Retrieval-Augmented Generation for AASLD Liver Disease Guidelines

Abhigna Nimmagadda
UFID: 31864878

Rohith Kumar Ballem
UFID: 30969136

Ashfaq Ahmed Mohammed
UFID: 86835927

Harsha Vardhan Reddy Palagiri
UFID: 72699604

Abstract

Large Language Models (LLMs) offer significant potential for clinical decision support but are hindered by hallucinations, static training data, and a lack of direct grounding in authoritative medical protocols. In the domain of hepatology, adherence to constantly evolving guidelines—such as those provided by the American Association for the Study of Liver Diseases (AASLD)—is critical for patient outcomes. We present a domain-specific Retrieval-Augmented Generation (RAG) pipeline designed to answer clinical questions using a dynamically updated corpus of AASLD guidelines. Our system implements a robust scraping and structuring workflow to transform unstructured guideline documents into a semantically indexed knowledge base. We evaluate the performance of three open-weight models (Llama-3.2, Qwen, and Phi-2) using both MultiQuery (MQ) and Re-ranker (RR) retrieval strategies. Our experiments demonstrate that cross-encoder re-ranking significantly improves response quality, consistently outperforming query expansion in both retrieval ranking and answer faithfulness. The proposed system offers a scalable framework for providing trustworthy, citable, and guideline-compliant clinical recommendations.

1 Introduction

Liver disease represents a major global health challenge, contributing to approximately two million deaths annually. Effective management of conditions such as viral hepatitis, cirrhosis, and hepatocellular carcinoma relies heavily on adherence to evidence-based protocols. The American Association for the Study of Liver Diseases (AASLD) provides high-quality clinical practice guidelines that define thresholds for diagnosis, treatment dosages, and surveillance intervals. However, the complexity and frequent updates of these documents create a "knowledge gap," where clinicians may not always have immediate access to the latest recommendations at the point of care.

While Large Language Models (LLMs) have demonstrated impressive capabilities in medical question answering, they fundamentally lack access to real-time data and often hallucinate facts when queried about specific clinical values. This limitation is not merely theoretical; it has direct consequences for patient safety. For example, when queried about the standard treatment for Hepatitis C, a model trained on older data might recommend **Interferon-based therapy**, a regimen now considered obsolete. In contrast, current AASLD 2024 guidelines recommend **Direct-Acting Antivirals (DAAs)**, which offer significantly better outcomes and fewer side effects. Following the LLM's outdated advice would result in a patient receiving inferior, potentially harmful care.

To address these limitations, we developed a Retrieval-Augmented Generation (RAG) system tailored specifically to AASLD guidelines. Unlike generic medical chatbots, our pipeline grounds every response in retrieved guideline text, ensuring that answers are both accurate and verifiable. Our contributions are as follows:

- A dynamic data ingestion pipeline that web-scrapes and converts AASLD guidelines into structured medical content, enabling the system to stay synchronized with the latest online updates.
- A domain-specific preprocessing workflow that filters raw guideline files into high-quality content sections, preserving critical tabular data and clinical thresholds.
- A comparative analysis of retrieval strategies, specifically evaluating the trade-offs between MultiQuery expansion and Cross-Encoder Re-ranking across multiple lightweight LLMs.

2 Related Work

LLMs in Healthcare. The application of LLMs in healthcare has evolved rapidly, moving from classification tasks to generative question answering. Despite their fluency, generative models suffer from "hallucination," where they confidently generate incorrect medical advice. Prior work has attempted to mitigate this via fine-tuning on medical corpora (e.g., Med-PaLM), but fine-tuning alone does not solve the issue of outdated knowledge or the inability to cite sources dynamically.

Retrieval-Augmented Generation (RAG). RAG addresses the static nature of LLMs by retrieving relevant documents from an external knowledge base during inference. In the biomedical domain, RAG has been applied to PubMed abstracts and general textbooks. However, clinical decision-making requires more than general literature; it requires strict adherence to approved practice guidelines. Few studies have focused on RAG systems restricted specifically to official society guidelines like those of the AASLD, where the preservation of numerical thresholds and dosages is paramount.

Retrieval Strategies. Standard RAG pipelines often rely on dense retrieval using cosine similarity. Recent research suggests that advanced retrieval techniques, such as query expansion (MultiQuery) and cross-encoder re-ranking, can significantly improve context quality. Our work extends this analysis to the specific domain of hepatology, comparing how these strategies interact with smaller, efficient language models suitable for local deployment.

3 Method Description

Our system architecture consists of two decoupled pipelines (Figure 2) designed to separate knowledge maintenance from real-time inference.

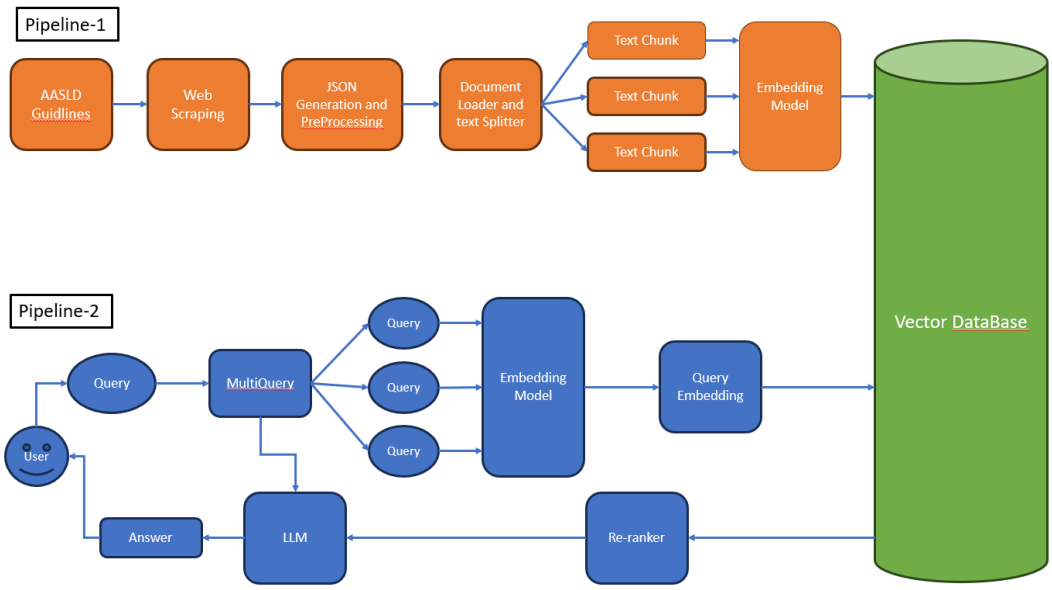

Figure 1: System Architecture. **Pipeline 1 (Top)** is an asynchronous process that refreshes the knowledge base every 2-3 months to incorporate new AASLD updates. **Pipeline 2 (Bottom)** is the online inference engine that processes user queries against the frozen vector index.

3.1 Pipeline 1: Asynchronous Knowledge Refresh

This pipeline operates offline and is triggered periodically (e.g., every 2-3 months) to ensure the knowledge base remains current.

- **Dynamic Web Scraping:** A crawler navigates the AASLD portal to identify and download new or updated guideline pages (HTML/PDF).
- **JSON Conversion & Preprocessing:** Raw documents are converted into structured JSON format. A cleaning module removes website boilerplate while preserving clinical hierarchy.
- **Indexing:** The cleaned text is segmented into overlapping chunks, embedded, and stored in a FAISS vector database. This ensures that the computationally expensive indexing process does not affect query latency.

3.2 Pipeline 2: Online Query Processing

The second pipeline handles real-time user interactions using the pre-built FAISS index.

- **MultiQuery Expansion:** The user's query is paraphrased into multiple variations to improve retrieval recall.
- **Retrieval & Re-ranking:** The system retrieves candidate chunks for these variations. A Cross-Encoder model then re-ranks them, filtering out irrelevant content.
- **Generation:** The top-ranked chunks are passed to the LLM to generate a final, cited answer.

4 Dataset Processing and Indexing

The dataset construction pipeline implements a rigorous multi-stage workflow designed to transform unstructured web content into a clean, semantically rich corpus suitable for clinical retrieval.

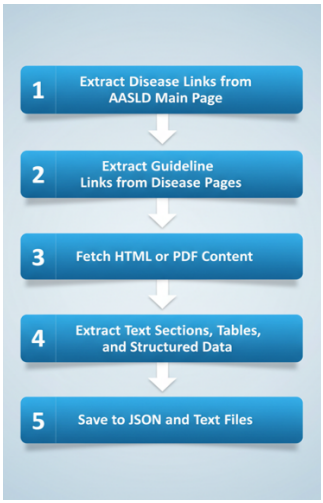

Figure 2: Data Extraction Flowchart

Data Acquisition and Normalization: We developed a custom crawler leveraging `Selenium` and `BeautifulSoup` to navigate the dynamic architecture of the AASLD portal. The ingestion module implements a robust traversal logic that handles JavaScript rendering and circumvents anti-bot challenges to systematically identify disease-specific guideline pages. The system discriminates between HTML and PDF sources: HTML content is parsed via DOM traversal to associate text with its nearest heading level, preserving hierarchical context, while PDF documents are processed using a stream-based parser to extract text while maintaining paragraph boundaries. All extracted content is normalized into a unified JSON schema that couples raw text with critical metadata—including source URL, publication date, and section hierarchy—to ensure full provenance tracking.

Domain-Specific Filtering: Raw web extractions inevitably contain non-clinical noise such as navigation menus, reference lists, and legal boilerplate. To address this, we designed a specialized filtering pipeline that processes the raw document structure through three distinct logical gates:

- **Semantic Header Filtering:** Every document section is evaluated against a set of exclusion criteria. Sections identified as administrative or non-clinical—such as bibliographies, funding disclosures, author acknowledgments, and supplementary data indices—are systematically pruned.
- **Content Validation:** The pipeline identifies and removes "empty" sections that contain metadata (like DOIs or page numbers) but lack substantial narrative text, ensuring that the retrieval index is not populated with low-value artifacts.
- **Deduplication Strategy:** We handle structural redundancies often introduced during scraping, specifically targeting introductory segments where section headings simply repeat the document title.

This rigorous filtering process reduced the initial corpus of 49 raw documents to **1,507 high-quality medical content sections**, significantly improving the signal-to-noise ratio of the knowledge base.

Semantic Segmentation and Indexing. Following filtration, the text is transformed into vector representations through a two-stage process.

First, we apply a **Recursive Character Splitting** strategy to segment the continuous text into discrete semantic units. We selected a chunk size of 512 tokens with a 50-token overlap. This specific configuration was chosen to prevent "context fragmentation" at sentence boundaries while fitting comfortably within the input window of our embedding model. The overlap ensures that logical assertions spanning across chunk breaks are preserved in at least one retrieval unit.

Second, these chunks are encoded using the `S-Bluebert-snli-multinli-stsb` model. We selected this specific architecture over generic transformers (e.g., `all-MiniLM`) because it is initialized from **BlueBERT**—a model pre-trained on PubMed abstracts and clinical notes—and further fine-tuned on Semantic Textual Similarity (STS) and Natural Language Inference (NLI) tasks. This dual-domain training enables the model to capture subtle clinical nuances (e.g., distinguishing between "hepatitis treatment" and "hepatitis diagnosis") that general-domain models often conflate.

The encoding process generated a dense vector space of **12,776 embeddings**. Each embedding vector has a dimensionality of **768**, which is the native hidden state size of the underlying BERT-base architecture, providing a rich representational capacity for capturing complex biomedical relationships. These vectors were indexed using a `FAISS IndexFlatIP` structure, which utilizes exact inner-product search to guarantee maximum retrieval recall without the approximation errors introduced by HNSW or IVF quantization methods.

5 Retrieval

The retrieval subsystem (Pipeline 2) is designed to maximize recall by mitigating the semantic rigidity inherent in single-vector search. While the retrieval logic is architectural and thus constant across our system, the generative components are driven by swappable LLM backends. We deployed this architecture using three distinct inference engines—**Llama-3.2-3B**, **Qwen-2B**, and **Phi-2**—which function both as query expanders and final answer generators.

MultiQuery Expansion Logic. Clinical queries often suffer from vocabulary mismatch, where a user's lay terminology (e.g., "liver scarring check") fails to map effectively to the technical language of guidelines (e.g., "fibrosis assessment"). To resolve this, we implemented a query expansion module that leverages the reasoning capabilities of the LLM. Before interacting with the vector index, the system intercepts the user's input and prompts the model to generate $N = 3$ independent, semantically equivalent variations.

The prompt engineering for this step is critical. We utilize a structured few-shot prompt designed to force the model to produce diverse, standalone questions without conversational filler. The exact instruction provided to the model is as follows:

```
"Given this medical question about AASLD liver disease guidelines:
{user_question}
```

```
What are {n} other related questions that would help
fully answer the above? Generate standalone, independent questions
(not paraphrases). Do NOT include the original question.
List one per line, no numbering, no explanations.
Related questions:"
```

Parallel Execution and Fusion. Once the variations are generated, the system executes parallel dense retrieval operations against the FAISS index. For the original query and each of its 3 variations, we retrieve the top-k most similar semantic chunks. This results in a broad initial pool of candidate segments.

Cross-Encoder Re-ranking. The initial retrieval phase—which aggregates results from the original query and its variations—casts a wide net. While this maximizes recall, it inevitably introduces "false positives": chunks that share keywords or vector proximity but lack true semantic relevance to the user's intent. To filter this noise, we implemented a second-stage **Re-ranking** module using a Cross-Encoder architecture (specifically `ms-marco-MiniLM-L-6-v2`).

Unlike the Bi-Encoder used for the FAISS index (which computes cosine similarity between pre-calculated vectors), the Cross-Encoder takes the query and a candidate document as simultaneous inputs, processing them through a full self-attention stack. This allows the model to evaluate the deep semantic relationship between the question and the text.

Cost-Benefit Analysis. This re-ranking step introduces a computational penalty: evaluating 20+ candidate pairs requires significantly more inference time than a simple dot-product lookup. However, the benefit is a dramatic increase in precision. By scoring chunks based on true relevance rather than just vector proximity, the Cross-Encoder effectively "promotes" the most informative guidelines to the top of the list and "demotes" irrelevant noise. This ensures that the limited context window of the generation LLM is filled only with high-quality, pertinent information, directly reducing the risk of hallucination in the final answer.

6 Augmentation and Generation

Following the retrieval and re-ranking stages, the final component of our pipeline is the Generation module. This stage synthesizes the retrieved clinical context into a coherent, citation-backed answer. We unified the generation logic across all three models (Llama-3.2, Qwen, and Phi-2) to ensure a fair comparison of their reasoning capabilities.

Context Integration. The top-k retrieved chunks are formatted into a string block where each chunk is preceded by its source metadata (e.g., `[Source: Hepatitis C Guidance 2024]`). This explicit labeling allows the model to attribute specific recommendations to their origin documents.

Prompt Engineering. We utilized a strict instruction-following prompt template designed to enforce clinical accuracy and prevent hallucination. The template explicitly constrains the model to answer *only* using the provided context. The exact prompt structure used in our experiments is as follows:

```
You are an expert hepatologist assistant.
Use ONLY the following context to answer the question.
If the answer is not in the context, say "I don't know".
Do not make up information.

Context:
{context}

Question:
{question}

Answer:
```

Inference Parameters. To balance creativity with factual adherence, we configured the generation parameters with a low temperature ($T = 0.1$) and a repetition penalty of 1.1. We set the maximum new token limit to 512, sufficient for detailed clinical explanations without inducing drift. This

configuration forces the model to act as a faithful synthesizer of the retrieved guidelines rather than a creative writer.

7 Experiment Settings

To evaluate the clinical utility of our RAG pipeline, we designed a comparative study isolating the impact of different Language Model backends and retrieval strategies.

Evaluation Dataset. We constructed a held-out test set of clinical queries derived directly from the AASLD guidelines (e.g., "What is the first-line treatment for HCC in non-cirrhotic patients?"). Each query was paired with a ground-truth answer extracted from the verified guideline text. The dataset covers three primary disease categories: Hepatitis B, Hepatitis C, and Hepatocellular Carcinoma.

Model Configurations. We evaluated three open-weight models chosen for their efficiency in resource-constrained environments. All models were loaded in 4-bit quantization using `bitsandbytes` (NF4 format) to simulate deployment on consumer-grade hardware (e.g., single NVIDIA T4 GPU).

- **Llama-3.2-3B-Instruct:** A commercially permissive model optimized for instruction following and reasoning.
- **Qwen-2-1.5B-Instruct:** A highly compact model from the Qwen series, selected to test the lower bound of parameter efficiency.
- **Phi-2 (2.7B):** A reasoning-focused model trained on textbook-quality synthetic data.

Baselines and Strategies. For each model, we compared two retrieval configurations:

1. **MultiQuery (MQ):** The baseline strategy where the LLM expands the query into $N = 3$ variations, retrieving the top-k chunks for each.
2. **Re-ranking (RR):** An enhanced strategy where the MultiQuery candidates are re-scored using the `ms-marco-MiniLM-L-6-v2` Cross-Encoder, selecting the top-5 chunks for the final context window.

Evaluation Metrics. We focused on three key indicators of RAG performance:

- **Retrieval Accuracy:** Measures the percentage of queries for which the correct ground-truth chunk was present in the top-k retrieved context.
- **NDCG@k (Normalized Discounted Cumulative Gain):** Evaluates the ranking quality of the retrieval system, penalizing relevant documents that appear lower in the list.
- **ROUGE:** Quantifies the textual overlap between the generated answer and the reference answer, serving as a proxy for factual faithfulness.

8 Experiment Results

We evaluated the performance of our RAG pipeline across three LLM backends (Llama-3.2, Qwen-2, and Phi-2). For each model, we conducted an ablation study comparing the baseline MultiQuery (MQ) configuration against the augmented pipeline with Cross-Encoder Re-ranking (RR). The quantitative outcomes are presented below.

Llama-3.2-3B Results. Figure 3 illustrates the performance delta for Llama-3.2. The model shows a measurable improvement in all metrics when moving from the baseline to the re-ranked setting. Notably, the ROUGE score increases from approximately 0.25 to over 0.31, the highest generation score recorded in our experiments.

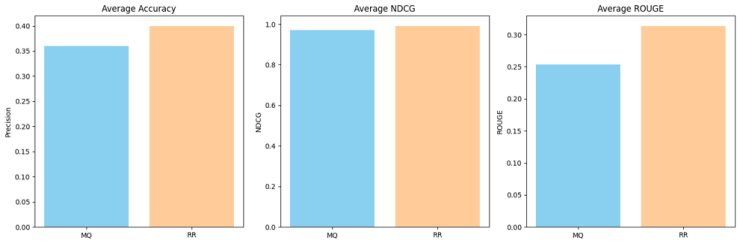

Figure 3: Llama-3.2 performance profile showing significant gains in generation quality (ROUGE) with Re-ranking.

Qwen-2B Results. Figure 4 presents the results for Qwen-2B. This model achieved the highest baseline retrieval accuracy ($\sim$0.45) among all candidates in the MultiQuery setting. However, its ROUGE scores remained comparatively low ($\sim$0.20) even after the application of re-ranking.

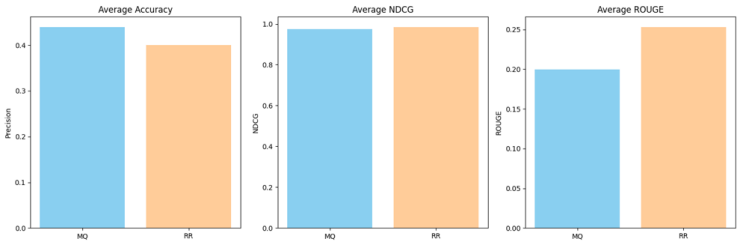

Figure 4: Qwen-2B performance profile. The model excels at retrieval accuracy but lags in generation scores.

Phi-2 Results. As shown in Figure 5, Phi-2 demonstrated consistent, monotonic growth. The application of Re-ranking improved retrieval accuracy from $\sim$0.44 to $\sim$0.48 and raised ROUGE scores to a competitive $\sim$0.25.

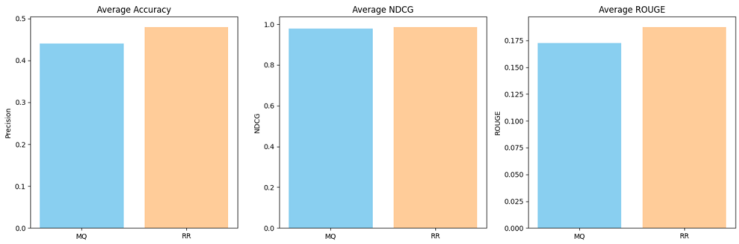

Figure 5: Phi-2 performance profile showing linear scaling across all metrics.

8.1 Latency Analysis

Figure 6 highlights the significant computational cost of the Re-ranking strategy. Across all architectures, adding the Cross-Encoder **more than doubles** the inference time. For example, the Llama-3.2 pipeline slows from 311ms to 722ms (a 132% increase), while Qwen-2B jumps from 215ms to 470ms. This confirms that the Re-ranker is a computationally expensive bottleneck, requiring full attention passes on candidate pairs, unlike the near-instant vector retrieval of the baseline MultiQuery approach.

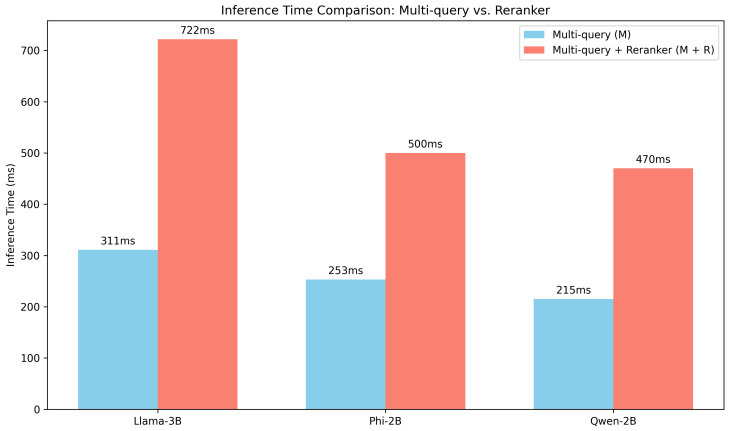

Figure 6: The Re-ranker (Red) drastically increases latency compared to the baseline (Blue).

9 Result Analysis

9.1 Model Performance Differences

Since all models received identical inputs, the differences in their results are due to how each model processes the information.

Llama-3.2 Performance. Llama-3.2 showed the biggest improvement when we improved the context quality. Its ROUGE score increased significantly after we applied the Re-ranker. This suggests that Llama-3.2 is very good at generating human-like answers, but it gets easily confused by irrelevant information. When the Re-ranker removes the noise, the model is able to write detailed, accurate answers that match the reference text well.

Qwen-2B Performance. Qwen-2B was the best at finding the right documents (highest retrieval accuracy). This is likely because it followed our instructions to "generate related questions" very strictly, creating better search terms. However, its ROUGE scores were low. This is likely because Qwen tends to give very short, direct answers. While these answers might be correct, they do not use the same wording as the detailed medical guidelines, leading to a lower score on text-overlap metrics.

Phi-2 Performance. Phi-2 was the most consistent model. It improved steadily in all metrics as we improved the retrieval method. It did not have the high retrieval peak of Qwen or the high generation peak of Llama, but it was reliable and predictable. This makes it a safe baseline choice.

9.2 System Design Recommendations

Based on the latency-accuracy trade-off observed in our results, we propose a Dynamic Re-ranking architecture rather than a static pipeline.

- **Selective Application:** Since Re-ranking doubles the inference latency (Figure 6), it should not be applied universally. Simple queries with high initial vector similarity scores likely do not require the expensive Cross-Encoder step.
- **Computation-Aware Routing:** A robust clinical system should implement a gating mechanism. The Cross-Encoder should be reserved for ambiguous or complex queries where the initial retrieval signals are weak, ensuring that the computational cost is incurred only when it yields a necessary accuracy benefit.

10 Conclusion

In this work, we evaluated a domain-specific RAG pipeline for AASLD guidelines using three Language Models. Our experiments identified a critical trade-off: while the Cross-Encoder Re-ranker significantly improved answer quality (particularly for Llama-3.2), it doubled the system's inference latency across all architectures. Qwen-2B proved most effective for retrieval, while Llama-3.2

excelled at answer generation, but neither could overcome the computational cost of re-ranking alone. Consequently, we recommend a **dynamic re-ranking strategy** for future clinical systems, where the expensive re-ranking step is selectively triggered only for complex or low-confidence queries, balancing the need for high accuracy with real-time responsiveness.

Author Contributions

We adopted a collaborative, pipeline-based approach to this project. The specific contributions of each author, mapped to the system architecture, are detailed below:

Pipeline 1: Data Engineering and Knowledge Base Construction

- **Web Scraping: Rohith Kumar Ballem** and **Ashfaq Ahmed Mohammed** co-developed the automated scraping module to acquire guideline documents from the AASLD portal.
- **Initial Data Cleaning: Abhigna Nimmagadda** implemented the raw text filtration logic to remove non-clinical artifacts and ensure data quality.
- **Preprocessing & Indexing: Ashfaq Ahmed Mohammed** and **Harsha Vardhan Reddy Palagiri** engineered the preprocessing pipeline, which included extracting relevant medical sections from JSON files, implementing the text chunking strategy, and generating vector embeddings using S-Bluebert.

Pipeline 2: Inference Engine and Retrieval Strategies

- **MultiQuery Retrieval: Rohith Kumar Ballem** designed and implemented the Multi-Query expansion module utilized across all three model architectures.
- **Re-ranking Module: Ashfaq Ahmed Mohammed** integrated the Cross-Encoder re-ranker to refine the retrieval candidate pool and improve context precision.
- **Model Integration:** The development of inference backends was distributed among the team:
  - **Qwen-2B:** Implemented by **Rohith Kumar Ballem**.
  - **Llama-3.2-3B:** Implemented by **Abhigna Nimmagadda**.
  - **Phi-2:** Implemented by **Harsha Vardhan Reddy Palagiri** and **Ashfaq Ahmed Mohammed**.

Analysis and Architecture

- **Inference Latency Analysis: Harsha Vardhan Reddy Palagiri** and **Abhigna Nimmagadda** conducted benchmarking studies to quantify the computational cost of the re-ranking step.
- **Results Evaluation:** All authors contributed equally to the qualitative and quantitative assessment of their respective model outputs.
- **System Architecture: Ashfaq Ahmed Mohammed** and **Rohith Kumar Ballem** supervised the overall end-to-end design and integration of the pipelines.

The source code is available publicly on GitHub at: RAG Pipeline.

References

[1] Terrault, N. A., et al. (2024). *AASLD Guidelines for Treatment of Chronic Hepatitis B and C*. American Association for the Study of Liver Diseases. `https://www.aasld.org/practice-guidelines`

[2] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.

[3] Touvron, H., Martin, L., Stone, K., et al. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

[4] Bai, J., Bai, S., Chu, Y., et al. (2023). Qwen Technical Report. *arXiv preprint arXiv:2309.16609*.

[5] Javaheripi, M., Bubeck, S. (2023). Phi-2: The surprising power of small language models. *Microsoft Research Blog*.