**Tech Note of Q&A System Implementation with DeepSeek API**

***Team Members:*** *Kevin Yunpeng Xin(Technical Support), Amy Wang(Process Coordinator), Yushuang Sun(Optimization Advisor), Xiru Li(Model Tester)*

**System Architecture Overview**

This document analysis system combines Streamlit interactive interface with DeepSeek R1 model, implementing a three-stage processing workflow: document parsing - contextual chunking - intelligent response generation. Here is the core implementation:

A screenshot of a computer screen

Description automatically generated

**Document Processing Pipeline**

The system employs a double parsing strategy to handle complex PDF structures. First, PyMuPDF extracts basic page layouts and text flows. Then, pdfplumber focuses on table detection and structured data preservation:

A screen shot of a computer code

Description automatically generated

The parsing phase preserves document hierarchy through regular expression matching for section headers (header\_pattern) and explicit table boundary markers ([TABLE\_START/END]). This structured representation enables subsequent context-aware splitting.

A screen shot of a computer program

Description automatically generated**Adaptive Chunking Mechanism**

The text splitting strategy dynamically adapts to document structure using a prioritized separator list, ensuring logical grouping of related content:

This hierarchical approach enables three-level content organization: 1) Section groups, 2) Table clusters, and 3) Paragraph units. The keep separator parameter ensures structural markers remain visible to downstream processing stages, aiding the language model to understand of content relationships.

**Context Retrieval Workflow**

When handling user queries, TF-IDF vectorization is used. This retrieval method creates a shared vector space that bridges user queries and document content. The min\_df and max\_df parameters filter out overly common or rare terms and make model focus the similarity calculation on meaningful vocabulary. While simple, this approach effectively balances performance and computational cost for moderate document sizes.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated**Model Interaction Design**

The system integrates retrieved context with model instructions through structured prompting:

Key design considerations include:

* Explicit instructions for table data prioritization
* Source attribution requirements
* Progressive response generation via streaming
* Temperature parameter controlling answer strictness (adjustable through UI)

Interface Configuration

The Streamlit frontend provides interactive controls while maintaining processing state: The interface employs MD5 hashing to detect document changes, preventing unnecessary reprocessing. Chat history management retains recent interactions while preventing memory bloat through the adjustable history limit parameter.

A screen shot of a computer code

Description automatically generated

**Implementation Insights**

The current implementation demonstrates several notable characteristics:

1. **Structural Awareness**: By preserving section headers and table markers during chunking, the system provides the language model with implicit hints about content organization, significantly improving answer accuracy on structured queries.
2. **Progressive Enhancement**: The dual PDF parser architecture allows leveraging PyMuPDF's speed for basic extraction while using pdfplumber for targeted table enhancement. This balances performance and accuracy for practical usage.
3. **Configurable Precision**: Through the adjustable chunk size (500-2000 characters) and overlap size (0-300 characters) parameters, users can optimize context delivery based on document characteristics. Technical manuals may benefit from larger chunks with overlaps, while reports might use smaller focused chunks.

The main trade-offs involve balancing real-time responsiveness with processing depth. The adoption of TF-IDF rather than more advanced embeddings reflects this balance, providing acceptable semantic matching while maintaining Streamlit's deployment benefits.