Implementation of a Retrieval-Augmented Generation Model for a FAQ System

Introduction:

This task orchestrates a comprehensive pipeline for automatic question answering from a document, employing state-of-the-art natural language processing techniques. Subsequently, it configures a generative model for text generation, loads a PDF document, and preprocesses its content to construct a word embedding vector store. The setup includes defining a conversational prompt template and configuring both conversational retrieval and language model chains for answering questions. Following this, the script reads questions from a file, generates answers using the configured models, and evaluates the quality of the responses against gold standard answers using Rouge and BLEU scores. Finally, we report the individual and overall performance metrics, providing a thorough assessment of the question answering system's effectiveness.

Models, Tokenizers, and Word Embeddings Utilized:

Generative Model: The code uses the "nvidia/Llama3-ChatQA-1.5-8B" model for text generation. This model is loaded from Hugging Face and configured for causal language modeling.

Tokenizer: The AutoTokenizer from the Hugging Face Transformers library is employed to tokenize input text for processing by the generative model. It is initialized with the tokenizer associated with the chosen generative model ("nvidia/Llama3-ChatQA-1.5-8B").

Word Embeddings: For word embeddings, the code uses the "sentence-transformers/all-mpnet-base-v2" model from Hugging Face. This model is loaded and configured for generating word embeddings. The embeddings are utilized to represent the content of the PDF document and facilitate semantic understanding during question answering.

Step-by-step Approach:

Imports:

- The code begins by importing necessary libraries and packages.
- This includes packages like transformers, torch, nltk, and others required for natural language processing tasks.

Installation:

- It installs required packages using pip.
- These include datasets, sentence-transformers, faiss-cpu, bitsandbytes, pypdf, langchain_community, langchain-text-splitters, langchain, nltk, and rouge-score.

Setting Up:

- The script sets up configurations for using a pre-trained model, including defining terminators, setting stopping criteria, and loading a pre-trained model from Hugging Face.

Loading Data:

- It loads a PDF document using a PDF loader and extracts its content.
- Then, it prepares a word embedding vector store for the document pages.

Setting Up Generative Model:

- It configures a generative model for text generation using the Hugging Face Transformers library.
- This includes setting stopping criteria, temperature, and other generation parameters.

Prompt Template:

- A prompt template is created for asking questions in a conversational manner.

Question Answering Setup:

- It sets up a conversational retrieval chain and an LLM chain for answering questions based on the loaded document content.

Helper Functions:

- Functions are defined to extract helpful answers from generated responses and to generate answers using the LLM chain.

Processing Questions:

- The script reads questions from a file, processes them one by one, and generates answers for each question using the LLM chain.

Evaluation:

- The generated answers are evaluated against gold standard answers using Rouge and BLEU scores.
- These scores are calculated both individually for each question and overall for the entire dataset.

Printing Scores:

- Finally, the script prints out the individual and overall Rouge and BLEU scores to evaluate the performance of the question answering system.

Observation:

ROUGE-1:

Precision (0.3150): About 31.5% of the words generated by the model are relevant to the reference answers.

Recall (0.6104): The model captures around 61% of the relevant words from the reference answers.

F1-Score (0.3869): This combines precision and recall, indicating that the balance between generating relevant words and covering the reference words is moderate.

ROUGE-2:

Precision (0.1916): Around 19.2% of the bigrams (two-word sequences) generated are relevant to the reference answers.

Recall (0.3486): The model captures approximately 34.9% of the relevant bigrams from the reference answers.

F1-Score (0.2315): This suggests that the model's performance in generating relevant bigrams is relatively lower than its unigram performance.

ROUGE-L:

Precision (0.2794): About 27.9% of the longest common subsequence (LCS) between the generated and reference texts is relevant.

Recall (0.5293): The model captures around 5` 3% of the LCS from the reference answers.

F1-Score (0.3408): Indicates that the model's performance in capturing relevant sequences is moderate.

BLEU (0.1294):

This score shows the overall precision of the generated text compared to the reference text, considering various n-grams (up to 4-grams by default). A BLEU score of 0.1294 indicates that the model's output has a moderate level of correspondence with the reference answers.

Challenges:

- *Model Selection and Fine-Tuning:*

Choosing an appropriate pre-trained model for question answering and fine-tuning it to suit the specific domain of the document can be challenging.

- *Limited T4 GPU Utilization:*

Since there is limited GPU usage provided, compute intensive work cannot be performed with lots of queries and answers.

- *Exact Extraction of answer:*

The exact answer should be extracted from the response provided by the LLM for all the queries. The pattern recognition should be accurate such that the answer is captured correctly.

Future improvements:

- *Domain-Specific Fine-Tuning:*

  Fine-tuning the pre-trained models on domain-specific data can improve their performance for question answering tasks related to specific fields or topics.

- *Implementation on huge context:*

  Large PDF with higher quantity of contextual information on the questions might lead to better bleu and rouge scores.

- *Adjusting hyperparameters of transformer pipeline:*

  The temperature, max_new_tokens, repetition_penalty etc should be tuned properly such that the model generates accurate output.