DEVELOPMENT OF A RETRIEVAL AUGMENTED GENERATION SYSTEM

Machine Learning for Context Analytics Capstone

MSc in Business Analytics Part Time 25-26

G. Syrokosta - p2822434
geo.syrokostas@aueb.gr

Contents

Introduction

The objective of this project was to develop and evaluate a proof-of-concept Retrieval-Augmented Generation (RAG) system capable of searching through a consulting firm's historical deliverables and utilizing documented methodologies and real-world examples to generate contextually relevant responses to user queries. RAG systems represent a hybrid approach combining information retrieval with large language model generation capabilities (Lewis et al., 2020).

This technical report documents the business requirements, system architecture, implementation challenges, and lessons learned during the development process. The project faced significant resource constraints: development was undertaken by a single individual with limited programming experience, operating under strict time limitations. These constraints necessitated pragmatic architectural decisions and heavy reliance on existing frameworks and pre-trained models.

The project was conducted within a compressed timeframe by a single-person development team with limited technical expertise, preventing the implementation of several originally envisioned functionalities. The research conducted provides a foundation for future implementation, with this report serving as documentation of initial progress and a compilation of lessons learned for subsequent development phases.

This document should be interpreted as an initial progress report on an ambitious project constrained by its timeline, providing valuable insights for future implementation efforts.

Our project

The project entailed the development of a proof-of-concept RAG system for a consulting firm comprising approximately 250 employees with operations spanning multiple countries. The firm provides management, human resources, information technology, strategy, finance, engineering, and public sector consulting services.

The system, designated "Pallada" (referencing Pallas Athena, the goddess of wisdom, while avoiding copyright concerns), consists of several integrated components: a vector database for storing document embeddings, an embedding model for vectorization, a data ingestion pipeline capable of processing diverse document formats, a web-based interface for query submission and result retrieval, and a Large Language Model (LLM) for generating responses based on retrieved contextual chunks.

Regarding pre-existing infrastructure, the organization's technical foundation was minimal. A company portal currently under development could potentially host the frontend interface, addressing access control and security requirements. Beyond this, the entire system architecture required development from scratch, with no existing document management systems, standardized formats, or metadata schemas in place.

Our Vision & Goals

Pallada's Vision

The primary vision of the Pallada project is to **digitize and operationalize institutional memory** within the organization.

Currently, knowledge retrieval for new engagements relies on senior personnel recalling relevant past projects from memory. This approach presents multiple systemic limitations:

- **Decentralization**: Relevant engagement information may reside in different departments, creating information silos that impede knowledge transfer
- **Cognitive limitations**: Human recall is inherently limited and subject to availability bias, preventing comprehensive retrieval of all relevant prior work
- **Knowledge attrition**: Personnel retirement and turnover result in permanent organizational knowledge loss

The envisioned system aims to centralize all deliverables, enable comprehensive recall of relevant materials, and provide sustainable knowledge management for future operations.

First sprint goals

The initial sprint established modest, achievable objectives: implement a robust PDF parsing pipeline, evaluate various embedding models and chunking strategies, test different retrieval methodologies, develop a RESTful API for frontend-backend communication, and establish the framework for LLM integration upon achieving satisfactory retrieval performance.

While these objectives appear straightforward, they represented significant challenges given the development team's limited technical experience. The team possessed approximately 200 lines of aggregate coding experience at project initiation, making even basic implementations technically ambitious.

Methodology

The project employed an iterative development methodology alternating between research, expert consultation, and implementation phases.

Initial research focused on understanding RAG system architectures in their fundamental form. A high-level architecture was designed without specific technology commitments. This conceptual design was then reviewed with the department's Director for business alignment and with the Head of Software Development Consulting for technical feasibility assessment. The first consultation session established the ideal architecture, while the second refined it to an implementable scope, providing guidance on appropriate frameworks and services. Detailed meeting minutes are included in the appendix.

Following consultation, core functionalities were implemented, with various technical challenges identified during the development process.

Subsequent iterations involved additional research on potential solutions, consultation with subject matter experts regarding proposed approaches and their trade-offs, followed by implementation of selected solutions.

The project timeline accommodated two complete cycles of research-consultation-implementation.

Research phases consistently required the most significant time investment, consuming approximately 70-80% of allocated project hours.

Regarding development tools, AI-powered code generation tools were employed for debugging assistance and code optimization. A critical constraint was maintained that all generated code must remain comprehensible to the development team. While tools such as Cursor or Claude Code could potentially produce more efficient and scalable implementations, such approaches risked creating unmaintainable code beyond the team's technical capabilities

Data Collection

The available data presented the project's most significant challenge.

The organization maintains minimal documentation of historical projects, with existing materials being sparse and inconsistently formatted. No structured database of completed projects exists beyond a SharePoint-based file management system. Only basic file metadata is available for processing. Document formatting follows no consistent cross-project standards, though the majority of deliverables exist in PDF format.

Dataset Overview

From approximately 6,000 available deliverables, a subset was selected based on specific criteria:

- **Security classification**: Highly classified documents were excluded from the initial dataset to prevent potential data exposure
- **Domain familiarity**: Documents were selected where team members had participated in their creation, enabling accurate assessment of retrieval quality and relevance
- **Format diversity**: The selection included various deliverable types ranging from technical documentation to presentations, executive summaries, and project management reports
- **Content characteristics**: The dataset encompassed both information-dense technical documents and information-sparse administrative reports. Additionally, documents varied in their modality, with some being primarily

textual while others relied heavily on visual elements (e.g., Business Process Modeling notation, engineering diagrams)
- **Language distribution**: The firm's national and international operations necessitated testing with both Greek and English documents to evaluate multilingual processing capabilities

Based on these criteria, 19 documents were selected for the Phase 0 dataset. The computational constraints limited dataset size, as ingestion and embedding required approximately 3-4 hours for this corpus.

The Phase 0 dataset composition:

- 8 knowledge-intensive text documents
- 2 companion slide presentations containing condensed versions of text documents
- 5 knowledge-rich standalone presentations
- 4 knowledge-sparse progress reports

The language distribution comprised 16 Greek documents and 3 English documents.

This dataset selection enabled evaluation of critical system capabilities: the retriever's ability to discriminate between information-rich and information-sparse content without manual intervention, the pipeline's capacity to process presentation formats without additional preprocessing, the embedding model's effectiveness with charts, diagrams, and business process flows, and the system's multilingual document handling capabilities.

Data Processing/Annotation/Normalization

The data processing pipeline was designed for maximum automation. The objective was to enable ingestion of the entire SharePoint repository without manual preprocessing or annotation. The current pipeline implementation supports configurable parameters including chunk size, chunk overlap percentage, and embedding model selection. The system recursively traverses directory structures, identifies PDF files, and processes them automatically.

The processing workflow begins with intelligent text splitting using a token-aware splitter that respects document structure, creating breaks at non-textual elements (tables, images) while maintaining awareness of maximum chunk size constraints. For Optical Character Recognition (OCR) capabilities, essential for processing scanned documents and embedded images, Tesseract (Smith, 2007) was implemented with both English and Greek language models. The text normalization process removes special characters and formatting artifacts before converting chunks to plain text. These normalized text strings are then processed through the embedding model to generate vector representations stored in the database alongside document metadata (filepath, author, modification timestamp) and chunk-specific metadata (index, page number). The system processes documents sequentially, embedding

chunks in batches of 50 for memory efficiency. Production deployment would require parallelization across multiple compute nodes for scalability.

For embedding models supporting larger context windows (specifically Jina-v3 with 8192 token capacity; Günther et al., 2023), an alternative page-based chunking strategy was implemented, processing entire pages as single chunks while maintaining maximum size constraints.

Algorithms, NLP architectures/systems

The implementation leveraged the LangChain framework (Harrison, 2023) for Python, providing model-agnostic abstractions for embedding, retrieval, and generation components. This architectural decision enabled experimentation with multiple models without requiring fundamental system modifications.

The implemented architecture follows the standard RAG pipeline:

- Document ingestion and preprocessing
- Chunk generation and embedding
- Vector storage and indexing
- Frontend query interface
- API-mediated frontend-backend communication
- Query embedding and processing
- Similarity search execution
- Contextual chunk retrieval
- Prompt template construction with retrieved context
- LLM-based response generation
- Response delivery via API

For the generation component, OpenAI's GPT-3.5-turbo model was utilized (Brown et al., 2020), though initial testing employed GPT-2 (Radford et al., 2019) for cost optimization during retrieval system development.

Experiments – Setup, Configuration

Technology selection involved extensive experimentation facilitated by LangChain's abstraction layer, enabling component substitution with minimal code modifications. The following table summarizes evaluated technologies, with the final selections highlighted:

Function	Tech	Decision informing factors
Embedding	multilingual-RoBERTa (Liu et al., 2019)	Legacy multilingual model; MPNet-v2 demonstrates superior performance across all metrics
	MPNet-v2 (Song et al., 2020)	Combines BERT and XLNet architectures for improved performance; optimal for simple embedding requiring minimal preprocessing; encountered challenges with slide deck formats
	text-embedding-3-small (OpenAI)	Doubles the context window of MPNet-v2; marginal performance improvements did not justify usage costs
	Jina-v3 (Günther et al., 2023)	Supports 8192-token chunks; resolved slide deck processing issues through page-level embedding; retrieval performance affected by larger vector dimensions; superseded by Jina-v4 (see Future Work)
Vector Store	FAISS (Johnson et al., 2019)	Superior LangChain integration; native MMR with relevance scores; less intuitive API syntax
	ChromaDB (Trychkin, 2022)	Intuitive syntax appropriate for proof-of-concept; requires migration to production-grade database supporting complex schemas and hybrid retrieval (PostgreSQL with pgvector or Neo4j under consideration)
API	**FastAPI (Ramírez, 2018)**	Superior feature set for concurrent user support compared to Flask; steeper learning curve manageable given project scope
Generation Modelf	GPT-2 (Radford et al., 2019)	Lightweight testing placeholder during retrieval development
	Self-Hosted Qwen3-4b	Self-hosting attempted via Ollama; computational constraints resulted in prohibitive inference times
	GPT-3.5-turbo (OpenAI)	Acceptable performance for proof-of-concept; reasonable cost; adequate output quality for specific queries; retrieval improvements prioritized over model selection
Front-End	**Plain HTML**	Minimal viable interface; migration to modern framework required for production deployment

Results & Quantitative Analysis

Quantitative evaluation of RAG systems presents inherent challenges, particularly given the architecture's composition of multiple pre-trained models. Rather than reporting standard benchmarks for individual components, evaluation focused on system-level performance metrics.

The primary quantitative metric evaluated was cosine similarity relevance scores. Due to LangChain's limited support for Maximum Marginal Relevance (MMR) with scoring (available only for FAISS), comprehensive scoring required workarounds for ChromaDB. While technically feasible through manual score calculation, this approach was deemed inefficient pending proper implementation in the production database selection.

For MPNet-v2 embeddings, relevance scores ranged from 60% to 85%, with empirical analysis establishing 71% as the threshold for meaningful relevance. Jina-v3 demonstrated approximately 10% lower relevance scores overall, attributable to larger and more heterogeneous chunk content. The variation in these metrics across different document collections prevented establishment of hard cutoff thresholds, as such values would likely require adjustment as the document corpus expands.

Qualitative & Error Analysis

Qualitative assessment leveraged the team's familiarity with document content to evaluate precision and recall, though systematic metrics were not formally recorded. MPNet-v2 demonstrated superior precision with fewer irrelevant retrievals but exhibited inferior recall, frequently returning multiple chunks from the same document and exhausting available retrieval slots. Implementation of Maximum Marginal Relevance (MMR) search partially mitigated this issue. Increasing the number of retrieved sources yielded inconsistent results, often introducing additional noise rather than improving coverage.

Corpus expansion is expected to improve retrieval quality by providing more relevant content, though this will necessitate continuous re-evaluation of the optimal number of retrieved sources as a critical hyperparameter. As the database scales beyond individual familiarity with its contents, establishing systematic quantitative evaluation methodologies will become essential.

The system demonstrated superior performance with detailed, structured queries employing domain-specific terminology compared to keyword-based searches. Retrieval quality remained insufficient for meaningful generation, with persistent issues including irrelevant source selection, high redundancy (attributable to document structure and the absence of vector deduplication during initial setup), and poor-quality vectorization of presentation formats (largely resolved with Jina-v3). These retrieval deficiencies resulted in generation models receiving primarily irrelevant, corrupted, or duplicated context, severely impacting output quality.

Discussion, Comments/Notes and Future Work

The project encountered fundamental limitations inherent to text-based vector retrieval when applied to complex, multimodal PDF documents, compounded by resource constraints.

Several promising directions for advancement have been identified:

- **Database Architecture Enhancement**: Implementation of SQL or graph databases with comprehensive metadata schemas for deliverables
- **Automated Classification**: Deployment of zero-shot classifiers to generate chunk-level metadata annotations
- **Document Categorization**: Utilization of chunk classifications to enable document-level categorization for self-querying capabilities
- **Relationship Modeling**: Establishment of connections between deliverables, teams, and personnel expertise for complex schema development
- **Hybrid Retrieval**: Migration to systems combining dense and sparse retrieval methods leveraging structured metadata
- **Self-Querying Implementation**: Development of query routing based on metadata fields beyond embeddings to reduce irrelevant retrievals
- **Multimodal Embeddings**: Adoption of models such as Jina-v4 (Günther et al., 2024) or BAAI/bge-m3 (Chen et al., 2024) with appropriate vector databases (e.g., Vespa)
- **Conversational Capabilities**: Integration of LangGraph functionality for multi-turn dialogue support
- **Advanced Prompt Engineering**: Development of role-specific prompting strategies based on query categorization
- **Platform Migration**: Code refactoring to Java with Spring Boot framework for consistency with existing enterprise infrastructure (though this remains a low-priority consideration)

Members/Roles

Project Lead and Sole Developer: Galatea Syrokosta

Time Plan

- **June:** Initial research and literature review
- **July:** Continued research and project scoping
- **August:** Development and implementation
- **September:** Documentation and report composition

Bibliography

- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901.
- Chen, Z., Deng, X., & Zhang, Y. (2024). BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings. *arXiv preprint arXiv:2402.03216*.
- Günther, M., Mohr, F., & Wang, B. (2023). Jina Embeddings v3: A Novel Set of High-Performance Sentence Embedding Models. *arXiv preprint arXiv:2307.11224*.
- Günther, M., Mohr, F., & Wang, B. (2024). Jina Embeddings v4: Multi-Modal Embeddings for Document Understanding. *Technical Report, Jina AI*.
- Harrison, C. (2023). LangChain: Building applications with LLMs through composability. *GitHub repository*. https://github.com/langchain-ai/langchain
- Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535-547.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9.
- Ramírez, S. (2018). FastAPI: Modern, fast (high-performance), web framework for building APIs with Python 3.6+. *GitHub repository*. https://github.com/tiangolo/fastapi
- Smith, R. (2007). An overview of the Tesseract OCR engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2, 629-633.
- Song, K., Tan, X., Qin, T., Lu, J., & Liu, T. Y. (2020). MPNet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33, 16857-16867.
- Trychkin, A. (2022). ChromaDB: The AI-native open-source embedding database. *GitHub repository*. https://github.com/chroma-core/chroma