to embed the large text dataset into numerical vectors.
    - Utilize a quantized version of the Llama2 language model for efficient processing.
- Vector Database Creation:
    - Build a FAISS Vector Database to store and manage the embedded text vectors.
    - Implement a defined structure for storing project-related information.
- Prompt Engineering:
    - Design a prompt template to guide the LLM in generating responses based on the provided context.
    - Emphasize context-aware answers to avoid hallucination and maintain accuracy.
- Question-Answering Pipeline:
    - Load the Llama2 model, tokenizer, and vector database for question-answering tasks.
    - Utilize Streamlit for creating a user-friendly graphical interface.

Modular code overview:

Once you unzip the modular_code.zip file, you can find the following:

```
├ code
│  ├ sample_files
│  │  └ preprocessing.py
│  ├ configs.py
│  ├ main.py
│  ├ streamlit_main.py
│  └ utils.py
├ README.MD
├ requirements.txt
└ Solution Methdology.pdf
```

Here is a brief information on the files:

- The code folder contains all the Python scripts and sample files used in the project. You can execute either the main.py file or streamlit_main.py file according to your requirements.

- The README.MD and the solution methodology file provides essential information about the project, its purpose, usage, and setup instructions.
  Kindly follow all the instructions for running the code from Readme.md file

- The requirements.txt file has all the required libraries with respective versions. Kindly install the file by using the command **pip install -r requirements.txt**

Project Takeaways

1. Understand the significance of LLMs like Llama2 for advanced text generation tasks
2. Gain insight into vector databases, particularly FAISS, for efficient storage and management of embedded text vectors
3. Learn how vector databases support quick retrieval and comparison of numerical vectors, enhancing metadata organization
4. Familiarize yourself with Llama2, an open-source LLM with 13 billion parameters
5. Learn how to use Retrieval-Augmented Generation (RAG) techniques with LLMs like Llama2 and the FAISS Vector Database
6. Explore Python, langchain, torch, Textract, Tika, HuggingFace Transformers, and AWS technologies
7. Learn the process of setting up a Python environment and dependencies for project development
8. Understand the importance of designing effective prompts to guide LLMs in generating accurate responses
9. Gain insights into implementing a streamlined pipeline for question-answering using Llama2 and the FAISS vector database
10. Understand the process of configuring AWS EC2 instances for GPU-intensive tasks