Knowledge Base QA System Documentation

Overview

The Knowledge Base QA System is a web application that allows users to query text documents using natural language questions. The system uses AI to understand questions, search through documents for relevant information, and generate accurate answers based on the document content.

System Architecture

Components

1. **Frontend**: React-based user interface
2. **Backend API**: FastAPI server
3. **Embedding Model**: SentenceTransformer for semantic understanding
4. **Vector Database**: FAISS for efficient similarity search
5. **LLM**: Ollama running Mistral for question answering

Technical Flow

1. Document processing:
   - Text is split into manageable chunks
   - Chunks are embedded using SentenceTransformer
   - Embeddings are indexed in FAISS
2. Question answering:
   - User question is embedded
   - Similar document chunks are retrieved using FAISS
   - Retrieved chunks and question are sent to Mistral LLM
   - Answer is generated and returned to user

Installation Requirements

Backend Requirements

```
fastapi
uvicorn
pydantic
numpy
faiss-cpu (or faiss-gpu)
sentence-transformers
requests
```

Additionally, you need Ollama running locally with the Mistral model.

Frontend Requirements

```
react
react-dom
```

Setup Instructions

Backend Setup

1. Ensure you have Python 3.7+ installed
2. Install required packages: `pip install -r Requirements.txt`
3. Install and run Ollama with Mistral: `ollama run mistral`
4. Create a 'kb' directory or set KB_FOLDER environment variable
5. Start the backend: `python app.py`

Frontend Setup

1. Ensure you have Node.js and npm installed
2. Install dependencies: `npm install`
3. Update API_URL in App.js if backend is not at http://localhost:8000
4. Start the frontend: `npm start`

User Guide

Managing Knowledge Base Files

1. **View Files**: The system displays up to 4 files by default, prioritizing bookmarked files.
2. **Search Files**: Use the search box to find specific files in the knowledge base.
3. **Bookmark Files**: Click the star icon (☆) to bookmark important files for easy access.
4. **Refresh**: Click the "Refresh" button to update the file list.

Asking Questions

1. **Select a File**: Click on a file from the knowledge base to load it.
2. **Ask a Question**: Type your question in the search box and click "Search".
3. **Response Options**: Toggle "Enable real-time streaming response" to see answers appear in real-time.
4. **Start New Search**: Click "Start New Search" to select a different file.

Features

Core Features

- **Text Search**: Find documents based on keywords or phrases.

- **Natural Language Questions**: Ask questions in everyday language.
- **Real-time Streaming**: Watch answers appear as they're generated.
- **Bookmarking**: Save important documents for quick access.
- **Cross-Document Search**: Implemented but not exposed in UI.

UI Features

- **Responsive Design**: Works on desktop and mobile devices.
- **File Management**: Search, sort, and bookmark knowledge base files.
- **Error Handling**: Clear error messages for troubleshooting.

API Documentation

Endpoints

Endpoint	Method	Description
`/kb`	GET	List all files in the knowledge base
`/kb/{filename}`	GET	Get content of a specific file
`/process`	POST	Process a document for QA
`/ask`	POST	Ask a question (non-streaming)
`/ask/stream`	POST	Ask a question (streaming response)
`/clear`	POST	Clear current session data
`/search/kb`	POST	Search across all knowledge base files

Sample API Requests

Process a document

```
POST /process
Content-Type: multipart/form-data

file: [file upload]
save_to_kb: false
```

Ask a question

```
POST /ask
Content-Type: application/json

{
  "question": "What is the main topic of this document?"
}
```

Technical Details

Document Processing

1. Documents are split into chunks of 500 words with 100-word overlap
2. Each chunk is embedded using SentenceTransformer's all-MiniLM-L6-v2 model
3. Embeddings are indexed in a FAISS L2 index for fast similarity search

Question Answering

1. User question is embedded using the same model
2. Top 3 most relevant chunks are retrieved from FAISS
3. Context and question are combined into a prompt
4. Prompt is sent to Ollama's Mistral model
5. Answer is returned to the user

Session Management

The system maintains a session for each processed document, storing:

- The document chunks
- The FAISS index
- The filename
- A unique session ID

Troubleshooting

Common Issues

1. **Backend Connection Errors**
   - Ensure the backend is running
   - Check API_URL in App.js
   - Verify CORS settings if using different domains
2. **Document Processing Errors**
   - Verify file format is supported (.txt)
   - Check file encoding (UTF-8 recommended)
3. **Ollama Issues**
   - Ensure Ollama is running
   - Verify Mistral model is installed
4. **Empty or Incorrect Answers**
   - Check if document contains relevant information
   - Try reformulating your question
   - Ensure the document is properly processed

Future Enhancements

Potential improvements for future versions:

1. Support for more document formats (PDF, DOCX, etc.)
2. Authentication and user management
3. Persistence of sessions and user preferences
4. Advanced search filtering options
5. Document collections and organization features
6. Chat history and conversation context
7. Customizable LLM models and parameters
8. Data visualization for search results

Security Considerations

Current implementation is designed for development and not production use. For production deployment, consider:

1. Implementing authentication
2. Securing API endpoints
3. Validating file uploads
4. Rate limiting requests
5. Implementing proper session management
6. Sanitizing user inputs
7. Using HTTPS for all communications

License

This Knowledge Base QA System is provided as-is for educational purposes. For licensing, please contact Sukhpreet Singh at Sukhpreetsinghdeol@outlook.com

Credits

Developed by Sukhpreet Singh

Documentation last updated: May 4, 2025