

2025-12

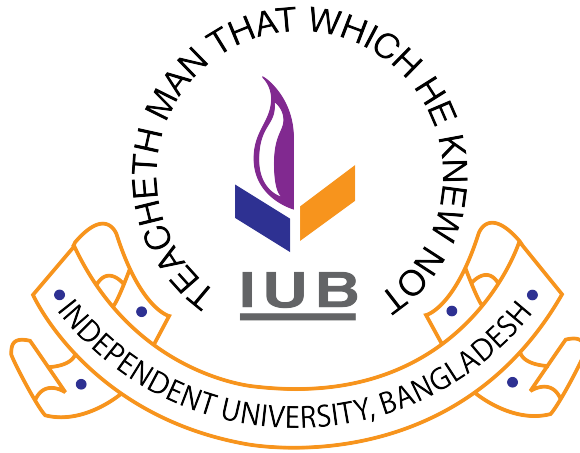
IUB-RAG-LLM: an extensible system for continuous knowledge ingestion and asynchronous indexing

Nahiyen, Zulker Nayeem

Independent University, Bangladesh (IUB)

<https://ar.iub.edu.bd/handle/11348/1641>

Downloaded from IUB Academic Repository



IUB-RAG-LLM: An Extensible System for Continuous Knowledge Ingestion and Asynchronous Indexing

December 2025

Prepared by:

Zulker Nayeem Nahiyen
ID: 1910063

Department of Computer Science and Engineering
Independent University, Bangladesh

Supervised by

Amin Ahsan Ali
Professor
Department of Computer Science and Engineering
Independent University, Bangladesh

Attestation

We acknowledge that plagiarism is strictly prohibited and conflicts with our university's regulations. We guarantee the authenticity of our work, and any copyrighted material or models developed by others used in this project have been properly cited in accordance with international standards and guidelines.

Author Name:

.....

Signature:

.....

Evaluation Committee

Supervisor

Name: _____ Signature: _____

Internal Examiner 1

Name: _____ Signature: _____

Internal Examiner 2

Name: _____ Signature: _____

External Examiner

Name: _____ Signature: _____

Acknowledgement

We would like to express sincere gratitude to our Professor, Amin Ahsan Ali, for his patience and guidance. His suggestions were crucial during the research work and his expertise contributed strongly to the project. We appreciate his time and support invested towards the completion of this academic endeavor.

Furthermore, we would like to thank the respected faculty at the CCDS Lab (Center for Computational & Data Sciences) for providing a work environment conducive to effective research, cross-domain collaboration, and lively discussions. We received consistent support and feedback starting from the early phases of the project, and this incubation was crucial in shaping the direction of our research. Additionally, we were granted unique opportunities to regularly attend international workshops, talks from esteemed speakers, and so on, which significantly broadened our professional horizon.

Finally, we would also like to acknowledge our colleagues, friends, and family for their timely encouragement towards the project.

Disseminating institutional knowledge at universities is challenging, resulting in information gaps for students and increased burdens on both faculty and staff who must answer repetitive queries. We present IUB-RAG-LLM, an extensible retrieval-augmented generation system designed to provide instant access to institutional knowledge. The system features an asynchronous data pipeline that reduces knowledge update latency by 6x, and a modular architecture that enables integration of new data sources and long-term maintainability. We created a corpus of 2834 documents produced from official university channels, which is processed into 7798 semantic chunks and continuously updated. A preliminary human evaluation study with 3 student participants assessing 60 question-answer pairs demonstrated the system’s effectiveness, with average correctness and completeness scores of 3.57/5.00 and 3.67/5.00, respectively. The IUB-RAG-LLM system shows promise in alleviating faculty and staff workload while improving student access to accurate, timely institutional information. Our project can be found at <https://github.com/znnahiyani/iub-rag-llm>.

Contents

1	Introduction	8
1.1	Background	8
1.2	Problem Statement	11
1.3	Outcome	11
2	Literature Review	13
2.1	Language Models and the RAG Paradigm	13
2.1.1	Evolution of Large Language Models	13
2.1.2	Retrieval-Augmented Generation	13
2.2	Institutional Knowledge Management Systems	14
2.2.1	University Chatbots	14
2.3	Limitations of Existing Approaches	14
2.4	Contributions of IUB-RAG-LLM	15
3	System Design	16
3.1	System Overview	16
3.2	Data Ingestion Pipeline	16
3.2.1	Web Data Source	17
3.2.2	Document Data Source	17
3.3	Asynchronous Indexing Pipeline	17
3.3.1	Document Chunking	17
3.3.2	Vectorization Process	18
3.4	Vector Database	18
3.4.1	Asynchronous Processing	18
3.5	Query Processing and Response Generation	18
3.5.1	Query Rewriting	18
3.5.2	Semantic Search	19
3.5.3	Response Generation	19
3.6	User Interface	19
3.7	Modular Architecture	19
4	Corpus Dataset	21
4.1	Motivation & Purpose	21
4.2	Source of Data Samples	21
4.3	Structure and Features of Dataset	21
4.3.1	Chunking Strategy	21
4.3.2	Embedding and Vectorization	21
4.3.3	Corpus Statistics	22

5	Experimental Design	23
5.1	System Configuration	23
5.2	Evaluation Methodology	23
5.2.1	Human Evaluation Survey Design	24
5.2.2	System Performance Benchmarking	25
6	Results and Analysis	26
6.1	Human Evaluation Results	26
6.2	Sample Question–Answer Analysis	26
6.2.1	Example of Good Performance	26
6.2.2	Example of Average Performance	27
6.2.3	Example of Poor Performance	28
6.3	Qualitative Analysis of User Feedback	28
6.4	Qualitative Analysis of System	28
6.5	System Performance Results	28
7	Conclusion and Future Works	29
7.1	Conclusion	29
7.2	Limitations	29
7.3	Ethical Considerations	29
7.4	Impact on the Society	29
7.4.1	Sustainability of the Work	30
7.5	Future Plan	30
	Bibliography	30

List of Figures

1	High-level architecture of the IUB-RAG-LLM system, showing the flow from data sources to the end user.	16
---	--	----

1 Introduction

1.1 Background

Institutional knowledge is the set of useful knowledge and wisdom collected by an institution from their working practices and historical data. It is generally divided into two types: implicit, referring to the unwritten, informal, and often tacit knowledge shared within an institution, and explicit, being the documented, formal knowledge, which may be intentionally communicated and easily transferred through prepared media such as reports, manuals, and handbooks.

Students, faculty, and staff regularly access institutional knowledge in order to perform their duties. These routine contacts typically consist of procedural updates, personal guidance, or official applications. Help desks, office appointments, and documentation are the primary channels for such communication.

Despite the availability of various traditional channels, disseminating institutional knowledge still remains difficult, as explicit instruction, orientation, or mentorship are limited by institutional factors, and thus prove insufficient in conveying pertinent information to key groups, such as new students and guests.

Direct inquiries to faculty and staff can put significant demand on their operational capacity, reducing their time available for their primary duties. For example, students are typically entitled to access multiple modes of communication, such as physical presence during office hours or prompt response through email. This strain between the faculty & staff’s constraints and the students’ demands can lead to delayed or incomplete access to information, hindering students’ academic progress and learning outcomes.

Manual searches for desired information from various sources (including search engines, websites, handbooks, notices, news articles, and social groups) can be very time-consuming and unengaging. Although it remains a viable option, users may still be unsure if their desired information can be found within a reasonable expenditure of time and effort. According to [8], people preferentially pursue actions with greater outcome certainty, despite higher costs. Consequently, students may tend to directly enquire faculty and staff instead as first choices, which puts further strain on their workloads and demands, and directly impacting their operational capacity.

Large language models (LLMs) have emerged recently as a new paradigm for accessing information [2, 24]. They provide a human-computer interface for users to communicate freely through natural language, making it an extraordinarily-accessible channel as compared to traditional digital channels such as search engines or mass communication media.

LLMs are created by optimizing a mathematical model with adjustable parameters to predict the patterns in a large corpus of text [1]. Once pre-trained, the model is further optimized to generate answers to prompts that align with objectives such as correctness, relevance, and safety [18]. The model is then applied repeatedly to input text in order to predict the likelihood of the next token in a recurrent loop, therefore generating an output

sequence of text. Training is typically an extremely computationally-expensive process which requires hundreds of thousands of servers to prepare a model, so such endeavors are restricted to entities such as large corporations or governments [13].

Given recent advancements in the technology, owing to both algorithmic breakthroughs as well as scaling improvements in datasets, models, and hardware, LLMs have become powerful enough to be able to drive **conversational AI** which can understand and generate human-like text [19, 21]. These advanced general-purpose closed-domain LLMs have become ubiquitous as they are offered as both free and paid services on all major digital platforms and devices, from websites and social media to desktops and smartphones. Entities which provide such services in this capacity are referred to as **LLM providers**.

LLMs provide an innovative solution to sharing institutional knowledge on a massive scale, at cheap cost structures, with instantaneous responses, and digital accessibility from anywhere and availability at all times. Hence, it is not surprising that both user and enterprise adoption are currently growing exponentially, and have reached hundreds of millions of weekly active users worldwide. These fundamental qualities present the implementation of conversational AI at large institutions as a very attractive proposition since it constitutes a potential solution to communicating institutional knowledge through a highly effective format.

However, due to the nature of the training process, the state-of-the-art models from LLM providers currently encompass several important limitations related to our particular goals:

- **Knowledge cut-offs.** As the training process is prohibitively expensive, operationally complex, and dependent on market considerations, LLM providers have proven to update their LLM offerings only on a semi-annual basis, or in 6 to 12 month intervals. This introduces dates from which knowledge has been "cut-off," as the models were not trained on recently available data, such as current events or newly-authored information.
- **Lack of specific institutional context.** Institutions generally have a large amount of valuable internal data, but it is not available to LLM providers. Instead, providers train their general-purpose models on a mixture of data from both openly-available and proprietary sources [22]. As such, their published models lack specific knowledge that is uniquely available to any single institution.
- **Risk of hallucinating incorrect information.** LLMs still make mistakes in their answers. This often happens when the model's training data underrepresents the relevant knowledge, which makes the prediction of the generated words or tokens uncertain. On the other hand, it can occur if errors are present in the training data itself, which causes the model to generate an incorrect answer confidently [27]. Both of these issues are the result of insufficient or incorrect training data used by the provider, an element that is not controlled by the institutions using their models.

Retrieval-augmented generation (RAG) is a recent methodology that aims to solve these LLM limitations, by first retrieving relevant information from an external knowledge base which is trusted and up-to-date [9]. Then, by feeding the retrieved data as part of the LLM's input, the generated output is conditioned on the external knowledge over the LLM's unreliable internal knowledge, therefore generating a more confident, correct answer.

In a traditional RAG system, the knowledge base is first populated by ingesting documents through a process called **indexing**, which requires a document chunking strategy,

a semantic text embedding model, and a vector database [11]. Each document is processed to produce document chunks, and the semantic text embedding model is applied to each chunk to produce semantic embedding vectors. Then, each document chunk, its corresponding embedding vector, and associated metadata, are used to update/insert the relevant record in the database.

Afterwards, in order to access the same knowledge base through a query text, we perform **semantic search**: the query text is used to compute the query embedding vector, which is used to perform a nearest-neighbor search to scan through the database and find the top- k embedding vectors with the highest cosine similarity scores and their corresponding document chunks, which are the result of this query [10].

Since a semantic embedding vector represents the semantic meaning of its corresponding text, the cosine similarity between two embeddings represents the similarity in meaning between their corresponding texts [23]. There are many popular sparse (non-parametric) and dense (parametric) semantic text embedding models, as well as hybrid models which utilize both together [6, 5]. (Also known as *learnable sparse embeddings*.) The database scan is performant due to the use of approximate nearest neighbor indices, which use various techniques to speed up the search step [12, 7].

Known as **in-context learning** (ICL), inserting data into the input prompt is a general technique has been found to make LLMs grounded by increasing the accuracy of its responses with respect to the given context information [3]. Most state-of-the-art models are now specially trained to improve performance on different categories of ICL tasks, which include: few-shot classification tasks, complex reasoning tasks, long-context retrieval tasks, and agentic/tool-use tasks [25]. All of these techniques applied through ICL improve the performance of RAG systems on external knowledge bases.

Yet, for traditional RAG systems, there are still challenges or shortcomings of the design which make it unsuitable for our exact requirements.

Knowledge staleness is knowledge which has been outdated by fresh knowledge at the data sources. Traditional RAG systems implement a slow, batch-based ingestion process, which exacerbate issues with knowledge staleness.

Ingestion complexity is an important, evolving challenge. Traditional RAG systems only support a single, static data source, which is updated infrequently through a slow, batch process. For example, the data ingestion pipeline may consist of a simple web crawler which is run on a weekly schedule, to download all pages and files from a website, and then index them all at the same time.

In the early world wide web, most websites were built with either *static sites*, *static-site generation* (SSG), or *server-side rendering* (SSR), the last of which was the rendering strategy of traditional PHP frameworks in which the server builds the document-object model (DOM) and then returns the final, rendered HTML to the client. In that ecosystem, a simple web crawler is sufficient for collecting all of the data from the website, and therefore comprises a very simple, small portion of the overall system design.

However, modern web development has transformed the web ecosystem and have made two alternative rendering strategies popular: *client-side rendering* (CSR) and *incremental static regeneration* (ISR), both of which return JavaScript code that render the content at the client side. The rise in popularity of these rendering strategies is strongly associated with the advent of JavaScript frameworks such as *Angular*, *React*, and *Vue*, which are used to implement single-page applications (SPAs).

As a result, simple web crawlers completely fail to ingest data from such SPAs, because they do not have the necessary capability to maintain a functional JavaScript execution environment, which is the problem domain of fully-featured browser engines. Therefore, the usage of advanced web crawlers is necessary to render the content fully on the client

side, which may bring additional complexity.

Performance bottlenecks. As we are regularly updating the entire knowledge base, which scales as the institution’s data grows, the number of documents that need to be processed for every full data ingestion run can be very large, typically amounting to thousands of documents. The time elapsed is the minimum possible update frequency of the knowledge base, and so it is a critical bottleneck towards achieving a real-time knowledge base. The indexing process (chunking, embedding, storing) can also be very slow and inefficient, thereby preventing timely updates [26]. The throughput for data ingestion and indexing have to be carefully considered: if each document took 30 seconds to ingest, then that would take an estimated 8.33 hours to ingest only 1000 pages.

Architectural rigidity. Many RAG systems are monolithic, making it hard to swap out components (like the embedding model or vector database) or add new data sources.

1.2 Problem Statement

Existing RAG systems are not well-suited for the dynamic environments of institutions such as universities, where information is constantly changing and diverse sources must be integrated efficiently. Therefore, the objective is not just to solve the challenges of RAG, but to also fit the system and solve the concrete needs of institutions themselves.

Knowledge Staleness. Traditional RAG systems have static knowledge bases, which is not appropriate for living institutions with regular events and updates. In the university setting, for instance, it may mean that a student gets outdated deadline information or that a faculty member follows an old policy, both of which are failures of the system with critical consequences for the user, which is unacceptable.

Therefore, a dynamic knowledge base is required, which also necessitates an automated, robust, and efficient data ingestion and indexing pipeline. If the pipeline is invoked periodically, then we can minimize the knowledge age by increasing the update frequency from the external data sources to our knowledge base.

Data Ingestion & Indexing Inefficiency. Advanced techniques such as parallelism and concurrency are crucial in managing data ingestion and indexing inefficiencies.

Architectural Inflexibility. As newer models for both LLMs and semantic text embeddings are developed, it becomes harder to migrate traditional RAG systems due to their tight coupling of software components. Hence, they often have a high level of technical debt.

An architecturally flexible system, which can orchestrate multiple interdependent models and components, is a requirement for long-term maintainability, and this is possible through abstract modeling of the interfaces for each different type of software component, such as databases, retrievers, generators, and so on. Thus, the orchestration of the system can proceed using the abstract interfaces rather than the concrete implementations of each library or API, which enables replacement of components.

1.3 Outcome

To address these problems, we developed **IUB-RAG-LLM**, an extensible system for continuous knowledge ingestion and asynchronous indexing.

We present an AI assistant adapted for institutions, including universities, to facilitate routine communication for its members, based on retrieval-augmented generation and large language models.

The framework has a modular implementation, allowing design parameters for ingestion, indexing, and inference to be independently optimized, while also being highly performant by using asynchronous data pipelines to achieve both job concurrency and batch processing.

Maintaining the plethora of models used by a chat system is difficult, as these models need to be closely interoperated, routinely updated with better-performing alternatives, and monitored individually for performance metrics and quality control. We use strategy design patterns, context objects, and dynamic configuration to run models adaptably. By decoupling models from the overall chat system, both software maintenance and architectural design are simplified.

Our proposal defines an AI assistant as a text-based system designed to answer questions and provide information to its users. It can efficiently handle a high volume of inquiries while being interactive, accessible, and responsive to basic issues with an adequate dataset. Our proposal significantly improves student experience, reduces the burden on faculty and staff, and delivers enhanced support services. Our specific contributions are as follows.

- We designed a modular architecture that decouples core components, enabling simple model swaps and integration of new data sources.
- We implemented an asynchronous indexing pipeline that reduces knowledge update latency by 6x.
- We created a 2834-document corpus from diverse institutional sources, enabling the AI assistant.
- We performed a preliminary human evaluation study demonstrating the system’s effectiveness in answering user queries.

2 Literature Review

2.1 Language Models and the RAG Paradigm

2.1.1 Evolution of Large Language Models

The development of modern language models is rooted in the Transformer architecture, which was introduced as a new method for sequence processing and eschewed recurrent structures in favor of attention mechanisms [24]. This architecture became the foundation for subsequent models like BERT, which demonstrated the power of pre-training on a large text corpus and then fine-tuning for specific tasks, significantly advancing the field of natural language understanding [4]. This progression culminated in highly capable generative models, such as GPT-3, which showcased remarkable few-shot learning abilities and the potential to serve as general-purpose knowledge engines [2]. These models, however, are trained on fixed datasets, which introduces inherent limitations for applications requiring current or specialized information.

2.1.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) incorporates the use of a large language model with a text embedding model and an external knowledge base [9]. This approach can bypass knowledge staleness and hallucination by conditioning generated output tokens to relevant, trusted, external knowledge retrieved at in real time. The implementation of a RAG system relies on two main technical models: modern vectorization techniques for effective retrieval and causal language models for fluent generation.

The retrieval component is powered by **vectorization**, the process of converting text into dense numerical representations that capture semantic meaning. Unlike lexical methods such as TF-IDF or BM25, which produce sparse vectors lacking deep contextual understanding, modern approaches leverage contextual embeddings from transformer-based models. These embeddings map text, such as a document chunk or a user query, to a high-dimensional vector where semantic similarity corresponds to spatial proximity. This allows a system to perform a semantic search, finding documents that are conceptually related to the user’s query, even if they do not share exact keywords. The quality of these embeddings is therefore critical for the RAG system’s ability to identify and retrieve the correct knowledge.

Once relevant information is retrieved, it is passed to a **causal language model** (also known as an autoregressive model) to generate the final answer. Models like the Generative Pre-trained Transformer (GPT) family are designed to predict the next token in a sequence based exclusively on the preceding context [2]. This autoregressive property can be expressed as maximizing the probability of the entire sequence, $p(x_1, \dots, x_T) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1})$ [14]. By providing the retrieved document chunks as part of the input context, the model’s generation is "grounded" in factual information, compelling it

to synthesize an answer that is both fluent and faithful to the supplied sources. This combination of semantic retrieval and contextual generation forms the foundation of modern RAG systems, directly addressing the inherent limitations of standalone LLMs.

2.2 Institutional Knowledge Management Systems

The application of RAG and similar AI-driven approaches for managing institutional knowledge has been explored in various contexts, from universities to large enterprises. However, we will focus on universities as there is no perfect analogue for a student-like user in other settings.

2.2.1 University Chatbots

Several academic studies have documented the implementation of RAG-based systems in university settings.

- Rahman et al. developed an AI-powered chatbot designed to serve as a mentor for students at BRAC University [20]. Their system features a data ingestion pipeline that processes information from diverse sources including CSV files and university webpages, to automatically generate a question-answer dataset with human curation. The chatbot utilizes a hybrid retrieval approach combining BM25 lexical ranking with ChromaDB semantic retrieval, and employs Meta’s **LLaMA-3.3-70B** model to generate conversational responses. Their evaluation showed promising results with a BERTScore of 0.831 and a METEOR score of 0.809, indicating semantically relevant responses. The system demonstrated efficient data processing, with update times of 106.82 seconds compared to 368.62 seconds for new data ingestion.
- Neupane et al. introduced BarkPlug v.2, a comprehensive chatbot system built using RAG pipelines for Mississippi State University [15]. Their system features a two-phase architecture consisting of context retrieval and completion components. For context retrieval, they utilized a similarity score threshold mechanism with ChromaDB as their vector database and OpenAI’s **text-embedding-3-large** model for embeddings. For text completion, they employed OpenAI’s **gpt-3.5-turbo** model. The system was evaluated using both quantitative measures (RAGAS framework) and subjective satisfaction surveys (System Usability Scale). Their system demonstrated impressive quantitative performance with a mean RAGAS score of 0.96, indicating strong performance in both retrieval and generation phases. The usability assessment yielded a SUS score of 67.75, suggesting satisfactory user experience with room for improvement.

2.3 Limitations of Existing Approaches

An analysis of existing systems reveals specific weaknesses that our work aims to address. The limitations are not abstract issues but are concrete challenges found in current implementations:

- The AI-powered chatbot for BRAC University, while demonstrating semantic relevance in its responses, suffers from several practical limitations. It exhibits entity resolution problems, failing to consistently match full names with their abbreviations, which leads to unreliable information retrieval. The system’s dataset is also

restricted to the CSE department, limiting its generalizability to the entire student population. Furthermore, its retrieval accuracy varies with the user’s query phrasing [20].

- BarkPlug v.2, despite its impressive RAGAS score of 0.96, has several notable limitations that affect its accessibility and reliability. The retrieval system sometimes fails to provide accurate results, occasionally producing incorrect information or hallucinations. Additionally, the system is limited by a maximum of 4096 output tokens and a context window of 16k, which can hinder its ability to maintain full conversation context [15].

2.4 Contributions of IUB-RAG-LLM

The IUB-RAG-LLM system is designed specifically to address these limitations. Our work introduces an asynchronous indexing pipeline that directly tackles knowledge staleness by reducing update latency by 6x. Furthermore, the system is built on a modular architecture, which provides the flexibility to easily replace components and integrate new data sources, overcoming the issue of architectural rigidity and dependency on proprietary ecosystems. By focusing on these core engineering challenges, IUB-RAG-LLM advances the state of the art in creating responsive, maintainable, and effective institutional knowledge management systems.

3 System Design

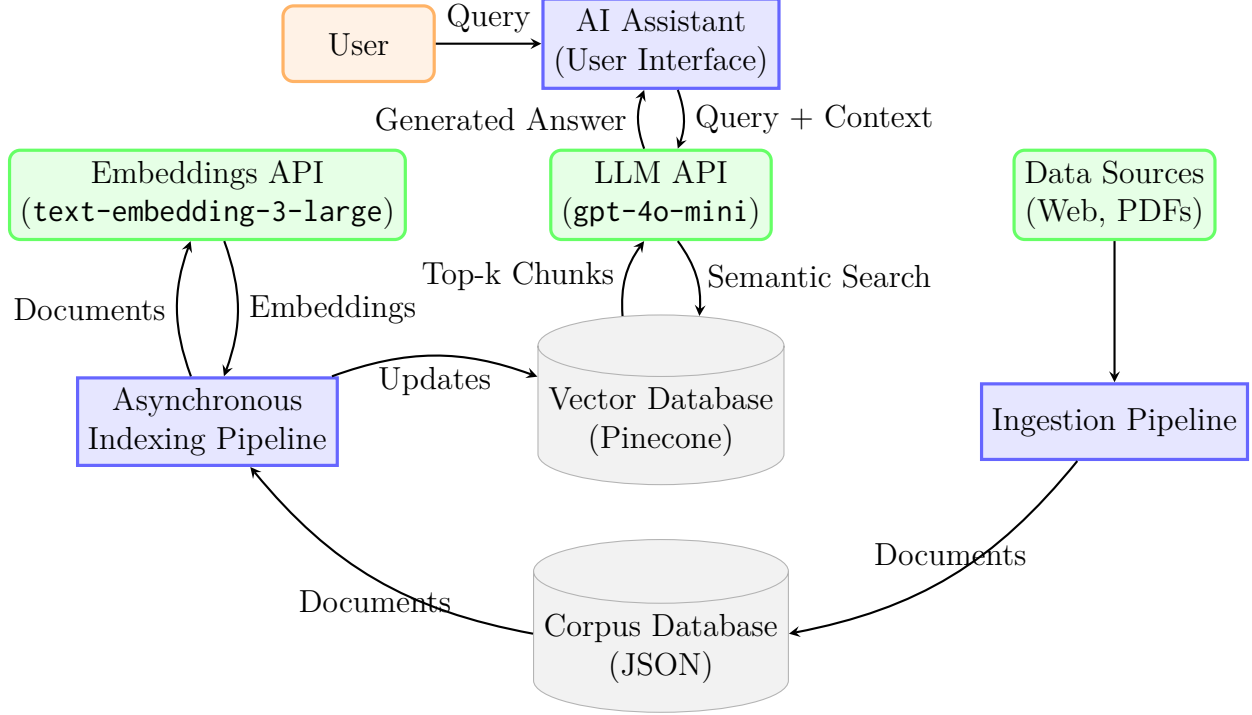


Figure 1: High-level architecture of the IUB-RAG-LLM system, showing the flow from data sources to the end user.

3.1 System Overview

The system design for IUB-RAG-LLM establishes a modular, extensible platform for institutional knowledge management. The architecture is composed of several interconnected components which work together to provide users with accurate, up-to-date information through a conversational interface. The system utilizes a retrieval-augmented generation approach, which combines the in-context learning capability of large language models with a specialized knowledge base of institutional data. Our system is based primarily on two parametric models from OpenAI: **gpt-4o-mini** for text generation [16], and **text-embedding-3-large** for semantic embeddings [17].

3.2 Data Ingestion Pipeline

The data ingestion pipeline is responsible for collecting information from various institutional sources and preparing it for indexing. This component handles two primary types of data sources:

For web-based sources, we implemented a sophisticated web scraping system capable of handling modern web architectures. The system combines Scrapy, Playwright, and modular logic for each source together to implement a modern web scraper, which integrates to provide data in a consistent format to the ingestion pipeline.

Playwright, a Node.js library, through Python bindings provides a high-level API to control Chromium over the DevTools Protocol. This allows us to render JavaScript-heavy single-page applications (SPAs), handle client-side rendering (CSR), incremental static regeneration (ISR), provide access to the document-object model (DOM), JavaScript environment, and browser logs, extract content from dynamic web pages, and schedule regular updates to capture the latest information.

Scrapy, a Python library, provides a flexible web crawling framework that supports our concurrent, parallel, and modular system. It is comprised of several parts: a crawler engine, basic spiders, middleware, and feeds. This enables us to integrate into our data ingestion pipeline for the JSON corpus database as a feed and perform queue management with retry, wait, and skip functionality. We also implement our own `PlaywrightCrawlerSpider` class, which performs page interactions to ensure all elements are dynamically loaded,¹ renders multiple times through a specialized debouncer mechanism to ensure complete rendering before page extraction, and skips downloading resources such as images, stylesheets, and other types that are not necessary to increase throughput.

The web scraper navigates the institution’s website systematically, following internal links to discover and capture relevant content. We used two spiders to perform our experiment, one which extracts text content while preserving metadata such as URLs, page titles, and last modification dates, and the other which downloads all documents found for further extraction.

3.2.2 Document Data Source

For document-based sources like PDFs, we implemented a document parser using the PyPDF2 library in order to extract text content from PDF files, preserve simple document structures, while splitting the document at page boundaries, and maintain metadata including document titles, source information, and timestamps.

The ingested data is stored in a JSON-based corpus database, which serves as the raw material for the indexing pipeline.

3.3 Asynchronous Indexing Pipeline

The asynchronous indexing pipeline is a core innovation of our system, designed to efficiently process large volumes of documents and keep the knowledge base up-to-date. This pipeline operates independently of the user-facing components, allowing for continuous updates without impacting system responsiveness.

3.3.1 Document Chunking

The pipeline begins by loading documents from the JSON corpus database and then splitting them into chunks with recursive character text splitter. This process uses a target chunk size of 1000 tokens with 200-token overlap and preserves semantic context across chunk boundaries.

¹In SPA frameworks, some elements are not loaded until scrolled into view or clicked.

Through this process, documents are broken down into smaller, semantically coherent chunks. This is performed both in order to increase retrieval precision by allowing queries to match parts of documents, and to fit large documents into the embedding model’s context window size. Each chunk is tagged with metadata, such as relationships to source documents and other nodes.

3.3.2 Vectorization Process

Each text chunk is converted into a high-dimensional vector representation using OpenAI’s `text-embedding-3-large` model. This model produces 3072-dimensional vectors that capture the semantic meaning of the text. The vectorization process sends chunks to the OpenAI API in large batches (e.g., $n = 1000$) for efficiency, handles retry logic, rate limiting, and API quotas gracefully, and the resultant vectors, original text, and metadata for retrieval are then stored in the vector database.

3.4 Vector Database

The vectorized chunks are stored in a Pinecone vector database, which provides efficient similarity search capabilities. Pinecone offers Approximate Nearest Neighbor (ANN) indexing for fast similarity search, managed infrastructure that scales automatically, and support for metadata filtering alongside vector search.

Our implementation uses cosine similarity as the distance metric and configures Pinecone’s index to optimize for both recall and performance.

3.4.1 Asynchronous Processing

The key innovation of our indexing pipeline is its asynchronous architecture, which defines a pipeline and its functions using our `AsyncGenerator` and `AsyncGeneratorChain` classes. These classes facilitate a producer-consumer pattern using queues, which allows our indexing pipeline to process multiple documents, embedding requests, and database operations simultaneously.

This design allows the pipeline to achieve a 6x improvement in indexing speed compared to synchronous approaches, significantly reducing knowledge update latency.

3.5 Query Processing and Response Generation

When a user submits a query through the AI Assistant interface, the system processes it through several stages:

3.5.1 Query Rewriting

Rephrase the user’s query with OpenAI’s `gpt-4o-mini` model by using the information from their conversational history to convert it to a standalone query, in order to contextualize the query embedding vector for semantic search. If the conversational history is empty, this step is skipped.

The query vector is used to perform a similarity search against the Pinecone vector database. Our implementation converts the user’s query into a vector representation using the same embedding model (`text-embedding-3-large`) used for indexing. This ensures that the query embedding vector exists in the same semantic space as the document vectors. It then retrieves the top- k most relevant document chunks based on cosine similarity, applies Maximum Marginal Relevance (MMR) to reduce redundancy and selects the top chunks after MMR processing, and uses a parent retriever to replace retrieved document chunks with the unique set of their associated source documents.

3.5.3 Response Generation

The retrieved context and original query are formatted into a prompt template and sent to OpenAI’s `gpt-4o-mini` model for response generation. The prompt template clearly defines the assistant’s role and purpose, with instructions for handling unknown information, and citation of source documents. It also performs query-rewriting from conversational history, in order to maximize the context window tokens available for retrieved documents.

The generated response is then presented to the user through the AI Assistant interface.

3.6 User Interface

The AI Assistant provides a user-friendly conversational interface implemented with Streamlit that provides a simple, intuitive interface for all user types, supports natural language queries and responses, renders rich text formatting such as bold/italics, lists, and links, and maintains conversation history for context.

The interface is implemented as a web application using modern frontend technologies, ensuring accessibility across devices and platforms.

3.7 Modular Architecture

A key design principle of the IUB-RAG-LLM system is modularity, which enables easy replacement of components (e.g., swapping embedding models or vector databases: both of which would otherwise be tightly-coupled in traditional RAG systems), integration of new data sources without architectural changes (through clear separation of concerns), independent configuration, reduced technical debt and improved maintainability of different system components.

This modularity is achieved through well-defined interfaces between components and the use of design patterns such as strategy and dependency injection. In particular, we used the strategy pattern comprehensively to instantiate modules according to the system configuration, and dependency injection to apply our custom integrations with Scrapy.

We used LangChain as an abstraction layer to interface with APIs for LLMs and embedding models, vector databases, and others. In some instances, we modified the standard classes through subclassing in order to suit our requirements. One such example is the implementation of the parent retriever class, which returns the source document of a retrieved chunk from the vector database.

We used Pydantic, a powerful data validation library, through all parts of the system. Common use-cases were centralized within each system component for simple and

consistent management, such as for data models and system configuration.

Firstly, for data models, it was used to represent all instances of data with a consistent type system, dynamic initialization, and custom serialization. This provides optimal performance for handling large amounts of data because Pydantic Core is implemented in Rust, an efficient systems programming language. The library was used to maintain strict data format consistency between various frameworks, libraries, and our own code base. Nested data models were applied in order to maintain the DRY principle.

Secondly, for system configuration, it was used to create a system configuration class hierarchy, which allows settings defined at the module-level to be overridden globally or at run-time. Both environmental variables and command-line arguments are dynamically-generated, which enables settings field definitions at the module-level to be easily configured without any additional changes.

Overall, these adaptations made it possible to decouple the system components, data model, and system configuration for the modular architecture.

4 Corpus Dataset

4.1 Motivation & Purpose

The primary motivation for creating a dedicated institutional dataset is to directly address the problem of knowledge staleness. General-purpose models and public datasets lack the specific, context-rich information required to answer queries unique to the university community. To build an effective AI assistant for IUB, we needed a high-quality, up-to-date, and comprehensive corpus of institutional knowledge. The purpose of this chapter is to detail the creation, structure, and features of the IUB-RAG-LLM Corpus, which serves as the external knowledge base for our system.

4.2 Source of Data Samples

Data for the IUB-RAG-LLM Corpus was sourced from a variety of official and publicly available institutional channels to ensure a comprehensive representation of university knowledge. The primary sources include:

- The main university website (www.iub.ac.bd, Next.js, ISR), encompassing departmental pages, academic program descriptions, and official news articles.
- The official university student handbook (PDF version, Spring 2025), which contains critical information on policies, procedures, and academic regulations.

4.3 Structure and Features of Dataset

The raw data from the sources mentioned above underwent a structured processing pipeline to transform it into a format suitable for semantic search and retrieval.

4.3.1 Chunking Strategy

To manage the context window limitations of the LLM and text embedding models, documents were segmented into smaller, semantically coherent chunks. We employed a recursive character text splitter with a target chunk size of 1000 tokens and a 200-token overlap. This strategy ensures that semantic context is preserved across chunk boundaries while keeping the chunks concise enough for effective embedding and retrieval.

4.3.2 Embedding and Vectorization

Each text chunk was converted into a high-dimensional vector representation using OpenAI's `text-embedding-3-large` model. This model produces a 3072-dimensional vector

that captures the semantic meaning of the text. These vectors, along with their corresponding source text and metadata (e.g., source URL, document title), were then stored in a Pinecone vector database, which provides an efficient Approximate Nearest Neighbor (ANN) index for fast similarity search.

4.3.3 Corpus Statistics

The final IUB-RAG-LLM Corpus consists of 2835 unique source documents. After applying the chunking strategy, this resulted in 7798 individual text chunks and their corresponding embedding vectors, which were indexed in the Pinecone database.

5 Experimental Design

5.1 System Configuration

The IUB-RAG-LLM system was configured with the following key components to facilitate both the asynchronous indexing pipeline and the end-user inference process.

- **Chunking Parameters:** Prior to embedding, we split the documents using a recursive-character splitter, with chunk size 1000, and chunk overlap 200.
- **Text Embedding Model:** For generating text embeddings, we utilized OpenAI’s `text-embedding-3-large` model via its API.
- **Generative Model:** For generating final answers, we utilized OpenAI’s `gpt-4o-mini` model via its API with temperature at 0.
- **Retrieval Parameters:** During inference, the top-80 most relevant document chunks ($k_{\text{fetch}} = 80$) were fetched from the Pinecone database based on cosine similarity, in maximum marginal relevance (MMR) mode with $\lambda = 0.5$ to keep the top-16 ($k = 16$) document chunks, which were then replaced with the full documents via the `parent` retriever.
- **Prompt Template:** A simple in-context learning template was used to structure the input for the LLM:

```
You are an assistant for question-answering tasks at the Independent
University, Bangladesh (IUB). Use the following pieces of retrieved
context to answer the question. If you don't know the answer, just say
that you don't know. Do not mention the context except only to cite
the sources of relevant documents.
```

```
<context>
{context}
</context>
```

5.2 Evaluation Methodology

Given the project’s focus on demonstrating a functional system, our evaluation was twofold: a performance benchmark of the data pipeline and a human evaluation of the final system’s output.

To assess the quality and usefulness of the AI assistant’s responses, we conducted a human evaluation survey.

- **Participants:** The study involved 3 student volunteers from IUB, who were representative of the system’s target user base.
- **Task:** Each participant was presented with a set of 20 diverse questions about university life, policies, and procedures. For each question, they were shown the answer generated by the IUB-RAG-LLM system.
- **Metrics:** Participants were asked to rate each answer on a 5-point Likert scale (1=Very Poor, 5=Excellent) across two dimensions: Accuracy of the factual information in the answer
 - **Correctness:** How accurate was the factual information in the answer provided?
 - **Completeness:** How complete were the details in the answer to fully address the question?
- **Qualitative Feedback:** An optional text box was provided for participants to leave general comments or suggestions for improvement.
- **Instructions:** A set of instructions on scoring was given for participants to maintain a standard basis of evaluation:

You have been asked to evaluate the test data from “IUB-RAG-LLM,” an AI assistant for students at the university.

You will be shown 2 categories, each containing 10 question–answer pairs, making a total of 20 pairs.

You will be asked to:

1. Rate the correctness of each answer, from 1 to 5.

Definition: Accuracy of the factual information in the answer.

Sample question: What is the grading system at IUB?

- 1/5 Incorrect: I don’t know.
- 3/5 Somewhat Correct: Letter grades. A+: 4.00, A: 3.70, ...
- 5/5 Fully Correct: Letter grades. A: 4.00, A-: 3.70, ...

2. Rate the completeness of each answer, from 1 to 5.

Definition: Whether the response provides all necessary details to fully address the question.

Sample question: What are the best places to study on campus?

- 1/5 Incomplete: I don’t know.
- 3/5 Somewhat Complete: Students can study at the library.

- 5/5 Fully Complete: Students can study at CRIB, ELRC, Library, Common Areas, or Cafes. Location, timing, and rules are as follows...

3. Identify the issues or gaps in the answer.

Definition: What information was incorrect or missing. Brief remarks for simple clarity only.

For each answer, you need to verify if the pieces of information are correct. This may mean checking:

- Phone numbers,
- Email addresses,
- Official websites (i.e. <https://www.iub.ac.bd/>), or
- Unwritten information (i.e. informal knowledge).

Please be aware of:

- **Incompleteness** – When presenting lists, are all points mentioned? For example, mentioning only two events instead of all of them.
- **Misattribution** – Is the information mistakenly attributed to the wrong person or group? For example, crediting a paper to an unexpected author.

Finally, to generate answers in batch for each question of the human evaluation survey, we invoked the inference chain directly and bypassed the user interface.

5.2.2 System Performance Benchmarking

To quantify the improvement gained from our asynchronous pipeline, we measured the total time required for a full knowledge base update.

- **Asynchronous Implementation:** The final asynchronous pipeline, which processes documents concurrently, was timed through logging timestamps.
- **Measurement:** The total wall-clock time from the start of the ingestion process to the final insertion of all vectors into the Pinecone database was recorded for both methods. The test was performed on the full corpus of 2834 documents.

6 Results and Analysis

6.1 Human Evaluation Results

The human evaluation provided positive feedback on the overall quality of the IUB-RAG-LLM system’s responses. The average scores across all 60 evaluated answers were as follows:

- Correctness: 3.57 / 5.00
- Completeness: 3.67 / 5.00

These results find that the system was generally perceived as clear and helpful, with a slightly lower but still strong score on factual accuracy, which suggests room for improvement in data source quality or retrieval precision.

6.2 Sample Question–Answer Analysis

To illustrate the system’s performance across different types of queries, we present representative examples of good, average, and poor responses.

6.2.1 Example of Good Performance

Question: What scholarships are available for IUB students?

System Response: IUB offers a variety of scholarships and financial aid options for students, including:

1. Scholarships based on Admissions Test & Previous Academic Results: 30%-100% coverage.
2. IUB Merit Scholarships based on Semester Results: 30%-100% coverage.
3. Need-based Aid for Returning Students: 20%-100% coverage.
4. For Children of Freedom Fighters: Up to 100% coverage.
5. For achievements in Extra-Curricular Activities: Up to 100% (national level achievements considered).
6. IUB-Saima Hall: A dormitory for female students funded by IUB & Shaheed Khalek & Major Salek Bir Uttam Trust, offering 100% coverage (subject to conditions and availability).

7. Aid for outstanding students from economically challenged areas: Up to 100% coverage.
8. Scholarship for Female Students: 10% coverage.
9. Sibling Discount: 30% coverage.
10. IUB employees and their dependents: 50%-100% coverage.
11. Offspring of IUB Alumni: 10% coverage.
12. IUB ARROWS: Up to 100% coverage (annual selection basis).

These scholarships aim to ensure that qualified students have access to financial support to pursue their education at IUB. (Sources: <https://iub.ac.bd/about/iub-at-a-glance>, <https://iub.ac.bd/campus-life/administrative-and-support-services/office-of-admissions-financial-aid-and-international-relations>)

Evaluation: Correctness: 5/5, Completeness: 4/5

Analysis: This response demonstrates excellent performance with accurate, comprehensive information directly addressing the user's question. The system correctly retrieved and synthesized information from multiple sources to provide a complete answer.

6.2.2 Example of Average Performance

Question: Where can I find career counseling and job placement services?

System Response: You can find career counseling and job placement services at the Career Guidance Placement and Alumni Relations (CGP&AR) office at Independent University, Bangladesh (IUB). They organize career and networking events, maintain a network with various organizations for internships and job opportunities, and facilitate events like career fairs and networking sessions with alumni. You can contact them for more information or assistance regarding your career development.

For more details, you can visit their office located at Room#1011, Ground Floor, Academic Block, IUB Campus, or reach out via email at cgp@iub.edu.bd.

Evaluation: Correctness: 3/5, Completeness: 3/5

Analysis: While factually correct, this response lacks specific details about career events, exact deadlines, and where to find the application forms. The system provided general information that applies to many job opportunities but missed specific details available in the source documents.

Question: How do I set up my university email account?

System Response: To set up your university email account at Independent University, Bangladesh (IUB), you can open your own email account at the Central Information and Technology Services (CITS). It is advisable to visit the CITS office or their website for specific instructions and assistance regarding the setup process.

Evaluation: Correctness: 2/5, Completeness: 2/5

Analysis: This response demonstrates a failure to retrieve relevant information, despite the system providing generic information which doesn't address the specific concerns, which includes any contact information for IT or links to the relevant website.

6.3 Qualitative Analysis of User Feedback

Analysis of the qualitative comments revealed common themes. Criticisms often pointed to answers that were too generic or failed to synthesize information from multiple retrieved chunks to fully answer complex, multi-part questions.

6.4 Qualitative Analysis of System

Inspection of the queries, retrieved documents, and the generated answer showed two anecdotal findings. Firstly, we observed a significant correlation between low retrieval precision and low completeness scores. As the LLM does not have detailed internal knowledge of the institution, it is highly reliant on the external knowledge provided by retrieved documents. Secondly, we could not find any significant instances of hallucination within this limited test set, but we acknowledge they remain a possibility in broader usage. Overall, we conclude that there are significant opportunities to improve the system's knowledge retrieval, while text generation seems to be optimally performant for our application.

6.5 System Performance Results

The performance benchmarking confirmed the significant efficiency gains of the asynchronous data pipeline.

- The data ingestion pipeline took 1 hour, 42 minutes, and 58 seconds to fully ingest the web data source.
- The synchronous baseline indexing pipeline took 52 minutes, and 22 seconds to fully index the entire corpus.
- The asynchronous pipeline completed the same task in 8 minutes, and 27 seconds.
- This represents a 6x improvement in indexing speed, demonstrating the effectiveness of the concurrent design in reducing knowledge update latency and enabling a more responsive knowledge base.

7 Conclusion and Future Works

7.1 Conclusion

This project addressed the challenge of disseminating institutional knowledge by developing IUB-RAG-LLM, an extensible RAG-based system. We identified key limitations in traditional RAG systems, namely knowledge staleness, ingestion and indexing inefficiency, and architectural rigidity. Our solution introduced a novel asynchronous data pipeline with a dynamic JS-rendering scraper, which successfully reduced knowledge update latency by over 6x. A preliminary human evaluation study suggests that the resulting AI assistant provides helpful and clear answers to user queries, validating the system’s potential to reduce the burden on faculty and staff while improving the student experience.

7.2 Limitations

The current work has several limitations. The human evaluation was conducted on a small and relatively homogenous group of participants, which may not generalize to the entire university community. The system’s knowledge is limited to the specific sources it was trained on and does not include real-time data. Furthermore, the system relies on proprietary APIs (OpenAI, Pinecone), which introduces dependencies and potential costs.

7.3 Ethical Considerations

The system is designed to provide information based on official sources, but the risk of generating incorrect or misleading answers, however small, still exists. This could have negative consequences for users making academic or administrative decisions. It is crucial that the system is positioned as an assistive tool and not a replacement for official human consultation. Transparency about the system’s limitations and data sources is an ethical imperative.

7.4 Impact on the Society

By providing instant, 24/7 access to institutional knowledge, the system can significantly enhance the efficiency and productivity of the university community. It empowers students to find information independently, freeing up faculty and staff to focus on more complex, high-value tasks. This contributes to a more supportive and effective educational environment.

7.4.1 Sustainability of the work

The modular architecture of the IUB-RAG-LLM framework ensures its long-term sustainability. Components can be easily swapped as better models or databases become available. The automated nature of the pipeline reduces the manual effort required for maintenance, making the system a viable long-term solution for the university.

7.5 Future Plan

Future work will focus on addressing the current limitations. We plan to expand the human evaluation to a larger, more diverse group of users, including faculty and staff. We will investigate integrating real-time data sources, such as university social media feeds or event calendars. Finally, we will explore fine-tuning a smaller, open-source LLM on our corpus to reduce dependency on external APIs and potentially improve performance on institution-specific jargon and context.

Bibliography

- [1] BENGIO, Y., DUCHARME, R., VINCENT, P., AND JANVIN, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3 (Mar. 2003), 1137–1155.
- [2] BROWN, T. B., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AGARWAL, S., HERBERT-VOSS, A., KRUEGER, G., HENIGHAN, T., CHILD, R., RAMESH, A., ZIEGLER, D. M., WU, J., WINTER, C., HESSE, C., CHEN, M., SIGLER, E., LITWIN, M., GRAY, S., CHESSE, B., CLARK, J., BERNER, C., MCCANDLISH, S., RADFORD, A., SUTSKEVER, I., AND AMODEI, D. Language Models are Few-Shot Learners, July 2020.
- [3] DAI, D., SUN, Y., DONG, L., HAO, Y., MA, S., SUI, Z., AND WEI, F. Why Can GPT Learn In-Context? Language Models Implicitly Perform Gradient Descent as Meta-Optimizers, May 2023.
- [4] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019.
- [5] FORMAL, T., LASSANCE, C., PIWOWARSKI, B., AND CLINCHANT, S. SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval, Sept. 2021.
- [6] FORMAL, T., PIWOWARSKI, B., AND CLINCHANT, S. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking, July 2021.
- [7] HAN, Y., LIU, C., AND WANG, P. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge, Oct. 2023.
- [8] KAHNEMAN, D., AND TVERSKY, A. Prospect Theory: An Analysis of Decision under Risk. *Econometrica* 47, 2 (1979), 263–291.
- [9] LEWIS, P., PEREZ, E., PIKTUS, A., PETRONI, F., KARPUKHIN, V., GOYAL, N., KÜTTLER, H., LEWIS, M., YIH, W.-T., ROCKTÄSCHEL, T., RIEDEL, S., AND KIELA, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Apr. 2021.
- [10] LI, W., ZHANG, Y., SUN, Y., WANG, W., ZHANG, W., AND LIN, X. Approximate Nearest Neighbor Search on High Dimensional Data — Experiments, Analyses, and Improvement (v1.0), Oct. 2016.
- [11] LIU, F. *Vector Database 101: Everything You Need to Know*. Sept. 2022.
- [12] MALKOV, Y. A., AND YASHUNIN, D. A. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs, Aug. 2018.

- [13] MCCANDLISH, S., KAPLAN, J., AMODEI, D., AND TEAM, O. D. An Empirical Model of Large-Batch Training.
- [14] MIIKKULAINEN, R., AND DYER, M. G. Natural Language Processing With Modular Pdp Networks and Distributed Lexicon. *Cognitive Science* 15, 3 (July 1991), 343–399.
- [15] NEUPANE, S., HOSSAIN, E., KEITH, J., TRIPATHI, H., GHIASI, F., GOLILARZ, N. A., AMIRLATIFI, A., MITTAL, S., AND RAHIMI, S. From Questions to Insightful Answers: Building an Informed Chatbot for University Resources, 2024.
- [16] OPENAI. GPT-4o System Card, Oct. 2024.
- [17] OPENAI. New Embedding Models and API Updates, Jan. 2024.
- [18] OUYANG, L., WU, J., JIANG, X., ALMEIDA, D., WAINWRIGHT, C. L., MISHKIN, P., ZHANG, C., AGARWAL, S., SLAMA, K., RAY, A., SCHULMAN, J., HILTON, J., KELTON, F., MILLER, L., SIMENS, M., ASKELL, A., WELINDER, P., CHRISTIANO, P., LEIKE, J., AND LOWE, R. Training Language Models to Follow Instructions with Human Feedback, Mar. 2022.
- [19] RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., AND SUTSKEVER, I. Language Models are Unsupervised Multitask Learners.
- [20] RAHMAN, M., ABEDIN, M., ABIR, M. Z., ANSARI, F. I., REZA, A., SADEQUE, F. Y., AND FARHAN, N. Transforming Mentorship: An AI Powered Chatbot Approach to University Guidance, 2025.
- [21] ROBERTS, A., RAFFEL, C., AND SHAZEER, N. How Much Knowledge Can You Pack Into the Parameters of a Language Model?, Oct. 2020.
- [22] SOLDAINI, L., KINNEY, R., BHAGIA, A., SCHWENK, D., ATKINSON, D., AUTHUR, R., BOGIN, B., CHANDU, K., DUMAS, J., ELAZAR, Y., HOFMANN, V., JHA, A. H., KUMAR, S., LUCY, L., LYU, X., LAMBERT, N., MAGNUSSON, I., MORRISON, J., MUENNIGHOFF, N., NAIK, A., NAM, C., PETERS, M. E., RAVICHANDER, A., RICHARDSON, K., SHEN, Z., STRUBELL, E., SUBRAMANI, N., TAFJORD, O., WALSH, P., ZETTLEMOYER, L., SMITH, N. A., HAJISHIRZI, H., BELTAGY, I., GROENEVELD, D., DODGE, J., AND LO, K. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research, Jan. 2024.
- [23] STECK, H., EKANADHAM, C., AND KALLUS, N. Is Cosine-Similarity of Embeddings Really About Similarity?, Mar. 2024.
- [24] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention Is All You Need.
- [25] WEI, J., WANG, X., SCHUURMANS, D., BOSMA, M., ICHTER, B., XIA, F., CHI, E., LE, Q., AND ZHOU, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Jan. 2023.
- [26] ZHANG, C. Optimizing RAG Applications: A Guide to Methodologies, Metrics, and Evaluation Tools for Enhanced Reliability, Dec. 2023.

- [27] ZHANG, Y., LI, Y., CUI, L., CAI, D., LIU, L., FU, T., HUANG, X., ZHAO, E., ZHANG, Y., CHEN, Y., WANG, L., LUU, A. T., BI, W., SHI, F., AND SHI, S. Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *Computational Linguistics* (Sept. 2023), 1–46.