RAG System Technical Report

1. Executive Summary

This document describes the RAG (Retrieval-Augmented Generation) system architecture and implementation details. The system is designed to enhance LLM responses by retrieving relevant information from a vector database.

Key Features:
- Multi-format document parsing (PDF, DOCX, XLSX, TXT, JSON)
- Semantic chunking for optimal context retrieval
- Hybrid search with BM25 and dense vectors
- Cross-encoder reranking for improved accuracy

2. System Architecture

The RAG pipeline consists of three main components:

2.1 Document Preprocessing
Documents are first parsed to extract text content. The text is then normalized and split into semantic chunks using embedding-based similarity analysis.

2.2 Vector Storage
Chunks are embedded using OpenAI's text-embedding-3-small model and stored in Weaviate vector database with metadata.

2.3 Retrieval and Generation
User queries are processed through a router that determines whether to use vector search or direct LLM response. Retrieved chunks are reranked and used as context for answer generation.

3. Performance Metrics

The system has been evaluated on the following metrics:

Metric	Value	Target
Precision@5	0.85	0.80
Recall@10	0.78	0.75
Avg Response Time	2.1s	3.0s
Throughput	150/min	100/min

4. Conclusion

The RAG system successfully meets all performance targets and is ready for production deployment. The semantic chunking approach has proven effective in maintaining context coherence, and the hybrid search strategy provides robust retrieval across diverse query types.

5. References

- Lewis et al., 'Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks'
- LangChain Documentation: https://python.langchain.com
- Weaviate Documentation: https://weaviate.io/developers/weaviate