QueryBuddy: User Guide

1. Introduction

QueryBuddy is an advanced question-answering application designed to empower users with the ability to extract insights from PDF documents through an intuitive and interactive interface. By leveraging cutting-edge natural language processing and machine learning techniques, QueryBuddy transforms static PDF content into a dynamic knowledge base that users can query using natural language.

This comprehensive guide provides detailed instructions on how to set up, use, and understand the inner workings of QueryBuddy. Whether you're an end-user looking to make the most of the application or a developer interested in the technical implementation, this document serves as your complete resource for all things QueryBuddy.

2. System Overview

QueryBuddy operates on a sophisticated pipeline that includes document processing, text embedding, vector storage, similarity search, and natural language generation. Here's a high-level overview of the system:

1. Document Ingestion: Users upload PDF documents to the system.
2. Text Extraction: The system extracts text content from the PDF.
3. Text Chunking: Large documents are split into manageable chunks.
4. Embedding Generation: Each chunk is converted into a high-dimensional vector representation.
5. Vector Storage: These embeddings are stored in a vector database for efficient retrieval.
6. Query Processing: User questions are processed and converted to vector representations.
7. Similarity Search: The system finds the most relevant document chunks based on the query.
8. Answer Generation: Using the retrieved context, an AI model generates a natural language answer.

9. Result Presentation: The answer, along with source segments, is presented to the user.

This process allows QueryBuddy to provide accurate and contextually relevant answers to user queries, effectively turning any PDF document into an interactive knowledge base.

3. User Guide

3.1 Uploading a PDF

1. Open the QueryBuddy application in your web browser.
2. Locate the file uploader widget at the top of the page.
3. Click on the "Browse files" button or drag and drop a PDF file into the designated area.
4. Once the file is uploaded, its name will appear below the uploader.

3.2 Processing the Document

1. After uploading the PDF, click the "Process PDF" button.
2. The system will extract text from the PDF, split it into chunks, and create vector embeddings.
3. A progress bar will indicate the processing status.
4. Once complete, a success message will display the number of chunks processed.

3.3 Asking Questions

1. In the text input field labeled "Ask a question about the document:", type your question.
2. Press Enter or click the "Ask Question" button to submit your query.

3.4 Viewing Responses

1. The system will process your question and display the answer in the "Latest Answer" section.
2. To see the document segments used to generate the answer, expand the "View Source Segments" section below the answer.

3.5 Continuing the Conversation

1. After receiving an answer, you can ask another question immediately.
2. If you want to clear the current question and answer, click the "Next Question" button.

3.6 Viewing History

1. The sidebar displays a history of your questions and answers for the current session.
2. Click on any question in the history to expand and view its answer.

3.7 Resetting the Application

1. To clear all data and start fresh, click the "Clear All & Reset" button in the sidebar.
2. This action will remove the processed document, clear the Q&A history, and prepare the system for a new document upload.

4. Technical Implementation

4.1 Architecture Overview

QueryBuddy is built on a modular architecture that combines several powerful technologies:

- Frontend: Streamlit
- PDF Processing: PyPDF2
- Text Embedding: Hugging Face Transformers
- Vector Storage and Retrieval: Pinecone
- Natural Language Processing: Cohere

The application follows a serverless architecture, with Streamlit handling both the frontend and backend logic.

4.2 Key Components

1. PDF Processor:
   - Utilizes PyPDF2 to extract text from uploaded PDF documents.
   - Implements a chunking mechanism to split large texts into manageable segments.
2. Embedding Generator:
   - Uses DistilBERT model from Hugging Face Transformers to create vector representations of text chunks.
3. Vector Store:
   - Leverages Pinecone as a vector database for efficient storage and similarity search.
4. Query Processor:
   - Converts user questions into vector representations for similarity matching.
5. Answer Generator:
   - Utilizes Cohere's language model to generate natural language answers based on retrieved context.

4.3 Data Flow

1. Document Ingestion:
   - PDF uploaded → Text extracted → Text split into chunks
2. Document Processing:
   - Chunks → Embedding Generator → Vector representations
3. Vector Storage:
   - Vector representations → Pinecone Index
4. Query Processing:
   - User question → Embedding Generator → Query vector
5. Similarity Search:
   - Query vector → Pinecone Index → Relevant chunks
6. Answer Generation:
   - Relevant chunks + Original question → Cohere API → Generated answer
7. Result Presentation:
   - Generated answer + Source chunks → User interface

4.4 Error Handling

The application implements comprehensive error handling to manage potential issues:

- File Upload Errors: Checks for correct file type and handles corruption or read errors.
- API Errors: Manages timeouts or failures in Pinecone and Cohere API calls.
- Processing Errors: Handles exceptions during text extraction, embedding generation, and chunk processing.

All errors are caught, logged, and presented to the user with informative messages to guide troubleshooting.

4.5 State Management

QueryBuddy utilises Streamlit's session state to manage application state across user interactions:

- `pdf_processed`: Tracks whether the current PDF has been processed.
- `num_chunks`: Stores the number of chunks processed for the current document.
- `qa_history`: Maintains a list of question-answer pairs for the session.
- `current_pdf_name`: Keeps track of the name of the currently processed PDF.
- `current_question` and `current_answer`: Store the latest Q&A interaction.
- `results`: Holds the retrieved context for the current question.

This state management ensures a seamless user experience and enables features like history viewing and context retention between questions.

5. Performance Considerations

QueryBuddy is designed to handle documents of varying sizes efficiently:

- Chunk-based processing allows for manageable memory usage even with large documents.
- Pinecone's vector index enables fast similarity searches, maintaining quick response times as the document base grows.
- Batch processing of embeddings reduces API call overhead.

However, users should be aware that very large documents (hundreds of pages) may require significant processing time during the initial upload and vectorization phase.

6. Limitations and Known Issues

- PDF Compatibility: Some highly formatted or scanned PDFs may not process correctly.
- Language Support: The current implementation is optimized for English-language documents.
- Context Window: There's a limit to how much context can be used for answer generation, which may affect accuracy for very complex queries.
- Stateless Nature: The application does not maintain state between sessions, so all data is lost upon closing the application.

7. Future Improvements

Potential enhancements for future versions of QueryBuddy include:

- Multi-language support
- Integration with additional document formats (e.g., DOCX, TXT)
- Improved handling of tables and images in PDFs
- User authentication and document persistence
- Fine-tuning capabilities for domain-specific knowledge
- Integration with external knowledge bases