LexAI: Chatbot(RAG) for AI regulation analyzer - Chloe Yan

System Architecture Overview

This implementation presents a production-ready RAG (Retrieval-Augmented Generation) system specialized for AI policy analysis, built with a modular Python backend and React frontend. The architecture demonstrates solid engineering practices with clear separation of concerns across four core components:

- Vector Store Manager: ChromaDB with HuggingFace all-MiniLM-L6-v2 embeddings
- Document Processor: LangChain-based chunking with recursive character splitting
- Chat Engine: OpenAI GPT-3.5-turbo with conversation memory and source attribution
- API Layer: FastAPI with React frontend for document upload and chat interface

Technical Implementation Decisions

The embedding strategy leverages HuggingFace's all-MiniLM-L6-v2 over OpenAI embeddings for cost optimization while maintaining semantic quality, with this 384-dimensional model providing adequate performance for policy document similarity matching. The chunking approach implements recursive character splitting with 1000-character chunks and 200-character overlap, a configuration that balances context preservation with retrieval granularity while preventing information fragmentation across chunk boundaries. The retrieval configuration employs top-k=5 retrieval with relevance score filtering to ensure comprehensive context while maintaining response quality and staying within token limits.

The system employs role-based prompting by establishing the AI as a "Policy Analysis Expert" with specific behavioral constraints including analyzing AI policy documents with expertise and nuance, providing clear well-sourced insights about AI governance, synthesizing information from multiple sources, and always citing specific documents referenced. This specialized persona improves response relevance and maintains focus on the domain-specific use case, representing a sophisticated approach beyond basic zero-shot prompting techniques. The chat engine constructs prompts with systematic integration of system role definition, conversation history covering the last 6 exchanges, retrieved document context with source metadata, and explicit instructions for source citation. This structure ensures conversational continuity while providing sufficient context for informed responses.

The GPT-3.5-turbo configuration utilizes a temperature of 0.7 for balanced creativity and consistency in analytical responses, top-p sampling at 0.9 for nucleus sampling that produces coherent but varied outputs, a maximum token limit of 800 sufficient for detailed analysis while managing costs, and GPT-3.5-turbo as the cost-effective choice for production deployment. The temperature of 0.7 provides analytical rigor while allowing for nuanced interpretation of policy documents, while the 800-token limit ensures comprehensive responses without excessive verbosity, optimizing both user experience and API costs.

Challenges and Limitations

The system faces several performance bottlenecks including CPU-based HuggingFace embeddings that create latency during document ingestion particularly for large documents, context window management that uses only top-5 sources potentially missing relevant information in large knowledge bases, and ChromaDB persistence that could become memory-intensive with substantial document collections. Retrieval quality is constrained by chunk boundary problems where fixed chunking may split semantically related content reducing retrieval accuracy, query-document mismatches where a single embedding model may not capture domain-specific terminology optimally, and ranking limitations where cosine similarity alone may not reflect document relevance for complex policy questions.

Potential Improvements

Architecture evolution encompasses extending document processing to handle multi-modal support for tables, figures, and structured data within policy documents, incorporating knowledge graph integration with entity extraction and relationship mapping for more sophisticated policy analysis, implementing automated evaluation frameworks and custom policy-specific benchmarks for continuous system improvement, and migrating to distributed microservices architecture with container orchestration for production scalability. The codebase demonstrates good practices with proper type hints, documentation, and error handling, though several areas require attention including centralized configuration management for environment variables and system parameters, comprehensive testing infrastructure with unit and integration tests plus performance benchmarks, production monitoring through logging and metrics collection with alerting systems, and enhanced security measures including authentication, input validation, and secure file handling.

Conclusion

The implemented RAG system demonstrates solid engineering fundamentals with appropriate technology choices for the AI policy analysis domain. The combination of cost-effective embeddings, robust document processing, and specialized prompt engineering creates a functional system suitable for production deployment. Key strengths include comprehensive error handling, efficient vector storage, and domain-specific optimization. Primary improvement opportunities lie in advanced retrieval techniques and systematic evaluation frameworks to measure and enhance system performance over time. The modular architecture facilitates future enhancements while maintaining system reliability and cost efficiency—critical factors for sustainable AI application deployment.