Document: Thought Process for Constructing the Dataset and Evaluation

1. How I constructed the Database

Data Source: The primary data source is the text extracted from uploaded PDF documents. This text serves as the basis for generating question-answer pairs through user interaction.

PDF Text Extraction: The `PyPDF2` library is used to extract text from the PDF documents. The entire text content of the PDF is extracted page by page and concatenated to form a single continuous text.

Text Splitting: The extracted text is split into smaller chunks to facilitate efficient processing and retrieval. This is done using the "CharacterTextSplitter" from the LangChain library. The parameters for splitting (chunk_size and chunk_overlap) are chosen to ensure chunks are of manageable size while maintaining contextual integrity. For example: Chunk Size: 1000 characters, Chunk Overlap: 200 characters

Embeddings Creation: The text chunks are then transformed into vector representations using OpenAI's embedding model through the "OpenAIEmbeddings" class. These embeddings enable semantic similarity searches.

Storage in Vector Store: The resulting embeddings and their associated text chunks are stored in a FAISS (Facebook AI Similarity Search) index. FAISS provides efficient similarity search capabilities, essential for retrieving relevant text chunks in response to user queries.

Question-Answer Storage: When users ask questions, the corresponding answers are stored in an SQLite database (rag_database.db). This database records each question, its answer, and a timestamp for future reference.

2. Evaluation Metrics

Semantic Similarity: This metric measures how similar the predicted answer is to the actual content in the text chunks. The Sequence Matcher from the difflib library is used for this purpose. It provides a ratio indicating the degree of similarity between two strings.

Exact Match: This metric checks if the predicted answer exactly matches any part of the text chunks. An exact match score of 1 is given if there is a perfect match; otherwise, it is 0.

Combined Match: This metric combines the semantic similarity and exact match metrics to provide a comprehensive evaluation. It

considers both exact and near-exact matches to determine the effectiveness of the model in retrieving relevant information. If the exact match score is 1 or the semantic similarity

score is above a certain threshold (e.g., 0.1), the combined match score is set to 1; otherwise, it is 0.

F1 Score: The F1 score is used to evaluate the balance between precision and recall in the model's responses. It is particularly useful for binary classification problems, providing a single metric that considers both false positives and false negatives.

Accuracy: This metric measures the proportion of correct predictions (combined matches) out of the total number of interactions. It provides a straightforward evaluation of the model's performance.

3. Efforts to Improve Accuracy

Chunk Overlap: By allowing for overlapping chunks, we ensure that important context is not lost at the boundaries of text chunks. This helps the model maintain coherence and context across chunk boundaries.

Context Length Management: To ensure that the combined length of the user query and retrieved text chunks stays within the model's token limit, we dynamically select text chunks based on their token count. This prevents truncation and loss of information during the model's response generation.

Conversation Memory: The use of ConversationBufferMemory helps retain the context of previous interactions, allowing the model to provide more coherent and contextually relevant answers over a series of questions.

Database Lookup: Before generating a new response, the model checks if a similar question has already been asked and answered. This not only saves computational resources but also ensures consistency in responses for repeated queries.

Threshold Tuning: The threshold for semantic similarity in the combined match metric can be fine-tuned based on empirical evaluation. This ensures a balance between exact matches and near-exact matches, improving the model's practical utility.

Regular Updates: The question-answer database is regularly updated with new interactions, allowing the system to learn from past interactions and improve over time.

Model Configuration: Using advanced language models (e.g., OpenAI's GPT) and fine-tuning their parameters can lead to more accurate and contextually appropriate responses.

Feedback Loop: Implementing a feedback mechanism where users can rate the accuracy of responses can provide valuable data for further model refinement and training.

Summary

By carefully constructing the dataset, selecting appropriate evaluation metrics, and implementing strategies to improve accuracy, we ensure that the question-answering system provides reliable and contextually relevant responses to user queries. The combination of semantic similarity, exact match, combined match, F1 score, and accuracy metrics offers a comprehensive evaluation framework, guiding continuous improvement efforts.