Statement of Work: (Secure ‘doc’ Management And Retrieval Technology) S.M.A.R.T.

Authors

Gurpreet K Hundal : gurpreet@sklose.com Tiffany Valdecantos: tiv001@g.harvard.edu
Hellen Momoh: hem299@g.harvard.edu Spiro Habasch: sph083@g.harvard.edu

Background and Motivation

Organizations struggle to manage and extract value from expanding digital document repositories while maintaining strict access control. Traditional search systems rely on keyword-based searches, which lack contextual understanding and result in inefficiencies, often leading to disruptions in internal communication. Recent advancements in Natural Language Processing (NLP) have made it possible to leverage large and unstructured data for intelligent retrieval and summarization. However, organizations are reluctant to use these technologies due to privacy and security concerns. This project aims to understand these concerns and implement a secure system that can be utilized for retrieval and summarization of sensitive documents.

Scope and Objectives

S.M.A.R.T. will be a secure document management and retrieval system that integrates Large Language Models (LLMs) with enterprise-grade security and attribution mechanisms. By leveraging advanced AI techniques, the system will provide context-aware document retrieval, secure access management, and attribution tracking. This will result in reduced response time, improved knowledge discovery, and enhanced organizational efficiency, offering an immediate return on investment (ROI) by streamlining internal operations and minimizing redundant inquiries. Below are the main components:

1. ***Data Sources***: Class notes, quizzes, and supplementary reading material from (>10) classes of Harvard will be utilized for this project. This will mimic the uniqueness of corporate data while providing a fair overlap. Additionally, quizzes will be the primary data source to test access and security.
2. ***Secure Document Storage***: Securely collect, store, and organize diverse class notes, quizzes, and organizational documents.
3. ***Intelligent Search & Ranking:*** Generate vector embeddings(encoder) for documents and queries.Implement a ranking model to improve document retrieval accuracy.
4. ***Scalable Backend Architecture***: Develop a robust and scalable backend to handle multiple queries efficiently. Fine-tune LLMs(decode) for structured response generation.
5. ***Fine-Tuning Techniques***: Implement Reinforcement Learning from Human Feedback (RHLF) and Low-Rank Adaptation (LoRA).
6. ***User-Friendly Frontend & Chatbot:*** Design an intuitive, secure, and responsive frontend with a chatbot to facilitate user interactions.

Application Design

Below is the preliminary application design, subject to revisement as we move through the milestones.

1. ***Frontend***: Users access the system via a web based frontend built with React.js/Next.js hosted on Vercel/Netlify. The user logs in using an identity provider like Google OAuth/Firebase Auth0, which provides a secure access token. The frontend sends the user query + authentication token to the backend via API Gateway.

2. ***API Gateway:*** The API gateway (Apigee / GCP API Gateway) validates the OAuth token. If valid, it forwards the request to the back-end for processing and document retrieval.

3. ***Redis Cache Check (NTH[1])***: Before running an expensive database query, the system checks Redis for cached results. If the query is already stored in Redis, it immediately returns the cached response to the user.

4. ***User Access Check and document retrieval***: The system verifies the documents a user can access and any applicable restrictions with metadata. Once user access is verified, the user query is converted into a vector embedding (SBERT all-MiniLM-L6_v2 or BAAI + BM25). The system queries Postgres(pgvector) to fetch vector embeddings of the document chunks that the user is allowed to access to find the most relevant embeddings based on the query vector (hybrid BM25 for keyword based matching). The retrieved document IDs and metadata are returned for further ranking. Using the document IDs retrieved from pgvector, the system fetches the corresponding links from Postgres. This ensures that the system does not need access to GCS where the documents are stored.

5. ***ReRanking***: Each query pair is re-ranked using a cross encoder (ms-marco-MiniLM-L-6-v2) to improve accuracy. The re-ranking model assigns new ranks to prioritize the best matches ensuring the most contextually relevant document chunks (not just the closest vector matches) are passed to the LLM.

6. ***LLM***: The top ranked document chunks (with storage links) are passed as context to the LLM (LlaMa2 or PaLM2). The LLM generates a structured response with an AI generated answer, top document link and relevant metadata (title, author, publication date, etc.) The final response is stored in Redis so that repeated queries don't hit the database unnecessarily(NTH). The answer from LLM is returned to the Frontend. The user sees the generated answer and top 3 document links to the documents stored in GCS.

7. ***Audit***: The audit system (Postgres audit table) logs interaction at two key points: document retrieval (logs who searched for what and which documents were retrieved) and LLM (logs which documents were passed to the LLM, query, text, and final response). This provides security, compliance, and debugging capabilities.

8. ***Attribution (NTH)***: The LLM analyzes its own response and estimates which parts came from retrieved document chunks vs. general knowledge. After generating the response, the LLM annotates each sentence/phrase with a confidence score for context relevance.

9. ***Reinforcement Learning from Human Feedback (NTH)***: Users can rate responses (thumbs up or down) to provide corrections. This feedback is logged and used to fine-tune the LLM over time. This can assist in adjusting the weight of retrieval ranking and improving similarity search accuracy.

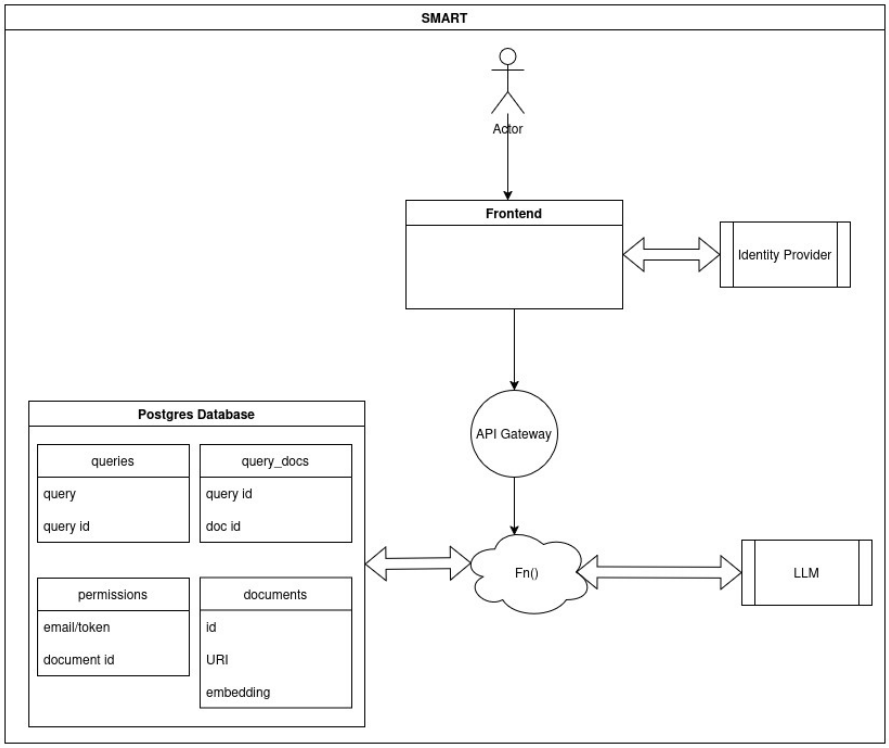
Milestones:

1. Data collection and preprocessing: Feb 27
2. MS2 - Set up a RAG workflow, including data chunking and integration with a vector database: Mar 13
3. MS3 - Revisit and update the app design for Midterm Presentation: Mar 27
4. Backend & Frontend Implementation and Chatbot Integration: Apr 13
5. MS4 - Development and Deployment: Apr 17

[1]NTH: Nice to have

6. MS5 - Final testing and deployment: May 10

Appendix



References

- *L. Marujo, J. Portêlo, W. Ling, D. Martins de Matos, J. P. Neto, A. Gershman, J. Carbonell, I. Trancoso, and B. Raj*, "Privacy-Preserving Multi-Document Summarization," arXiv:1508.01420, 2015.
- *S. Wang, T. Zhu, B. Liu, M. Ding, X. Guo, D. Ye, W. Zhou, and P. S. Yu*, "Unique Security and Privacy Threats of Large Language Models," arXiv:2406.07973, 2024.
- *S. Sousa and R. Kern*, "How to Keep Text Private? A Systematic Review of Deep Learning Approaches to Text Privacy," arXiv:2205.10095, 2022.