AskAlma: An Advanced Cache-Augmented RAG Chatbot for IIIT-Delhi website

Amartya Singh, Abhishek Bansal, Aditya Bagri

May 5, 2025

Contents

1 Introduction

The Indraprastha Institute of Information Technology Delhi (IIITD) website serves as a primary repository for a wide array of information crucial to its community, encompassing details on academic programs, admission procedures, faculty profiles, research activities, campus facilities, and administrative guidelines. However, the sheer volume and distribution of this information can make efficient access and synthesis challenging for users. Conventional search tools often fall short in providing direct, context-aware answers or in connecting related pieces of information scattered across different documents or pages.

To address this information access gap, the AskAlma project was initiated. AskAlma is conceived as an intelligent conversational agent, specifically designed to interface with the curated knowledge base derived from the IIITD website and associated official documents. Its core objective is to provide users—including prospective and current students, faculty, and staff—with accurate, relevant, and easily understandable answers to their IIITD-related inquiries.

This report documents the iterative development lifecycle of AskAlma. We detail the initial data gathering phase using BFS-based web scraping, the subsequent data refinement steps including fact extraction via SpanBERT, and the exploration of foundational AI strategies such as model fine-tuning (LoRA) versus Retrieval-Augmented Generation (RAG). We present the rationale, based on comparative evaluations, for adopting RAG as the core framework. Furthermore, we elaborate on the evolution from a baseline RAG implementation to the final, advanced architecture. This architecture distinctly combines Cache-Augmented Generation (CAG) for baseline stability and static data handling, parallel specialized retrievers (hybrid-search standard RAG and Graph RAG) for dynamic context acquisition, and a novel relevance-biasing mechanism using a smaller LLM to guide the primary generative model. This design seeks to maximize accuracy, robustness, and contextual relevance for IIITD-specific queries.

2 Methodology and Project Evolution

The development trajectory of AskAlma was characterized by systematic exploration and evaluation of different techniques, leading to progressive refinement of the system architecture.

2.1 Data Acquisition

The knowledge corpus for AskAlma was meticulously sourced from the official IIITD website and related resources.

- **Web Scraping Strategy:** A custom web scraper employing a Breadth-First Search (BFS) algorithm was implemented. The BFS approach systematically explores the website level by level from a starting page (IIITD homepage).
- **Scoping and Limits:** The scraper adhered to predefined rules, primarily targeting pages within the 'iiitd.ac.in' domain and respecting 'robots.txt' directives. For the initial development phase, the crawl was limited in scope (e.g., exploring approximately 20-21 linked pages) to manage data volume and processing time, resulting in roughly 300MB of initial data. This scope is designed to be easily configurable and expandable.
- **Content Type Extraction:** The scraper was designed to identify and download diverse content formats encountered during the crawl:
  - Raw HTML source code of web pages.
  - Linked document files (PDF, DOCX, DOC, XLSX).
  - Embedded HTML Tables (saved separately for potential structured processing).

- **Output Organization:** The scraped artifacts were systematically organized into distinct directories based on type (e.g., '/html', '/attachments', '/tables') to facilitate modular downstream processing.
- **Course Data Acquisition:** Separately, structured data pertaining to IIITD's Techtree (listing approx. 412 courses with details like descriptions, pre-requisites, anti-requisites) was obtained and prepared for graph-based modeling.

2.2 Data Preprocessing and Knowledge Base Construction

The raw scraped data underwent several transformation steps to create a usable knowledge base for the RAG system.

- **Document Content Extraction:** Standard libraries were employed to extract text from various file types: 'PyPDF2' for PDFs, 'python-docx' for DOCX, 'pandas' for XLSX, and 'BeautifulSoup' for HTML parsing. LibreOffice ('soffice' command-line tool) was integrated via 'subprocess' for robust conversion of legacy '.doc' files to text.
- **Text PDF Generation:** To standardize input from web pages, HTML files were programmatically converted into text-centric PDF documents. This involved specific libraries and custom functions designed to capture the core textual content while discarding heavy styling or complex layout elements irrelevant to text-based RAG.
- **Span-Based Fact Extraction:** A pre-trained SpanBERT model was applied to the extracted textual content (primarily from the generated Text PDFs and potentially key attachments). This process identified and extracted concise factual statements (spans) directly supported by the text. These facts, along with their source document provenance, were compiled into a large JSON file ('factual-data-spanbert.json'), serving as a source of high-precision factual snippets.
- **General Text Cleaning:** Standard text cleaning procedures were applied universally, including normalization of whitespace, case conversion (lowercase), and removal of non-essential special characters or processing artifacts (e.g., spurious 'nan' strings).
- **Knowledge Graph Construction (Courses):** The structured Techtree course data was transformed into a graph format suitable for Graph RAG. Nodes represent courses, and directed edges capture relationships like 'HAS-PREREQUISITE' and 'HAS-ANTIREQUISITE'. NetworkX or a similar graph library was used for this representation.
- **Procedural Paths File:** A dedicated structured file (e.g., JSON or Markdown) was manually made, containing clear, step-by-step instructions for common IIITD administrative or academic procedures identified as frequent query types (e.g., "How to apply for leave ?", "Where to find medical form ?").

The final dataset consisted of Text-pdfs we created, the raw HTML files, tables, and facts extracted using Spanbert, for the text retriver.

2.3 Initial Approaches and Baselines

Several modeling strategies were implemented and evaluated to determine the most effective approach for AskAlma.

2.3.1 Model Fine-tuning (LoRA)

- **Approach:** Fine-tuning experiments were conducted using Low-Rank Adaptation (LoRA) on a suitable base language model, a Deep-Seek distilled on a Qwen variant, for both reasoning power and large context window. The training data consisted of the processed textual corpus derived from our processed data.
- **Objective:** To adapt the model's internal representations and generative style to the IIITD domain.
- **Outcome & Evaluation:** While fine-tuning showed some improvement in domain-specific terminology and potentially better coherence in reasoning about IIITD concepts, it struggled significantly with factual recall and consistency. The model often generated plausible but incorrect information (hallucination) and could not reliably access or reproduce specific facts from the source documents. The performance gain in reasoning did not outweigh the loss in factual accuracy required for a reliable information bot.
- **Decision:** Based on these results, fine-tuning was deemed unsuitable as the core strategy for AskAlma.

2.3.2 Naive BM25 RAG Pipeline

- **Approach:** A baseline RAG system was implemented using the classic BM25 algorithm for retrieval.
- **Retriever:** Text data (from various sources) was chunked, and BM25 was used to retrieve chunks based on lexical (keyword) similarity to the user query.
- **Pipeline:** Query → BM25 Chunk Retrieval → Context Assembly → Base LLM Prompting → Response.
- **Outcome & Evaluation:** This approach demonstrated significantly better factual grounding compared to fine-tuning, as responses were directly conditioned on retrieved source text. However, its reliance on keyword matching limited its ability to handle semantically similar but lexically different queries.

2.3.3 Comparison: RAG vs. Fine-tuning

- **Key Finding:** RAG fundamentally outperformed fine-tuning in providing verifiable, factually grounded answers based on the IIITD knowledge base.
- **Hybrid Test Assessment:** A hybrid approach, using the fine-tuned model as the generator within the BM25 RAG pipeline, was also evaluated. This did not yield substantial overall improvements. The fine-tuned model's tendency to sometimes disregard or contradict the provided retrieved context offset potential gains in domain understanding.
- **Strategic Conclusion:** The RAG paradigm was definitively chosen as the foundational methodology for AskAlma due to its inherent strengths in factual grounding, transparency (traceability to source documents), and relative ease of knowledge updates (updating the index vs. retraining the model).

2.4 Advanced RAG Development

Having established RAG as the core strategy, efforts focused on overcoming the limitations of the naive BM25 approach and building a more capable system.

2.4.1 Motivation for Advancement

The need for improved semantic understanding (beyond keywords), the requirement to effectively handle the structured relational data of the course Techtree, and the desire for greater overall robustness drove the development towards more advanced RAG techniques.

2.4.2 Graph RAG for Course Data

- **Rationale:** Standard text retrieval methods are ill-suited for efficiently querying structured relationships like course pre-requisites or anti-requisites.
- **Implementation:** A dedicated Graph RAG component was developed. The previously constructed course knowledge graph was utilized. A Graph Retriever mechanism was implemented using appropriate libraries (LangChain's graph integrations) to translate natural language queries about courses into graph traversals, retrieving relevant course nodes and their connections.
- **Benefit:** This yielded a marked improvement in the accuracy and completeness of answers to course-related queries, correctly identifying dependencies and related courses in a way that was unreliable with text-only retrieval.

2.4.3 Enhanced Standard RAG for General Content

- **Embedding Model:** 'sentence-transformers/all-MiniLM-L6-v2' was selected for generating dense vector embeddings of text chunks, capturing semantic meaning.
- **Vector Store:** ChromaDB was employed as the vector database for efficient similarity search over the embeddings.
- **Multi-Level Chunking Strategy:** To balance context integrity and retrieval granularity, a two-stage chunking process was adopted:
  1. *Initial Broad Chunking:* 'RecursiveCharacterTextSplitter' was used with relatively large chunk sizes and substantial overlap. This minimizes the chance of splitting closely related information across chunks initially.
  2. *Semantic Refinement:* Techniques based on semantic similarity (e.g., using embedding distances or dedicated semantic chunking algorithms) were then applied to these larger chunks. This step aims to break down or regroup text based on topical coherence, creating more focused chunks for the LLM while leveraging the context captured by the initial overlap.
- **Hybrid Search Retrieval:** The retriever combined dense vector search (via ChromaDB embeddings) with sparse keyword search (e.g., BM25). This hybrid approach leverages the strengths of both methods – semantic understanding from vectors and precise term matching from keywords. MMR was potentially used for re-ranking to ensure diversity in the final retrieved set.

3 Final Architecture: AskAlma System

The culminating architecture of AskAlma represents a synthesis of the explored techniques, designed for robustness, accuracy across diverse data types, and guided generation.

3.1 Architectural Philosophy

The core idea is to establish a reliable baseline using cached static knowledge (CAG) and then augment this with dynamically retrieved, specialized context (Standard RAG + Graph RAG), with a guiding mechanism to help the final LLM focus effectively.

3.2 Components

1. **CAG Layer & Static Knowledge Cache:**
   - **Mandatory First Step:** Processes every incoming query initially.
   - **Static Data Store:** Utilizes a pre-loaded, efficiently accessible cache (conceptually, an external knowledge KV store) holding foundational information: AskAlma's defined persona, essential IIITD facts (e.g., founding details), common FAQs, and the curated procedural "how-to" guides.
   - **Function:** Prepares a 'static context block' containing relevant baseline information derived from the cache. This ensures the chatbot always operates within its defined persona and can handle basic static queries reliably, irrespective of dynamic retrieval outcomes.
2. **Parallel Dynamic Retrieval Pipeline:** Activated subsequent to the CAG layer.
   - **Standard RAG Retriever:** Executes hybrid search (vector + keyword) over the indexed general content (website text, docs, facts) using the multi-level chunking strategy.
   - **Graph RAG Retriever:** Queries the course knowledge graph using graph traversal techniques based on the user query, retrieving structured course relationship information.
3. **Small LLM Relevance Classifier (Qwen 0.5B):**
   - **Input:** The original user query.
   - **Task:** Performs rapid classification of the query's primary information need (Static/FAQ, General Website Info, Course Info, Mix).
   - **Output:** Generates a concise textual "Relevance Bias Instruction" (e.g., '"Instruction: Focus primarily on the provided graph context for course details."' or '"Instruction: Blend information from static context and general text chunks."').
4. **Context Aggregation & Final Prompt Construction:**
   - Assembles the complete input for the main LLM. This includes:
     (a) Static Context Block (from CAG Cache).
     (b) Retrieved Text Chunks (from Standard RAG, potentially ordered).
     (c) Retrieved Graph Information (formatted as text, from Graph RAG).
     (d) Relevance Bias Instruction (from Small LLM).
     (e) The Original User Query.
   - Structured formatting (e.g., distinct headers for '[Static Context]', '[Website Document Context]', '[Course Graph Context]', '[Guidance Instruction]') is employed within the prompt template to clearly delineate these information sources for the main LLM.
5. **Main Generative LLM (Qwen 8B):**

- **Input:** The fully assembled prompt containing all context types and the relevance bias instruction.
- **Process:** Utilizes its large context window and reasoning capabilities, guided by the bias instruction, to synthesize an answer that draws appropriately from the provided static and dynamic information sources.
- **Output:** Generates the final response adhering to the AskAlma persona.

3.3 Pipeline Diagram

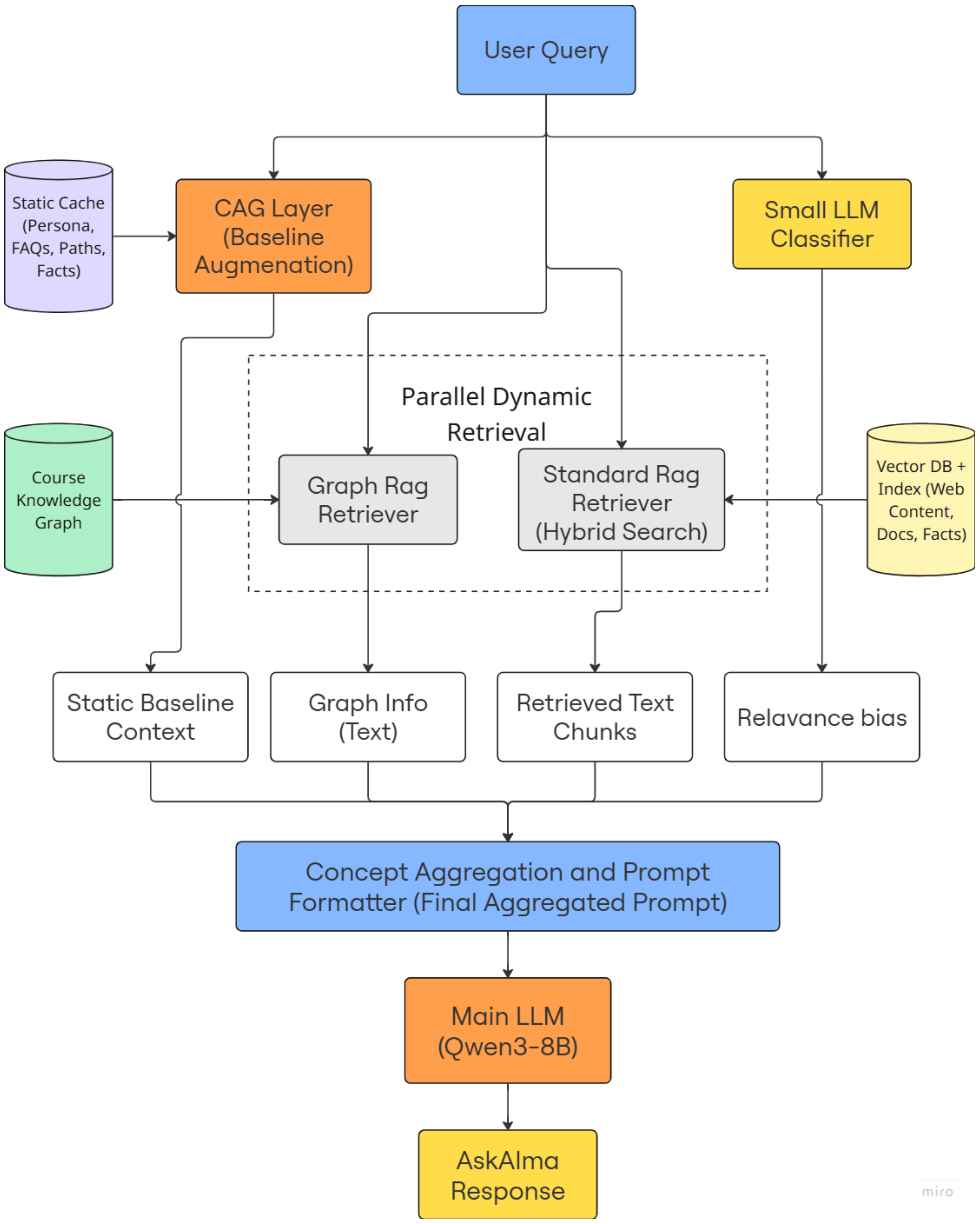

User Query
Static Cache (Persona, FAQs, Paths, Facts)
CAG Layer (Baseline Augmentation)
Small LLM Classifier
Parallel Dynamic Retrieval
Course Knowledge Graph
Graph Rag Retriever
Standard Rag Retriever (Hybrid Search)
Vector DB + Index (Web Content, Docs, Facts)
Static Baseline Context
Graph Info (Text)
Retrieved Text Chunks
Relavance bias
Concept Aggregation and Prompt Formatter (Final Aggregated Prompt)
Main LLM (Qwen3-8B)
AskAlma Response
miro

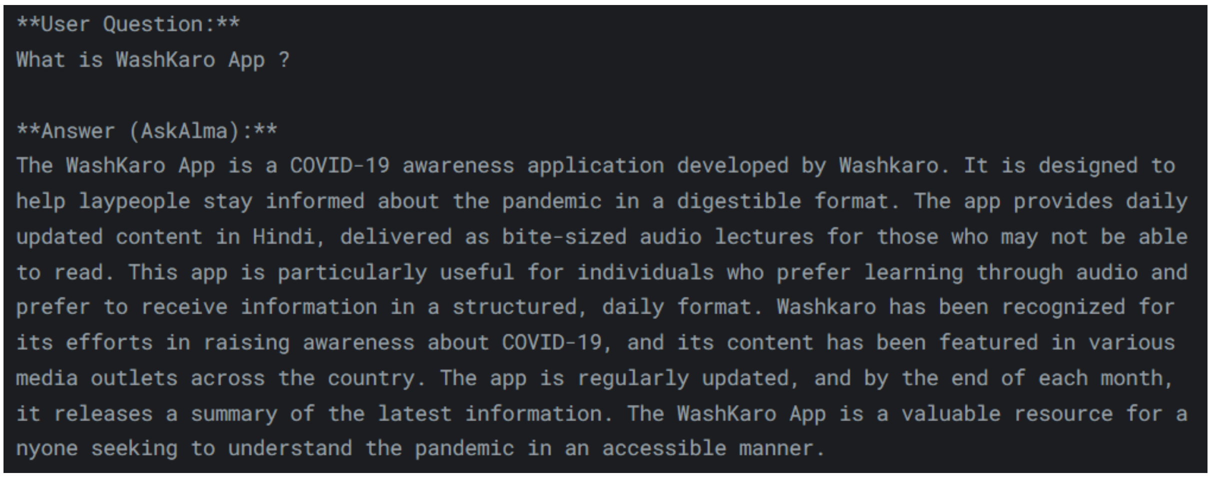
Figure 1: AskAlma - Pipeline

4 Implementation Details

- **Core Frameworks/Libraries:** Primarily developed using Python with core libraries including LangChain for pipeline orchestration, Transformers for model interaction, PyTorch as the ML backend, ChromaDB for vector storage, NetworkX for graph representation, Pandas/PyPDF2/python-docx/BeautifulSoup for data ingestion, and embeddings for semantic chunking and Graphrag for graph database interaction.
- **Execution Environment:** Development and testing leveraged Kaggle notebooks with GPU acceleration for model inference and embedding computations. Then later we shifted to college server for final pipeline as larger model caused issues in Kaggle environments.
- **Selected Models:**
  - Embedding Generation: 'sentence-transformers/all-MiniLM-L6-v2'.
  - Initial Fact Extraction: SpanBERT.
  - Relevance Classification: A small, efficient model: Qwen2.5 - 0.5B.
  - Main Generative Model: An 8B parameter Qwen3 model variant.

5 Evaluation (Preliminary)

- **Evaluation Strategy:** Performance was assessed iteratively throughout development. Initial comparisons focused on qualitative analysis of responses from fine-tuned vs. RAG models. The final architecture was evaluated using a set of test queries covering static information, general website content, and specific course-related questions. Metrics emphasised on manual assessment of factual accuracy, relevance, coherence, and adherence to the AskAlma persona and facts.
- **Key Findings:**
  - RAG consistently outperformed fine-tuning for factual accuracy based on the provided knowledge base.
  - The Graph RAG component demonstrably improved handling of queries involving course pre-requisites and anti-requisites compared to standard text retrieval alone.
  - The CAG layer reliably enforced the AskAlma persona and provided correct answers for cached static/FAQ data, acting as an effective safety net.
  - The small LLM relevance classifier showed potential in guiding the main LLM towards more focused answers in many cases, and did not introduce significant errors in any case thus improving focus overall.
- **Example Interactions:**

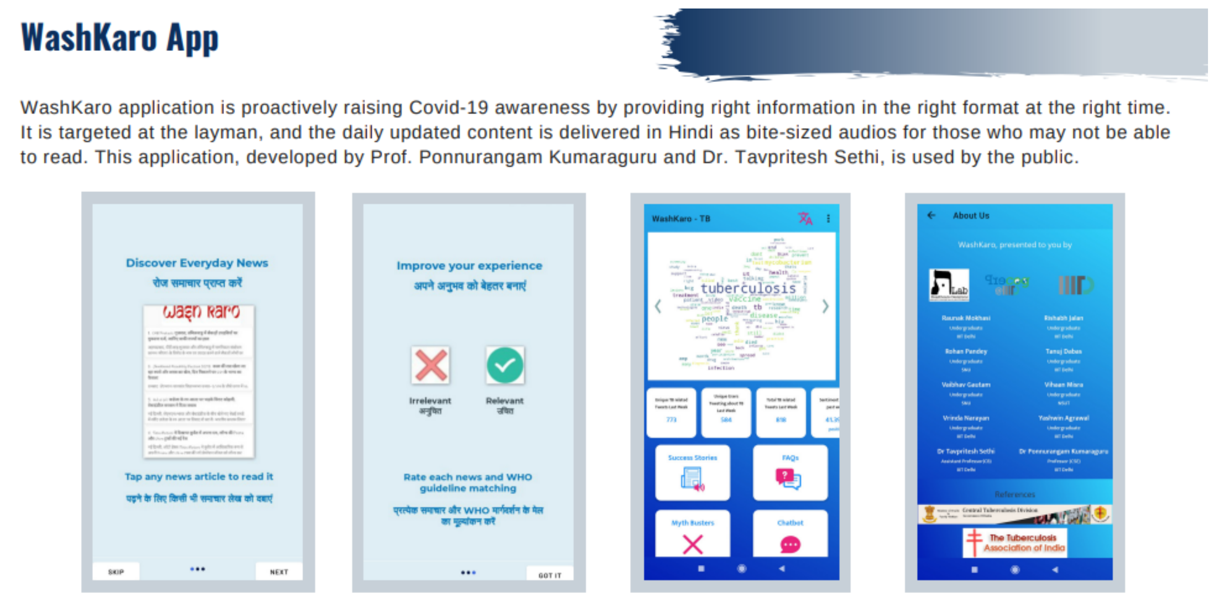


```
**User Question:**
What is WashKaro App ?

**Answer (AskAlma):**
The WashKaro App is a COVID-19 awareness application developed by Washkaro. It is designed to
help laypeople stay informed about the pandemic in a digestible format. The app provides daily
updated content in Hindi, delivered as bite-sized audio lectures for those who may not be able
to read. This app is particularly useful for individuals who prefer learning through audio and
prefer to receive information in a structured, daily format. Washkaro has been recognized for
its efforts in raising awareness about COVID-19, and its content has been featured in various
media outlets across the country. The app is regularly updated, and by the end of each month,
it releases a summary of the latest information. The WashKaro App is a valuable resource for a
nyone seeking to understand the pandemic in an accessible manner.
```

Figure 2: Actual response of the pipeline

This comes form the IRD Newsletter of Jan 2021, Vol - 1, its image is attached below.



WashKaro App

WashKaro application is proactively raising Covid-19 awareness by providing right information in the right format at the right time. It is targeted at the layman, and the daily updated content is delivered in Hindi as bite-sized audios for those who may not be able to read. This application, developed by Prof. Ponnurangam Kumaraguru and Dr. Tavpritesh Sethi, is used by the public.

Discover Everyday News
रोज समाचार प्राप्त करें
Tap any news article to read it
पढ़ने के लिए किसी भी समाचार लेख को दबाएं
SKIP
NEXT
Improve your experience
अपने अनुभव को बेहतर बनाएं
Irrelevant
Relevant
Rate each news and WHO guideline matching
GOT IT
WashKaro - TB
Success Stories
FAQs
Myth Busters
Chatbot
About Us
WashKaro, presented to you by

Figure 3: Source of the response from IRD Newsletter

- **Identified Limitations:** Current limitations include the restricted scope of the initial web crawl (impacting knowledge breadth), latency from the multi-stage pipeline and the need for further tuning of the multi-level chunking and relevance classification components.

6 Conclusion

The AskAlma project successfully navigated the complexities of building a domain-specific conversational AI for IIITD. Through a process of iterative development and evaluation, starting from data acquisition and progressing through baseline model testing to advanced RAG techniques, a robust and sophisticated final architecture was realized. This architecture leverages Cache-Augmented Generation (CAG) to ensure a stable operational baseline and handle static information efficiently. It incorporates specialized retrieval mechanisms—hybrid search for general content and Graph RAG for structured course data—within a parallel pipeline. Furthermore, it introduces a novel relevance guidance step using a small LLM classifier to enhance the main generative model's focus. The resulting system demonstrates a powerful approach to integrating diverse knowledge sources and AI techniques, providing a strong foundation for an effective IIITD information chatbot.

7 Future Work

Potential directions for enhancing AskAlma include:

- **Knowledge Base Expansion:** Significantly increase the breadth and depth of the web scraping process to cover a larger portion of the IIITD website and potentially incorporate other official documents or portals.
- **Chunking Retrieval Optimization:** Conduct detailed experiments to fine-tune the multi-level (recursive + semantic) chunking parameters and evaluate alternative hybrid search weighting strategies.
- **Rigorous Evaluation:** Implement a comprehensive evaluation suite using established frameworks like RAGAS to quantify performance across metrics like faithfulness, answer relevance, context recall, and context precision.
- **Relevance Classifier Tuning:** Experiment with different small LLMs or prompt strategies for the relevance classifier and rigorously assess its impact on end-to-end performance versus simpler context combination methods.
- **Graph RAG Enhancements:** Refine the knowledge graph schema and explore more advanced graph querying techniques for complex course-related inquiries.
- **Latency Reduction:** Analyze pipeline latency and implement optimization strategies, such as asynchronous retrieval or caching intermediate results (e.g., retrieval outputs for common queries).
- **Feedback Integration:** Develop mechanisms for capturing user feedback on responses to enable continuous learning and improvement cycles.