Blake Green

Professor Xihao Xie

CS 5393.004

29 April 2025

Exam 3: Comprehensive Report

Introduction

Retrieval-Augmented Generation, commonly known as RAG, represents a transformative approach in which information retrieval engines and large language models collaborate to produce contextually gathered responses. By embedding knowledge sources into a vector space and leveraging nearest-neighbor search to fetch relevant content, RAG systems combine the precision of information retrieval with the generative feature of modern LLMs. The program described in this report seeks to evaluate how different embedding methodologies influence the retrieval quality and answer accuracy when applied to a corpus of aviation accident reports. Ten accident investigations drawn from National Transportation Safety Board (NTSB) documents were processed, indexed, and queried through the RAG pipeline in order to compare three open source embedding models. In this report you will find a description of the system's architecture, detailed results from both single-document and multi-document question sets, and comparative analysis of each embedding model's performance. Subsequent sections unpack the strengths and weaknesses of each approach, dive into general lessons learned about embedding-based retrieval, and reflect upon the real-world implications for safety-critical domains. The report concludes with a discussion of the technical and logistical hurdles encountered during implementation and the strategies used to resolve them. Through this exploration, the report aims to demonstrate not

only which model achieved the best retrieval outcomes in this specific context, but also how RAG architectures can be shaped by embedding choices and operational constraints.

Program Overview and Embedding Models

The program features a pipeline that begins with document text retrieval and results in an LLM-driven question-answering loop. It loads environment variables from a .env file to securely manage API keys and file paths. The directory specified by PDF_DIR is scanned for PDF files, each of which is read page by page using a PDF reader utility. Extracted text from each document is concatenated into a single string, and the collection is truncated to ten reports to balance coverage with computational cababilities. Once the raw texts are loaded, the program computes an “optimal” chunk size based on the average document length: specifically, 25 percent of the average length defines the chunk size, with a ten percent overlap to preserve contextual continuity between adjacent segments.

Embedding generation uses three distinct models from the Sentence Transformers framework. The first model, known as all-MiniLM-L6-v2, produces 384-dimensional embeddings via a distilled Sentence-BERT architecture optimized for speed and low memory consumption. The second model, hkunlp/instructor-xl, is an instruction-tuned transformer yielding 768-dimensional vectors that can incorporate task-specific prompts into the embedding process. The third, all-mpnet-base-v2, also produces 768-dimensional representations and is trained using an objective combining masked and permuted language modeling. These three embeddings were selected for their free availability, contrasting dimensionality, and reported performance diversity. All three embedding functions convert lists of text chunks directly into NumPy arrays. After embedding, the program initializes a Qdrant client, creates three

collections—one for each embedding type—with cosine similarity as the distance measure, and uploads every vectorized chunk along with its source text.

For querying, the system wraps each Qdrant collection into a LangChain vector store, specifying the appropriate embedding model to encode incoming questions. A lightweight RetrievalQA chain of type “stuff” concatenates the top-k retrieved chunks and feeds them to an LLM. Rather than relying on a large language model that requires cost per token, the program integrates ChatMistralAI, a small Mistral-based chat model, configured at zero temperature with built-in retry logic. Two distinct question banks are defined. The MultiRAG questions require reasoning across the entire corpus to identify the most dangerous accidents, count serious injuries, and so on. The SingleRAG questions focus on individual incident IDs such as DCA22LA182, querying details about cause, hiring dates, and damaged parts. The execution loops iterate over each embedding collection, run every question in sequence, and deliberately pause for thirty seconds between queries to restrict token rate limits and avoid overloading the LLM.

Detailed Results (SingleRAG and MultiRAG)

Multi-document retrieval questions produced varied results across the three embedding methods. When asked to identify the most dangerous accident—expected to be the midair collision labeled CEN23MA034 with six fatalities—both the MiniLM model and the instruction-tuned embedding correctly described the Dallas airshow collision and fatality count, although neither cited the report code explicitly. In contrast, MPNet misinterpreted “most dangerous” as the birdstrike incident CEN24LA035, focusing on structural damage rather than human loss. In the query about the least populated accident, MiniLM slightly miscounted two people instead of

one, while MPNet correctly reported one occupant. The instructor-xl model returned an entirely incorrect incident, demonstrating a tendency toward false positives on the number of people on board. When counting serious non-fatal injuries, the instructor and MPNet models accurately identified three accidents with significant injuries, but MiniLM under-counted by returning only two. Finally, identifying the oldest pilot failed across all models: none retrieved the expected sixty-seven year-old aviator in CEN23MA034, instead naming pilots aged sixty and sixty-two from unrelated incidents. An aggregate analysis revealed that only MiniLM and instructor-xl correctly handled the fatality-based definition of "most dangerous," while instructor-xl and MPNet excelled at quantifying injuries.

Single-document retrieval exhibited a more uniform performance overall. The probable cause of incident DCA22LA182—a lateral runway excursion—was correctly retrieved by all three embeddings, matching the official NTSB phrasing. However, inquiring about the hiring date and airline for DCA23LA384 exposed a consistent hallucination: each model invented "June 10" dates and attributed United Airlines as the employer, deviating from the true April 2001 hiring record. On the question regarding the airshow name for the fatal incident, all three correctly answered "Wings Over Dallas," with the instructor-xl model adding precise location details at Dallas Executive Airport. Finally, when asked which object was struck in accident DCA21LA137 and which wing sustained damage, each model correctly returned "light pole" and "left wing," demonstrating high reliability on defined collision details. The consistency across embedding techniques in answering cause, airshow, and object-wing queries suggests that certain incident-specific facts are deeply encoded in the vector space, whereas contextual employment details proved otherwise.

Analysis of Model Strengths and Weaknesses

The all-MiniLM-L6-v2 embeddings consistently delivered high retrieval speed with minimal memory footprint, an advantage for large collections or constrained environments. In multiple-document scenarios, MiniLM correctly associated fatalities with “most dangerous,” showcasing solid semantic understanding of high-impact events. Its primary weakness emerged in precise numeric gathering: it miscounted passenger numbers in the least-populated accident and under-counted serious injuries, indicating that the condensed 384-dimensional space may sometimes smooth over fine-grained distinctions.

Instructor-xl embeddings offered a balanced blend of contextual sensitivity and response quality. The instruction-tuned nature utilized the model to interpret queries with nuanced prompts, resulting in the correct count of serious injuries and enhancing single-document responses with additional geographic detail. Nevertheless, this occasionally led to overfitting on surface patterns, as seen in the selection of a ten-person accident for the “least amount of people onboard” question. This suggests that high-dimensional, instruction-conditioned embeddings may introduce scattered correlations when faced with few or unclear examples.

The all-mpnet-base-v2 embeddings stood out for their balanced performance on injury counts and accurate retrieval of single-document collision details. MPNet’s architecture, trained on both masked and permuted language objectives, appears good at capturing structural relationships in longer texts, which likely contributed to its success on injury and object-wing questions. However, MPNet’s greatest shortcoming was its misinterpretation of “most dangerous,” where it correlated structural damage with danger rather than human cost. This misalignment shows that even high-capacity embeddings can diverge from intuitive human definitions when the objective function does not explicitly encode certain semantic hierarchies.

Taken together, the comparison shows that embedding dimensionality and training objectives shape retrieval behavior in meaningful ways. Lower-dimensional models may sacrifice granularity for speed, while high-dimensional, instruction-tuned embeddings may introduce random associations without careful prompt engineering.

Insights Gained about RAG and Embedding Models

This exam showcased several new thoughts about retrieval-augmented generation and the role of embedding design. First, embedding choice influences not only which text segments are retrieved but also how retrieved context guides the generative model's answer. When embeddings misrank relevant chunks, even a highly capable LLM may produce plausible but incorrect responses. Second, question phrasing emerged as a critical factor: terms like "most dangerous" require shared definitions between human and the embedding's perception of semantic similarity. Without explicit defenition, embeddings may default to damage metrics rather than fatality counts, revealing a gap between human intentions and vector semantics. Third, factual consistency across retrieval steps is not guaranteed; single-document queries about cause or collision details were highly accurate due to literal matches in the text, while questions demanding numeric aggregation or historical context (such as hiring dates) consistently led to hallucinations. This highlights that RAG systems remain vulnerable to retrieval gaps and LLM imagination when source data is sparse or unevenly represented. Fourth, the process of computing optimal chunk sizes based on document statistics proved valuable: dynamically sizing chunks prevented the predominance of either overly broad summaries or syntactically fragmented excerpts. Last, integrating an open-source LLM with deterministic settings allowed

for comparisons across embedding methods, a crucial requirement for continued evaluation even at the cost of added latency and throughput constraints.

Practical Implications

From a deployment perspective, the trade-offs observed here inform model selection for production RAG systems. When rapid retrieval over very large documents takes place, a lightweight model such as all-MiniLM-L6-v2 suffices, trading off some numeric precision for speed and lower compute cost. In applications demanding fine-grained semantic matching—legal precedents or medical literature, for instance, an instruction-tuned, higher-dimensional embedding like hkunlp/instructor-xl may result in richer context, provided that prompt engineering aligns the retrieval objective with domain-specific definitions. MPNet is a middle path, offering robust structural understanding that excels at large text retrieval but requires careful consideration around subjective measures of importance, like the perception of "damage". In aviation safety reporting, where both numeric counts and written context are critical, a hybrid approach or ensemble of embeddings could mitigate individual model biases and inaccuracies. Furthermore, the reliance on an open-source LLM for answer generation showcases the easiness of cost-effective RAG solutions. Organizations seeking to incorporate such systems into safety-critical workflows must balance compute budgets, latency constraints, and the tolerance for occasional hallucinations, potentially augmenting RAG chains with verification layers or human-in-the-loop review for high-stakes decisions.

Challenges and How I Overcame Them

The progression of the program was marked by numerous technical obstacles. Establishing a stable connection to the Qdrant cloud cluster required repeated adjustments to authentication parameters and network settings. Frequent timeouts during collection creation and bulk uploads were mitigated by disabling gRPC fallbacks and inserting brief retries into the upload loop. Selecting an appropriate LLM posed another major challenge. Initial experiments with the OpenAI API yielded high-quality answers but proved cost-prohibitive for extensive batch testing. Transitioning to Ollama introduced on-premise inference capabilities but encountered compatibility issues with tokenization limits. Groq's GPU-accelerated LLM promised high throughput but imposed a 4 K token ceiling that truncated long-form retrieval contexts. Ultimately, Mistral-small provided a middle ground of sufficient context window and free pricing. Integrating the Mistral API involved adapting the LangChain ChatMistralAI wrapper to support zero-temperature settings and retry logic to handle transient server errors. Through each iteration, systematic logging of request-response cycles and proactive rate limiting ensured that no single component became a performance bottleneck.

Conclusion

This exam demonstrated that embedding selection pushes a huge impact on the effectiveness of Retrieval-Augmented Generation. While all three models achieved high accuracy for concrete, single-document queries, their performance diverged on multi-document aggregation tasks, reflecting inherent biases in dimensionality and training objectives. Instructor-xl and MPNet excelled at numeric injury counts, whereas MiniLM offered unmatched speed and correctly interpreted fatality-based danger. Practical deployment of RAG systems must therefore

weigh these trade-offs against operational constraints. The technical hurdles encountered—from Qdrant connectivity to LLM integration—highlight the complex interplay between retrieval infrastructure and generative models. Yet, by methodically addressing each challenge, I established a robust evaluation framework that can guide future efforts in aviation safety analysis and beyond. The insights gained here pave the way for more reliable, contextually aware RAG applications in domains where accurate information access is not just convenient but essential.