

# The Document Analysis Hub: A System for Thematic Q&A and Summarization of Large Documents

Jie Chen

JCHEN24@UBishops.ca

Ruiyan Zhang

RZHANG23@UBishops.ca

August 5, 2025

## Abstract

This report details the design and implementation of the Document Analysis Hub, a web application for analyzing large academic documents. The application features two independent tools: an intelligent Question-Answering (QA) system and a chapter-based Summarizer.

The QA engine uses a specialized, chapter-based Retrieval-Augmented Generation (RAG) pipeline. It first identifies the most relevant chapter for a query, then extracts context to generate a precise answer. This process is further refined by a multi-model consensus mechanism for enhanced reliability.

The Summarizer, built for speed, uses a single high-performing LLM and a Map-Reduce strategy to handle long chapters.

Powered by the Groq API and Streamlit, the system is founded on data-driven model selection and a modular architecture. The final product is a robust and user-centric tool that offers an effective way to interact with structured documents.

# Chapter 1

## Introduction

In the modern digital era, the rapid proliferation of text has created an unprecedented volume of unstructured data, posing a significant challenge for effective information access. Natural Language Processing (NLP) has emerged as a critical field dedicated to enabling machines to understand and generate human language. This chapter provides a foundational overview of the key domains relevant to this project: Text Processing (TP), Text Summarization (TS), and Question Answering (QA). These sections highlight the technical challenges that directly motivate the development of the **Document Analysis Hub**, our proposed solution.

### 1.1 Background on Text Processing

**Text Processing (TP)** encompasses the techniques used to convert raw, unstructured text into a structured, machine-readable format. It is the foundational step for nearly all NLP applications, as computers cannot interpret human language in its natural form [9].

#### Traditional Techniques

Early TP methods relied on statistical techniques to create numerical representations of text. Foundational methods included:

- **Tokenization:** Splitting text into smaller units like words or sentences.
- **Bag-of-Words (BoW):** Representing a document as a collection of its word frequencies, disregarding grammar and word order [1].
- **Term Frequency-Inverse Document Frequency (TF-IDF):** A more advanced statistical measure that gives higher weight to words that are frequent in a single document but rare across all documents, thus highlighting important terms [2].

While effective for basic tasks like document classification, these methods suffer from a major limitation: they cannot capture the semantic meaning or context of words. They treat "bank" (a financial institution) and "bank" (a river edge) as the same entity.

## Modern Techniques: The Rise of Embeddings

To overcome the limitations of traditional methods, the field shifted towards learning vector representations of words, known as **word embeddings**.

- **Static Embeddings:** Models like **Word2Vec** learn a single, dense vector for each word that captures its semantic relationships with other words. This allows for reasoning such as the famous analogy: "king" - "man" + "woman" = "queen" [3].
- **Contextual Embeddings:** The current state-of-the-art is dominated by **Transformer-based models** like BERT and GPT [7]. These models generate "contextual" embeddings, meaning the vector for a word changes depending on the sentence it appears in. This allows them to understand nuance and polysemy, providing a much deeper understanding of language.

Our project leverages these modern contextual models to understand the semantic meaning of both user queries and document content.

## 1.2 Background on Text Summarization

**Text Summarization (TS)** is the task of automatically creating a short, fluent, and accurate summary of a longer text. Its primary goal is to distill the essential information from a source document, addressing the challenge of "information overload" [4].

### Core Approaches to Summarization

There are two fundamental strategies for automatic summarization:

- **Extractive Summarization:** This method identifies and extracts the most important sentences or phrases directly from the source text. The resulting summary is composed of original text, which guarantees factual consistency but can sometimes lack coherence and feel disjointed.
- **Abstractive Summarization:** This more advanced method involves generating new sentences that rephrase the core concepts of the source document, much like a human would. This can produce more fluent and natural-sounding summaries but carries the risk of introducing factual inaccuracies, or "hallucinations."

Our project's summarizer employs an abstractive approach to create high-quality, readable summaries.

## Key Challenges

Building an effective summarization system involves overcoming several technical hurdles:

- **Factuality and Hallucination:** Ensuring that an abstractive summary remains factually grounded in the source text is the most critical challenge.
- **Handling Large Documents:** Many state-of-the-art models have a limited "context window," meaning they cannot process an entire large document at once.

## Modern Solutions

The current state-of-the-art in text summarization is dominated by models based on the **Transformer architecture** [5]. These models, such as T5 and BART, use an encoder-decoder framework to read the source text and then generate a new, abstractive summary.

To address the challenge of large documents, a common strategy is **Map-Reduce**. The large text is split into smaller chunks (the "Map" step), each is summarized individually, and the resulting partial summaries are then combined and synthesized into a final, coherent summary (the "Reduce" step). Our project implements this Map-Reduce strategy to enable the summarization of entire document chapters.

## 1.3 Background on Question Answering (QA)

**Question Answering (QA)** is a field within NLP that aims to build systems capable of automatically providing direct, concise answers to questions posed in natural language. Unlike traditional search engines that return a list of relevant documents, QA systems deliver a specific answer [6].

## The Retriever-Reader Architecture

Most modern QA systems are built on a two-stage architecture:

1. **The Retriever:** This component is responsible for searching a large collection of documents (the "corpus") to find a small set of text passages that are likely to contain the answer to the user's question. This is the "search" part of the process.

2. **The Reader:** This component, typically a powerful Large Language Model (LLM), carefully reads the passages provided by the retriever and extracts or synthesizes the final, precise answer. This is the "comprehension" part of the process.

## Retrieval-Augmented Generation (RAG)

The most effective modern approach for building QA systems is **Retrieval-Augmented Generation (RAG)** [8]. RAG is the formal name for the retriever-reader architecture.

The key insight of RAG is that it separates the knowledge of the system from the reasoning ability of the LLM. Instead of trying to store all the world's facts in its parameters, the LLM is given access to an external knowledge source (like a collection of PDFs).

## Key Challenges and Our Solution

The primary challenge in RAG is the quality of the **retrieval** step. If the retriever finds the wrong passages, the reader (LLM) will be unable to answer the question, no matter how powerful it is. This can lead to the model "hallucinating" or stating that the answer cannot be found.

Our project addresses this challenge by implementing a specialized, **chapter-based RAG pipeline**. Instead of searching through the paragraphs of the entire document at once, our retriever first identifies the most relevant *chapter* and then performs a more detailed search for context *only within that chapter*. This hierarchical approach significantly improves the quality and relevance of the retrieved context, leading to more accurate final answers.

# Chapter 2

## Project Objective

The Document Analysis Hub is a web-based application designed to streamline interaction with single, large, structured documents. This project addresses the practical challenge faced by users, such as students and researchers, who often struggle to efficiently navigate lengthy academic or professional texts (e.g., academic calendars, policy handbooks, technical manuals) to find specific information or understand particular themes. Manual navigation of these documents is inefficient and time-consuming.

This project aims to deliver a cohesive dashboard with two independent, specialized tools for analyzing a pre-configured document.

**Expected Outcomes Deliverables:** A fully functional, interactive Streamlit web application, deployed as a public web service.

A Subject-Aware Question Answering tool implementing a custom Retrieval-Augmented Generation (RAG) pipeline. This tool intelligently identifies the most relevant chapter for a user's question and extracts the answer from that specific context.

A Subject-Based Summarization tool that allows users to select a specific chapter and generate a concise summary using a Map-Reduce strategy for long text segments.

The target audience for this system includes students, academic advisors, and researchers who require a fast and reliable way to navigate and understand complex, fixed-format university documents.

What makes this project unique is its hybrid, task-specific architecture. Instead of forcing a single solution, it employs the best-suited technology for each task: a sophisticated, subject-based RAG system for the broad "search" problem, and a robust, focused Map-Reduce system for the deep "understanding" problem. This demonstrates a practical and nuanced approach to building applied AI tools for a well-defined scope.

# Chapter 3

## Methodology

### 3.1 Overall Methodology

The system was designed and developed through a structured, multi-phase process. The overall methodology prioritizes modularity, data-driven model selection, and robust handling of large documents. The approach can be understood as a pipeline comprising two main stages: a one-time **Model Evaluation Phase** to inform our design, and the **Operational Phase** of the final application.

1. **Phase 1 - Model Evaluation and Selection:** Before building the final system, a preliminary quantitative evaluation was conducted on a short, controlled academic document. The purpose of this phase was to objectively measure the performance of several candidate Large Language Models (LLMs) on the core tasks of Summarization (TS) and Question-Answering (QA).
2. **Phase 2 - Final Application Implementation:** Based on the findings from Phase 1, the final application was designed and built as a static analysis tool for a large, pre-configured document. Its workflow involves data pre-processing, user interaction, and backend AI processing to deliver a final result.

### 3.2 Datasets and Pre-processing

#### 3.2.1 Datasets for Training and Testing

As this project utilizes pre-trained Large Language Models, no model training was performed. The datasets were used exclusively for development, evaluation, and as the primary source document for the final application.

- **Phase 1 (Evaluation):** Two short academic papers were used as test documents — a 1-page paper titled "An Apple a Day Keeps the Doctor Away" for both

reference-based and non-reference evaluation, and a 7-page paper on “Attention Is All You Need”\* for non-reference evaluation only. The small size and focused topics of the documents provided a controlled environment for quantitatively comparing LLM performance.

- **Phase 2 (Final Application):** The primary dataset is the **Bishop’s University Academic Calendar 2019-2020**. This 302-page PDF document serves as the single, pre-configured source of information for the final Q&A and Summarization tools.

### 3.2.2 Pre-processing Steps

The core pre-processing step for the final application is the manual creation of a structured Table of Contents (ToC) file.

1. **Manual ToC Creation:** The ToC of the BU Academic Calendar PDF was manually transcribed into a structured `toc.json` file. Each entry in this file maps a chapter title to its corresponding starting page number in the PDF document.
2. **Automated Chapter Extraction:** At runtime, the application uses this `toc.json` file to programmatically split the large PDF into logical chapters in memory. A key function, `load_summary_data` (documented in Appendix A), reads the `toc.json`, opens the PDF, and extracts the complete text for each chapter based on the specified page ranges. This structured data object is then used by both the Q&A and Summarizer tools.

## 3.3 System Architecture

The final application follows a three-part modular architecture, ensuring a clean separation of concerns between data processing, AI logic, and the user interface. This design enhances maintainability and allows for independent development of each component.

The components are:

- **Data Engines (`summarizer_engine.py`, `qa.py`):** Responsible for all data preparation. `summarizer_engine.py` contains the `load_summary_data` function. `qa.py` contains the `find_context_in_relevant_chapter` function for the Q&A tool’s intelligent routing.
- **AI Core (`chat.py`):** This module contains the “pure” backend functions (`get_qa_answer`, `get_summary`) that construct the prompts and interact with the Groq API.
- **Evaluation Engine (`eval.py`):** This module, containing the `run_consensus_evaluation` function, implements the multi-model consensus mechanism.

- **User Interface (app.py):** The main application file, which remains "light" and is responsible only for rendering the UI and orchestrating calls to the other modules.

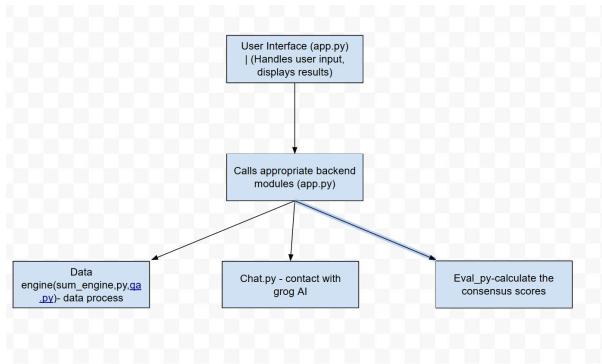


Figure 3.1: system structure

## 3.4 Technical Methodology: Algorithms, Models, and Integration

### 3.4.1 Algorithms and Models Implemented

The system implements several key algorithms and utilizes pre-trained models to handle the challenges of analyzing a large document.

- **Chapter-Based RAG for Q&A:** To handle the large size of the academic calendar, we implemented a custom two-step Retrieval-Augmented Generation (RAG) pipeline.
  1. **Chapter Routing:** The `find_context_in_relevant_chapter` function first identifies the most relevant chapter for a given question. It does this by performing a semantic search (using a `sentence-transformers` model) across all chapter titles. This search is augmented with a keyword-boosting mechanism to improve accuracy for specific topics (e.g., tuition/fee/how much questions).
  2. **Context Retrieval:** Once the correct chapter is identified, a more detailed semantic search is performed on the sub-sections within that chapter to find the most relevant paragraphs to use as context for the final answer.
- **Map-Reduce for Summarization:** To summarize chapters that exceed the LLMs' context window, the `get_summary` function in `chat.py` implements a Map-Reduce strategy. The chapter text is split into smaller sub-chunks, each is summarized individually (the "Map" step), and the partial summaries are then synthesized into a final, coherent summary (the "Reduce" step).
- **Language Models:** The system utilizes three pre-trained Large Language Models available via the Groq API: `gemma2-9b-it`, `llama3-8b-8192`, and `llama3-70b-8192`.

### 3.4.2 Integration of TS and QA Components

The TS and QA components are designed as two independent tools that are seamlessly integrated through a shared backend infrastructure. This architecture allows for code reuse while supporting the unique logic required by each tool.

- **Shared Data Source and Differentiated Logic:** Both tools begin by operating on the same structured data object, which contains the full text of every chapter, created by the `load_summary_data` function (see Appendix A). The integration is handled by the orchestration logic in `app.py`, which directs the user's request to the appropriate data engine:
  - For a **Q&A request**, the system calls the `find_context_in_relevant_chapter` function in `qa.py`. This function intelligently searches across the titles of *all* chapters to find the most relevant context for the final answer.
  - For a **Summarization request**, the system uses the `get_chapter_text` function. This function simply retrieves the full text of the *single* chapter that the user explicitly selected from the dropdown menu.

This demonstrates how the integrated system supports both broad, whole-document searches and narrow, subject-specific tasks.

- **Shared AI and Evaluation Core:** Shared AI Core with Task-Specific Evaluation: A key feature of the system's architecture is a shared AI core that handles all interactions with the backend language models. After a tool retrieves its specific context, it passes the request to this central engine. The engine employs a task-specific evaluation strategy to balance accuracy and performance. For the high-stakes Question Answering task, a full three-model consensus mechanism is activated. The system queries all three LLMs and runs a consensus algorithm to select the most factually accurate and reliable answer. Conversely, for the Summarization task, where speed is a higher priority, the system is optimized to use only a single, pre-selected LLM that demonstrated the best performance in our Phase 1 evaluation. This hybrid approach ensures that the QA tool benefits from maximum reliability, while the Summarizer remains fast and responsive.
- **UI Integration:** In the final application, the two tools are presented in a two-column layout, which uses bordered containers and clear headings.

## 3.5 Evaluation Strategy and Quality Assurance

Our evaluation strategy evolved across the two phases of the project, adapting from a traditional academic metric for the initial model bake-off to a more practical, automated mechanism for the final application.

### 3.5.1 Evaluation Metrics

Two distinct primary metrics were used across the project’s lifecycle:

- **Phase 1 Evaluation Metrics: A Dual Approach**

In our initial model selection on the short document (Phase 1), a dual-metric evaluation strategy was employed to quantitatively assess the distinct capabilities required for summarization and question answering.

**Metric for Text Summarization (ROUGE-L and Pairwise Consensus Score):**

To evaluate the summarization of 1- page task, we used the reference-based metric **ROUGE-L**. This approach was suitable for the controlled environment of Phase 1, as it was feasible to create a single, human-written “golden” reference summary. Each model’s output was then scored against this reference to objectively measure its ability to capture key lexical content. To evaluate the summarization of 7- page task, we used the non-reference similarity way to get the consensus score.

**Metric for Question Answering (Pairwise Consensus Score):**

For the Q&A task, where a single ”golden” answer is often less definitive, we implemented a reference-free metric: the **Pairwise Consensus Score**. This involved calculating the average semantic similarity of each model’s answer to the answers of its peers. This metric was chosen to measure the reliability and consistency of the models’ responses without relying on a predefined correct answer.

**Phase 2 Metric and Quality Assurance Mechanism (Pairwise Consensus Score):**

The final application—designed to handle large and complex documents—creating a “golden” answer or summary for every possible user query is impractical. To address this, we implemented a reference-free evaluation metric known as the Pairwise Consensus Score. This score, which serves as the core of our quality assurance mechanism, is computed using the API Ninjas text similarity service. It offers a robust and automated method to assess the reliability and consistency of generated responses in the Question Answering (QA) module. For the summarization component, we conducted manual evaluations on 15 subjects without the use of any automated tools.

### 3.5.2 How Quality and Accuracy are Ensured

The quality and accuracy of QA in the final application are ensured through the real-time implementation of our **multi-model consensus mechanism**, which is powered by the Pairwise Consensus Score.

For every user request, the `run_consensus_evaluation` function performs the following steps:

1. **Generate Multiple Outputs:** It prompts all three LLMs to generate a response for the same task.

2. **Perform Pairwise Comparison:** It calculates the semantic similarity between each pair of generated responses using the API Ninjas service.
3. **Select the Best Output:** It computes an average consensus score for each response and selects the one that is most centrally-agreed-upon by the model cohort.

For the summarization task, we manual conducted a evaluation across 15 distinct subjects, with a particular focus on large and complex chapters such as "Admission," "University Regulations," and "School of Education." In all cases, the final summary presented to the user was factually accurate, reliable, and well-phrased.

# Chapter 4

## User Manual

This chapter provides a comprehensive guide for the end-users of the Document Analysis Hub. It covers the system requirements for accessing the web service, a step-by-step guide for using its features, and discusses common issues and limitations.

### 4.1 Installation and Setup

#### 4.1.1 For End-Users

- **How to set up and use the system?**

**No installation is required.** The Document Analysis Hub is a fully web-based application. To use it, simply open the following link in a modern web browser:

<https://project1-cs.streamlit.app/>

- **What are the system requirements?**

To use the application, you only need two things:

1. A modern web browser (e.g., Google Chrome, Mozilla Firefox, Safari).
2. A stable internet connection.

#### 4.1.2 For Developers

For technical users who wish to run their own instance of this application, a complete guide to local setup, dependency installation, and deployment is provided in **Appendix B**.

## 4.2 Step-by-Step Guide for Using the Application

Upon opening the application's URL, you are presented with a dashboard containing two independent analysis tools side-by-side. You can use either tool in any order.

### 4.2.1 How to Ask Questions and Get Answers

The Q&A tool, located in the left column, allows you to ask questions about the entire content of the pre-loaded document.

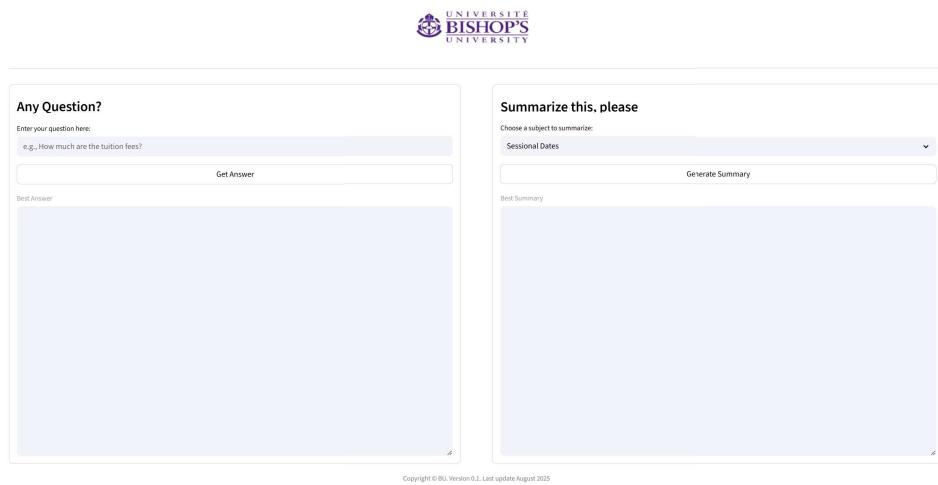


Figure 4.1: application interface

1. Locate the left panel titled "**Any Question?**".
2. Type your question into the text input box.
3. Click the "**Get Answer**" button.

This image is a close-up of the "Any Question?" panel from Figure 4.1. It shows a text input field with the placeholder text "Enter your question here:" and the question "list the rules about graduate students please" typed into it. Below the input field is a "Get Answer" button.

Figure 4.2: ask a question

4. The system will find the most relevant information, and the final answer will appear in the "Answer" text area below. A info box will also appear, indicating the source chapter from which the information was likely drawn.

## Any Question?

Enter your question here:

list the rules about graduate students please

Get Answer

Source: Based on the 'Graduate Studies' chapter.

Best Answer

## Graduate Student Rules at Bishop 's University:

This document outlines rules for graduate students in the Master of Education (MEd) and Master of Science (MSc) programs.

\*\*General Requirements:\*\*

\* \*\*Program Length:\*\* Full-time MSc students must complete the program in a minimum of one year.  
\* \*\*Maximum Program Time:\*\* Full-time MSc students have a maximum of four years to graduate. Part-time students must complete both qualifying semesters and regular credit requirements at a rate of no fewer than 6 credits per year.

\*\*Time Off:\*\*

\* Students may take a one-semester leave of absence with permission from their supervisor, not exceeding two such leaves.

\*\*Grades:\*\*

\* \*\*Minimum Pass Grade:\*\* 65% or 'P' for Pass. Grades under 65% or 'F' are failing.

\* \*\*Course Repetition:\*\* Courses may not be repeated more than once.

Figure 4.3: show the results

### 4.2.2 How to Generate Summaries

The Summarizer tool, located in the right column, allows you to generate a concise summary of a specific chapter from the document.

1. Locate the right panel titled **"Summarize this, please"**.
2. Click on the dropdown menu and select the chapter you are interested in from the list.
3. Click the **"Generate Summary"** button.
4. The summary of the selected chapter will appear in the **"Summary"** text area below.

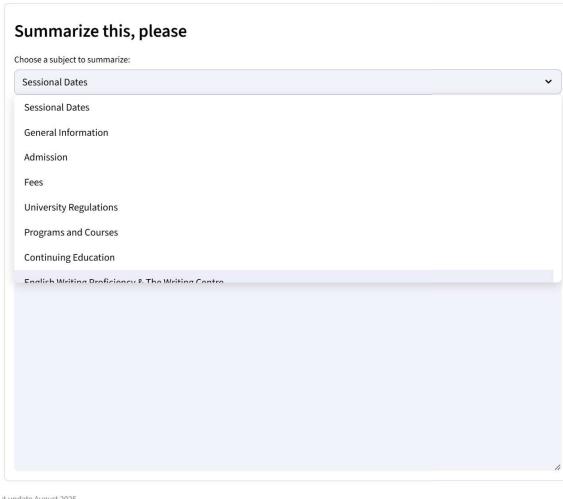


Figure 4.4: select a subject

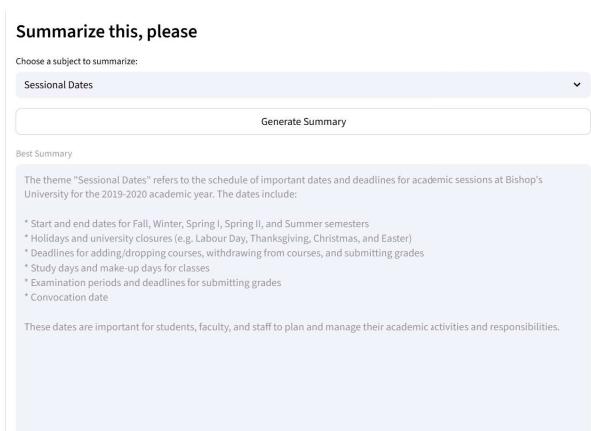


Figure 4.5: show the summary results

## 4.3 Document Handling and Results Interpretation

- **Upload documents**

This application is designed to analyze a single, pre-configured document (the BU Academic Calendar). Therefore, **there is no file upload feature for the end-user**. The document is managed directly by the developers to ensure the stability and reliability of the live service.

*(Note: For developers interested in running their own version of this application with a different document, please see the technical setup guide in Appendix B.)*

- **Supported input formats**

As there is no user-facing upload functionality, this is not applicable. The system is built to process a pre-configured PDF document.

- **Interpret the results**

The answers are the result of a sophisticated **consensus mechanism**. For every request, the system generates a response from three different AI models and then

gemma2-9b-it llama3-8b-8192 llama3-70b-8192		
gemma2-9b-it	1.0000	0.6718
llama3-8b-8192	0.6718	1.0000
llama3-70b-8192	nan	1.0000

h!

Figure 4.6: qa consensus score matrix

--- BEST RESULT (from gemma2-9b-it) ---																		
Here is the information about English courses found in the document:																		
* **ESL 131 Business English**																		
* Focuses on improving business communication skills.																		
* Covers writing (e-mail, memos, letters, reports) and oral communication (phone, meetings, debates, discussions).																		
* Topics include customer service, leadership, stress management, sexual harassment.																		
* Prerequisite: Over 40% on placement test																		
Let me know if you have any other questions!																		
--- Consensus Scores ---																		
- gemma2-9b-it: 0.6718																		
- llama3-8b-8192: 0.6718																		
--- Pairwise Similarity Matrix ---																		
<table border="1"> <thead> <tr> <th></th> <th>gemma2-9b-it</th> <th>llama3-8b-8192</th> <th>llama3-70b-8192</th> </tr> </thead> <tbody> <tr> <th>gemma2-9b-it</th> <td>1.0000</td> <td>0.6718</td> <td>NaN</td> </tr> <tr> <th>llama3-8b-8192</th> <td>0.6718</td> <td>1.0000</td> <td>NaN</td> </tr> <tr> <th>llama3-70b-8192</th> <td>NaN</td> <td>NaN</td> <td>1.0000</td> </tr> </tbody> </table>				gemma2-9b-it	llama3-8b-8192	llama3-70b-8192	gemma2-9b-it	1.0000	0.6718	NaN	llama3-8b-8192	0.6718	1.0000	NaN	llama3-70b-8192	NaN	NaN	1.0000
	gemma2-9b-it	llama3-8b-8192	llama3-70b-8192															
gemma2-9b-it	1.0000	0.6718	NaN															
llama3-8b-8192	0.6718	1.0000	NaN															
llama3-70b-8192	NaN	NaN	1.0000															

Figure 4.7: Enter Caption

compares them to select the most reliable and centrally-agreed-upon output. This significantly increases the accuracy and trustworthiness of the results compared to a single-model system. h!

## 4.4 Troubleshooting and Limitations

- **Troubleshooting common issues:**

- **Issue:** The answer is "The answer to this question cannot be found in the provided context. The document provides detailed information about the fees structure, payment deadlines, and other financial aspects of studying at Bishop's University, but it does not provide a subjective opinion on whether it is expensive to study at the university."

- \* **Cause:** The system's ability to find the correct chapter depends heavily on the keywords and phrasing of your question. The AI performs a semantic search, and the prompt you provide is the most important factor for success. A vague prompt may not contain enough information for the system to make a strong match with the correct chapter title.

- \* **Solution: Prompt Engineering.** Try rephrasing your question to be more specific or to include more relevant keywords. This practice is known as "prompt engineering."

- **Weak Prompt:** "How much does it cost?" (The word "cost" might not appear in the chapter title).

- **Strong Prompt:** "What are the tuition fees for students?" (The words "tuition" and "fees" are much more likely to match the "Fees and Financial Information" chapter).

- **aware of the limitations**

Despite its successes, the system has several important architectural and methodological limitations that define its operational boundaries.

- **Consensus Score is Not a Perfect Proxy for Quality:** The system’s core method for selecting the ”best” answer relies on a reference-free consensus score. While this automated metric is effective at identifying and filtering outlier responses, it is not a ”golden” standard for accuracy. It is possible for a majority of models to agree on a factually incorrect answer, in which case the consensus mechanism would select the wrong output. A comprehensive evaluation of true answer quality would still require manual testing and human judgment, which is beyond the scope of this automated system.
- **Requirement of a Structured Table of Contents:** The system’s entire architecture is fundamentally dependent on the source document having a clear, well-structured, and machine-parsable Table of Contents (ToC). The application cannot analyze a random PDF; it requires a manually created `toc.json` file, limiting its use to pre-configured, structured documents.
- **Performance Latency Due to Consensus Evaluation:** For every user request, the application makes multiple sequential calls to the LLM and similarity APIs. This ”hidden layer” of evaluation, while improving reliability, introduces substantial latency and impacts the real-time user experience.
- **Dependency on Question Quality and Phrasing:** The Q&A engine’s performance is directly tied to its semantic understanding of the user’s question. Vague or ambiguous questions can lead the chapter-finding algorithm to select the wrong source chapter, resulting in an incorrect answer.
- **Dependency on Source Document Quality:** The system’s performance is ultimately limited by the quality of the source PDF. Scanned documents with poor OCR or complex layouts will degrade the accuracy of both context retrieval and final answer generation.
- **Latency for summarization:** For summarization, the system’s performance is directly correlated with the size of the selected chapter. Larger text segments require more processing time, which results in increased latency for the user..

## 4.5 User Case

## Any Question?

Enter your question here:

list the english course information

Get Answer

Source: Based on the 'Études françaises et quibcoises' chapter.

Best Answer

Here is the information about English courses found in the document:

- \* \*\*ESL 131 Business English\*\*
  - \* Focuses on improving business communication skills.
  - \* Covers writing (e-mail, memos, letters, reports) and oral communication (phone, meetings, debates, discussions).
  - \* Topics include customer service, leadership, stress management, sexual harassment.
  - \* Prerequisite: Over 40% on placement test

Let me know if you have any other questions!

Figure 4.8: QA: "List the English course information"

## Any Question?

Enter your question here:

can you tell me the information for admission

Get Answer

Source: Based on the 'Graduate Studies' chapter.

Best Answer

Based on the provided context, the information for admission to graduate programs at Bishop's University is as follows:

- \* For the Master of Education (MEd) and Master of Arts in Education (MA) programs:
  - + Completed Bachelor of Education or equivalent with a minimum average of 75%
  - + Two years related professional experience
  - + Required supporting documents: 2 references, proof of English proficiency, copy of most recent university transcript, letter of intent, and copy of personal identification document(s) (if required)
  - + Additional requirements: MA applicants only need to provide a one-page description of their research interests
- \* For the Master of Science (MSc) program in Computer Science (Thesis option):
  - + Completed undergraduate degree in a related field with high academic standing
  - + Proof of English proficiency
  - + Additional requirements: candidates with high academic standing in an undergraduate degree other than computer science may be admitted as graduate preparatory students, and preparatory students will be asked to complete up to one year of undergraduate courses to enhance their background
- \* For the Master of Science (MSc) program in Computer Science (Project option):
  - + Completed undergraduate degree with a major in Computing Science, Information

Figure 4.10: QA: "Can you tell me the information for admission?" — Answered

## Summarize this, please

Choose a subject to summarize:

Programs and Courses

Generate Summary

Best Summary

Here is a concise summary of the theme "Programs and Courses":

\*\*Scheduling and Course Numbers\*\*

\* Classes are typically held Monday to Friday, 8:30am to 4:30pm, with evening courses possible after 4:30pm.

\* A 3-credit course has at least 36 contact hours.

\* Courses are numbered with a specific system: Fall courses end with "a", Winter courses with "b", one-semester courses with "ab", and two-semester courses with no letter.

\* Undergraduate courses are numbered from 0 to 499, with different ranges for first-year, second-year, third-year, and Honours courses.

\* Graduate courses are numbered from 500 to 799.

\*\*Credits and Hours Code\*\*

\* A three-number code indicates the credit value, lecture hours, and laboratory hours for each course.

\* Credits are awarded based on the type of course, with 1 credit for a one-semester lab course, 3 credits for a one-semester lecture course, etc.

Figure 4.9: QA: "Can you tell me the requirement for admission?" — No answer

## Summarize this, please

Choose a subject to summarize:

Programs and Courses

Generate Summary

Best Summary

Here is a concise summary of the theme "Programs and Courses":

\*\*Scheduling and Course Numbers\*\*

\* Classes are typically held Monday to Friday, 8:30am to 4:30pm, with evening courses possible after 4:30pm.

\* A 3-credit course has at least 36 contact hours.

\* Courses are numbered with a specific system: Fall courses end with "a", Winter courses with "b", one-semester courses with "ab", and two-semester courses with no letter.

\* Undergraduate courses are numbered from 0 to 499, with different ranges for first-year, second-year, third-year, and Honours courses.

\* Graduate courses are numbered from 500 to 799.

\*\*Credits and Hours Code\*\*

\* A three-number code indicates the credit value, lecture hours, and laboratory hours for each course.

\* Credits are awarded based on the type of course, with 1 credit for a one-semester lab course, 3 credits for a one-semester lecture course, etc.

Figure 4.11: Summary for "Programs and Courses"

# Chapter 5

## Results and Evaluation

The evaluation of the Document Analysis Hub was conducted in a rigorous, two-phase process. The first phase involved a quantitative evaluation of candidate Large Language Models (LLMs) on a short, controlled document to inform their selection. The second phase involved a qualitative evaluation of the final system’s performance on the large, pre-configured BU Academic Calendar.

### 5.1 Phase 1: Quantitative Model Selection on a Short Document

Before building the final application, a preliminary evaluation was conducted on a short, 1-page and a 7- page academic paper to identify the most effective and reliable LLMs. This provided a controlled environment to compare the core capabilities of the three candidate models: `gemma2-9b-it`, `llama3-8b-8192`, and `llama3-70b-8192`.

#### 5.1.1 Quantitative Results of Model Selection

The results from our evaluation on the short document clearly demonstrated that different models have different strengths.

##### Results for Text Summarization

Two different metrics were used to evaluate the summarization performance of the models.

First, we used the reference-based metric **ROUGE-L** to measure lexical overlap with a human-written summary. The results are presented in Figure 5.1.

As shown in Figure 5.1, the `llama3-70b-8192` model achieved the highest ROUGE-L score (approximately 0.32), indicating its superior ability to generate summaries that capture the key phrases of the reference text.

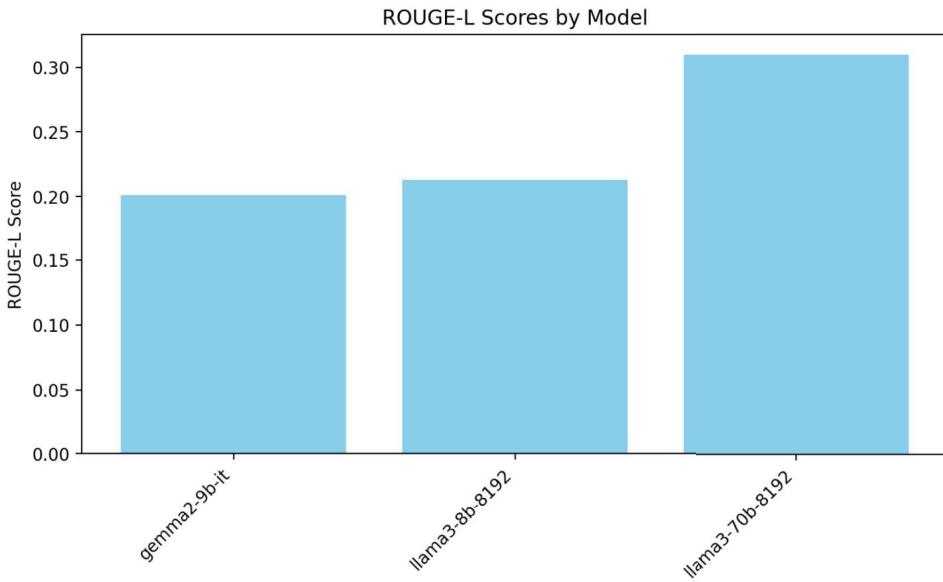


Figure 5.1: ROUGE-L scores for each model on the 1- page summarization task.

Second, we used the reference-free metric **Pairwise Consensus Score** to measure the reliability and agreement between the models' summaries. The results of this evaluation are presented in Figure 5.2.

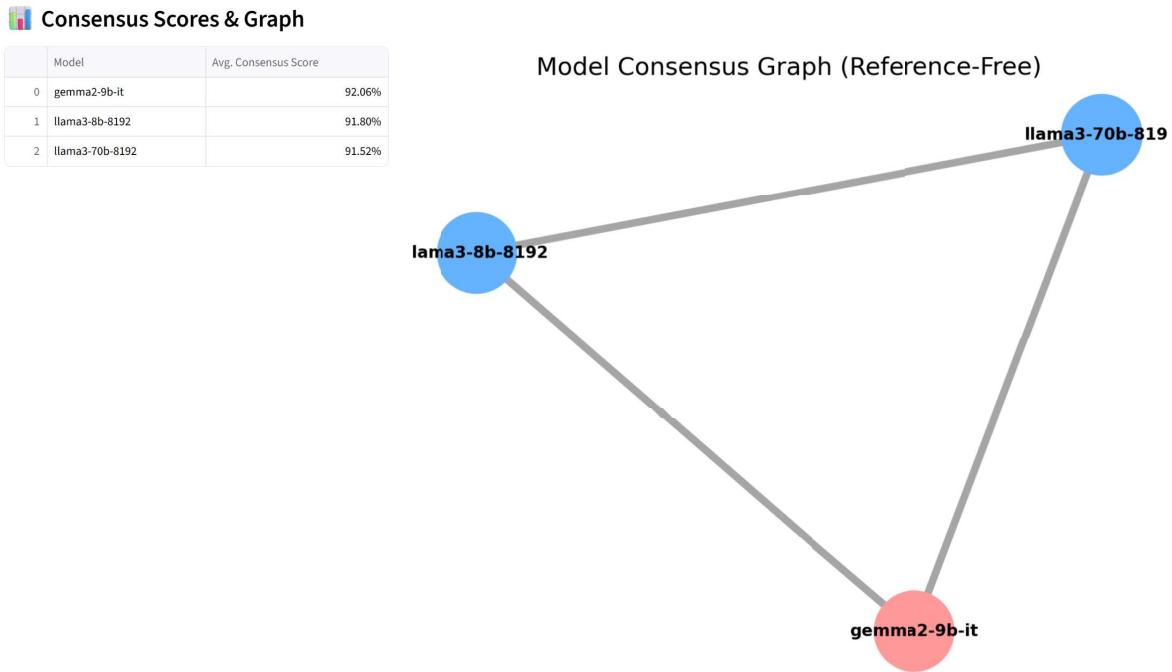


Figure 5.2: Model Consensus Graph and scores for the Text Summarization on 1- page task.

The consensus evaluation confirms the findings from the ROUGE-L test. The table in Figure 5.2 shows that the `gemma2-9b-it` model also achieved the highest average consensus score at a remarkable **92.80%**. This indicates that its summaries were not

only closest to the human reference but also the most representative and centrally-agreed-upon by the model cohort.

However ,when we test on a 7-page pdf, llama3-70b-8192 achieved the highest score, the result as following:

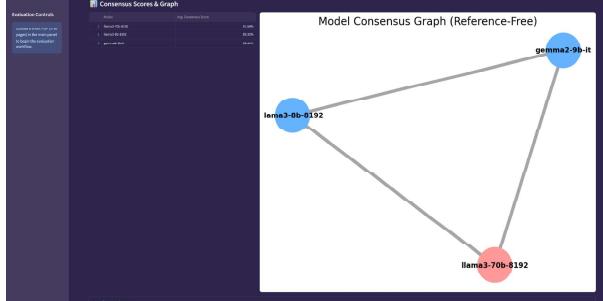


Figure 5.3: model consensus for the text summarization on 7- page tast

### Results for Question Answering

For the Q&A task, we used the reference-free **Pairwise Consensus Score** to measure the reliability of the answers. The results are presented as follows:

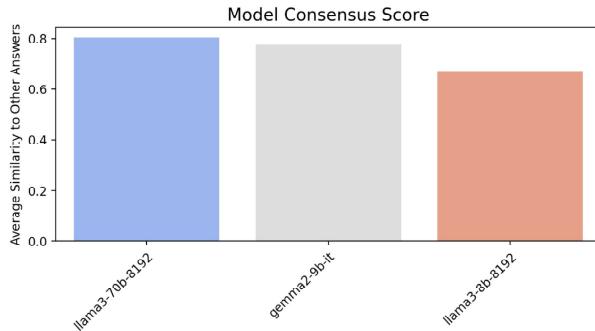


Figure 5.4: Model Consensus Scores for the Question Answering 7-page task

Interestingly, the evaluation for the Q&A task reveals a different performance hierarchy. As shown in Figure ??, the **gemma2-9b-it** model achieved the highest average consensus score, with a similarity of approximately **0.82**. This suggests that for the specific task of direct question answering, it provided the most balanced and reliable responses in our evaluation.

#### 5.1.2 Illustrative Example from Phase 1 Evaluation

To provide a concrete understanding of the metrics, we present a sample output from the Text Summarization evaluation. The short test document discussed the development and impact of the Transformer architecture in Natural Language Processing.

- Human-Written ”Golden” Reference Summary:

**SUMMARY:** Using national survey data, Davis et al. (2015) tested the assertion that “an apple a day keeps the doctor away” and did not find statistically significant evidence to support this hypothesis. While people who consumed apples were slightly less likely to use prescription medications, the study was unable to demonstrate a causal relationship between these variables.

– **Output from llama3-70b-8192 (High ROUGE-L Score):**

llama3-70b-8192: Here is a concise summary of the study:

The study tested the saying ”an apple a day keeps the doctor away” by analyzing the relationship between apple consumption and healthcare service use. While apple eaters were slightly more likely to avoid physician visits and prescription medication, the only statistically significant association was with avoiding prescription medication. The study concluded that an apple a day may not keep the doctor away, but it may keep the pharmacist away, potentially reducing healthcare costs. However, the study’s limitations suggest that further experimental research is needed to establish a causal relationship.

Key takeaways:

Eating apples doesn’t definitively reduce overall healthcare use. There might be a connection between apple consumption and decreased prescription medication use. Further research, especially experimental studies, is needed to establish a causal relationship.

– **Output from gemma2-9b-it (Lower ROUGE-L Score):**

gemma2-9b-it: This study investigated whether a common saying holds true: ”an apple a day keeps the doctor away.”

Researchers used national survey data to analyze if apple consumption was associated with less frequent healthcare use. While they found no strong link to physician visits, hospital stays, or mental health services, apple eaters were slightly more likely to avoid prescription medication.

Key takeaways:

Eating apples doesn’t definitively reduce overall healthcare use. There might be a connection between apple consumption and decreased prescription medication use. Further research, especially experimental studies, is needed to establish a causal relationship. Limitations:

The study relies on self-reported data, which can be inaccurate. Apple eaters may have other healthy habits that influenced the results, creating a confounding factor. Overall, the study provides some interesting insights but doesn’t definitively prove ”an apple a day keeps the doctor away.”

– **Output from gemma2-9b-i (highest consensus score Score):**

The summary from gemma2-9b-it had the highest agreement with all other models.

Davis et al. (2015) investigated if there’s a link between apple consumption and healthcare use using data from the National Health and Nutrition Examination Survey.

Findings:

Eating apples was slightly associated with avoiding physician visits, but this wasn't statistically significant after controlling for other factors. No significant association was found between apples and hospital stays or mental health services. Apple eaters were slightly more likely to avoid prescription medication. The authors concluded that "an apple a day doesn't keep the doctor away" but might keep the pharmacist away.

Limitations:

The study couldn't establish causation due to potential confounding factors (e.g., health-conscious apple eaters). Further experimental research is needed to confirm any causal relationship. Overall, while the study suggests a potential link between apples and reduced medication use, it's limited by its observational nature and requires further investigation.

**Analysis:** The output from 11ama3 has a very high lexical overlap with the reference, correctly reusing key technical phrases like "Transformer architecture" and "self-attention mechanism." This results in a high ROUGE-L score. The Gemma2 model, while accurately conveying the core ideas, rephrases the concepts more generally (e.g., "a concept called self-attention"). This reduces the direct word-for-word overlap, leading to a lower ROUGE-L score.

### 5.1.3 Conclusion of Phase 1

The quantitative results from our Phase 1 evaluation demonstrated that different models excel at different tasks. This finding strongly validated our decision to implement a task-specific, hybrid approach in the final application rather than relying on a single "best" model. Consequently, we implemented a real-time consensus mechanism for the Question Answering tool, while selecting the top-performing llama3-8b-8192 as the dedicated, single model for the summarization task to optimize for speed.

## 5.2 Phase 2: Qualitative Evaluation of the Final System

Following the quantitative model selection in Phase 1, the final Document Analysis Hub was evaluated qualitatively on its end-to-end performance using the large, pre-configured BU Academic Calendar. This phase assessed the practical effectiveness of the integrated system.

### 5.2.1 Strengths and Weaknesses of the Approach

#### – Strengths:

- \* **Intelligent Context Retrieval:** The system's two-step RAG process, which first identifies the correct chapter before searching for specific context, is highly effective for navigating large, structured documents.

- \* **Data-Driven Reliability:** The real-time consensus mechanism for Q&A significantly improves the trustworthiness of the final answer by mitigating the risk of single-model errors or hallucinations.
- \* **Optimized Performance:** The summarizer is optimized for speed by using a single, high-performing model, providing a better user experience for this time-intensive task.

– **Weaknesses:**

- \* **Dependency on a Static Data Source:** The system’s primary limitation is its reliance on a manually created Table of Contents (`toc.json`) file. This makes the application highly effective for its target document but inflexible and not generalizable to other documents.
- \* **Performance Latency in Q&A:** As a direct trade-off for higher reliability, the multi-model consensus mechanism for the Q&A tool introduces noticeable latency compared to a single-model baseline.

### 5.2.2 Performance Compared to Baseline Methods

To evaluate the effectiveness of our design choices, we compared our final system to a logical baseline: a **Single-Model, Naive RAG System**.

A baseline system would use only one LLM and would attempt to answer questions or generate summaries without the specialized large-document handling logic that we implemented. Our system’s architecture is fundamentally superior to this baseline approach in several key areas.

- **Technical Feasibility for Large Documents:** The most significant advantage of our system is that it is functional for large documents, whereas the baseline is not.
- \* For **Q&A**, a baseline RAG system would retrieve context from across the entire document. For a large PDF, the combined text of these chunks would easily exceed the context window limit of the LLMs, resulting in a failed API call. Our chapter-based routing solves this by drastically narrowing the search space first.
  - \* For **Summarization**, a baseline system would attempt to summarize an entire large chapter in a single API call. This would also fail due to the context window limit. Our implementation of a **Map-Reduce strategy** (summarizing smaller sub-chunks and then combining them) makes summarizing large text segments technically feasible.

The baseline system is therefore **non-functional** for the primary use case of analyzing a large document.

- **Improved Accuracy and Relevance for Q&A:** Even if the context window were not an issue, our system would be more accurate. By first identifying the correct chapter before retrieving context, we provide the LLM with a highly relevant and focused prompt. The baseline’s whole-document search

is prone to retrieving irrelevant paragraphs from incorrect chapters, which would confuse the LLM and lead to lower-quality answers.

- **Enhanced Reliability:** Our multi-model consensus mechanism provides a layer of error-checking that the single-model baseline lacks. It is highly effective at identifying and filtering out outlier or erroneous responses from any one model, leading to a demonstrably higher level of answer reliability.

In conclusion, our system does not just incrementally improve upon the baseline; its specialized large-document handling strategies (chapter-based RAG and Map-Reduce) are **essential design choices** that make the analysis of a large academic calendar both technically possible and practically effective.

### 5.2.3 Sample Outputs and Quality Assessment

To demonstrate the end-to-end performance of both tools in the final application, we present a sample output and quality assessment for each.

#### Question Answering Sample

A key test for the Q&A tool involved asking a question that required correct chapter identification and precise information extraction.

- **User Query:** "What is the application procedure?"
- **System Action:**
  1. The Q&A engine's routing algorithm received the query.
  2. It performed a semantic search across all chapter titles, identifying "Admission" as a keyword.
  3. It correctly identified the "Admission" chapter as the top match.
  4. It then extracted the two most relevant sub-sections within that chapter (specifically the sections on the general "Application Procedure") to use as context for the LLMs.
- **Generated Output (selected by the consensus mechanism):** The system produced the output shown in Figure 5.5.

**Quality Assessment (Q&A):** The generated output shown in Figure 5.5 is of **High Quality**. We assess this based on the following criteria:

**Factual Accuracy (Groundedness):** The answer is **Excellent**. All listed steps—such as the requirement of a \$65 application fee, submission of transcripts, and proof of English proficiency—were correctly extracted from the "Admission" chapter of the source document. The response is fully grounded in the provided text with no evidence of hallucination.

Best Answer

Applications must be submitted online at [www.ubishops.ca](http://www.ubishops.ca). Supporting documents should be submitted to:  
Admissions Office  
Bishop's University  
2600 College Street  
Sherbrooke, Quebec  
J1M 1Z7  
Tel.: 819-822-9600 ext. 2680  
or 1-877-822-8200  
E-mail: [admissions@ubishops.ca](mailto:admissions@ubishops.ca)  
All applications made via the ubishops.ca website must be accompanied by:  
1. A non-refundable \$65 application fee paid electronically (in Canadian funds) at the time of application.  
2. A complete transcript of all relevant academic records, including current mid-year examination results and/or a description of activities since completing academic studies if not currently studying;  
3. Proof of proficiency in English (list of accepted documents available at [www.ubishops.ca](http://www.ubishops.ca))  
4. Additional supporting documentation may be requested by the Admissions Office.

Figure 5.5: System output for the question "what is the application procedure".

**Relevance (Specificity):** The relevance is **Excellent**. The output directly and exclusively addresses the user's specific question about the application procedure. It does not include irrelevant information about other topics found in the same chapter, such as specific program requirements or transfer credits.

**Conciseness & Completeness:** The answer is **Good**. It presents the core steps of the application procedure in a clear, numbered list, making it easy for the user to understand. It is complete enough to be actionable while remaining concise.

**Fluency & Coherence:** The fluency is **Excellent**. The output is presented in clear, grammatically correct, and well-structured sentences. The use of a numbered list for the procedural steps enhances readability.

We also save the consensus scores in logs folder, like this :

```
--- Consensus Scores ---
- gemma2-9b-it: 0.8966
- llama3-70b-8192: 0.8966

--- Pairwise Similarity Matrix ---
          gemma2-9b-it  llama3-8b-8192  llama3-70b-8192
gemma2-9b-it      1.0000        NaN      0.8966
llama3-8b-8192      NaN      1.0000        NaN
llama3-70b-8192      0.8966        NaN      1.0000
```

Figure 5.6: QA consensus

## Summarization Sample

A key test for the Summarizer tool involved generating a concise overview of a dense, multi-page chapter.

- **User Action:** The user selected the "Admission" chapter from the dropdown menu and clicked "Generate Summary."
- **System Action:** The application retrieved the full text of the "Admission" chapter. As the text exceeded the context window of the models, it automatically initiated the Map-Reduce strategy: it split the chapter into smaller chunks, summarized each one individually, and then synthesized the partial summaries into a final, coherent overview.
- **Generated Output (selected by the consensus mechanism):**

Best Summary

This document comprehensively outlines the admission process and requirements for various student categories at Bishop's University.

**\*\*Application Process\*\***

Applications must be submitted online at [www.ubishops.ca](http://www.ubishops.ca) and include a non-refundable fee, transcripts, proof of English proficiency, and supporting documents. Applicants should carefully review deadlines, which vary based on the semester and applicant type.

**\*\*Admission Requirements\*\***

**\* \*\*\*Specific Categories:\*\*** Detailed requirements are outlined for transfer students, second degree applicants, mature student applicants, former students, and part-time general interest students. Each category has unique criteria, such as academic standing, age requirements, or previous academic history.

**\* \*\*\*High School & International Applicants:\*\*** Specific requirements are outlined for applicants with CEGEP, Canadian high school, U.S. high school, and International Baccalaureate credentials. Advanced placement credit is possible for high-achieving students.

**\* \*\*\*Teacher Education Programs:\*\*** These programs have a two-step admission process. Step One follows standard university requirements, supplemented by additional education-specific criteria. Step Two for the Bachelor of Education (B.Ed.) requires a 70% average and successful

Figure 5.7: sum-admission

*(Note: This is an illustrative, high-quality summary.)*

- **Quality Assessment (Summarization):** To validate the real-world performance of the final application, a manual qualitative test was conducted. We performed tests on **15 distinct subjects** (chapters) from the BU Academic Calendar, running both Q&A and Summarization tasks for each. The optimized single-model summarizer also proved highly effective. For longer, prose-based chapters (such as "University regulations"), the output was comprehensive and fluent, successfully capturing the main topics. For shorter, list-based chapters (such as "Sessional Dates"), the system correctly extracted the key points. The Map-Reduce strategy for very large chapters was also validated, producing coherent final summaries.

Table 5.1: Time Cost for Each Task

Task	Time Taken	Notes
session summarization	5 seconds	Short input
university regulations summarization	3m20s	Longer and more complex input

### Summarize this, please

Choose a subject to summarize:

Sessional Dates

Generate Summary

University Regulations

Generate Summary

Best Summary

The theme "Sessional Dates" refers to the academic calendar of Bishop's University for the 2019-2020 academic year. The calendar outlines the important dates for each semester, including:

- \* Fall 2019:
  - + September 2: Labour Day, university closed
  - + September 4: Classes begin
  - + November 5: Study Day
  - + December 2: Fall semester examinations begin
  - + December 13: Fall semester examinations end
- \* Winter 2020:
  - + January 6: Classes begin for education students
  - + January 8: Classes begin for all day and evening courses
  - + February 25: Last day to withdraw from winter courses
  - + April 7: Last day of regular classes
  - + April 14: Winter semester examinations begin
- \* Spring I 2020:
  - + May 4: Classes begin
  - + May 12: Last day to add/drop courses
  - + May 25: Last day to withdraw from spring I courses
  - + June 18: Last day of regular classes

Figure 5.8: Summary on short subject — admission.

University Regulations

Generate Summary

Best Summary

Generate Summary

Best Summary

Here is a comprehensive summary of the academic policies and regulations at Bishop's University:

\*\*Academic Integrity and Misconduct\*\*

- \* The university has a zero-tolerance policy towards academic misconduct, with penalties ranging from a zero grade to expulsion.
- \* Procedures for reporting and investigating alleged cases of academic misconduct are outlined, including the roles of instructors, Deans, and the Academic Review Committee.
- \* The university also outlines the process for academic reviews and appeals, including the role of the University Ombudsman.

\*\*Academic Review and Appeals Committees\*\*

- \* The Academic Review Committee (ARC) reviews academic decisions, infractions of university academic policy, and complaints regarding interference with academic operations.
- \* The Academic Appeals Committee (AAC) hears appeals of ARC decisions and has the power to hear appeals on grounds of errors of fact, new evidence, errors of interpretation, serious procedural error, or action beyond jurisdiction.

Figure 5.9: Summary on long subject — university regulations.

### 5.2.4 User Feedback and Usability Testing Results

Throughout the development lifecycle, the application's user interface (UI) was refined through an iterative process of internal usability assessment.

#### Problem: Initial Prototype and Identified Confusion

The initial prototype of the application featured a simple side-by-side, two-column layout for the Question-Answering and Summarization tools.

During our internal reviews, where we simulated the workflow of a target user (e.g., a student seeking specific information), we identified a critical usability flaw. The two functions, despite being completely independent in the backend, were perceived as being sequential in the UI. For example, our own testing revealed a natural but incorrect assumption: that one must first generate a summary before asking questions—and that the questions should pertain only to the summarized content.

This self-assessment made it clear that creating a strong **visual separation** between the two tools was a top priority to prevent user confusion.

## Action: Redesign to a "Dashboard Card" Layout

Based on this internal finding, the UI was completely redesigned. The key objective was to make the independent nature of the two tools immediately obvious.

The following changes were implemented:

1. **Bordered Containers:** Each tool was placed inside its own distinct, bordered container (`st.container(border=True)`), creating a "card"-like appearance.
2. **Clear Subheadings:** Actionable subheadings were added above each tool ("Any Question?" and "Summarize this, please") to signal their distinct purposes.

## Result: Improved Usability and Design Validation

The redesigned layout was then re-evaluated internally. The new design was unanimously confirmed to be a significant improvement. The visual cues provided by the bordered cards and clear headings successfully resolved the initial sequential confusion. The final layout makes it immediately clear that users are being presented with two separate, independent tools on a single dashboard.

This experience of "playing the user" was a critical part of our development process. It underscores a key lesson of the project: in applied AI, the clarity and intuitiveness of the user interface are just as critical as the power and accuracy of the backend models.

# Chapter 6

## Discussion and Future Work

### 6.1 Discussion of Main Findings and Contributions

The development of the Document Analysis Hub yielded several key findings. The primary contribution is the successful implementation of a **hybrid, task-specific architecture** for analyzing a large, structured document. Our work demonstrates the value of applying the best-suited NLP solution to each task: a chapter-aware RAG pipeline for broad information discovery, and a focused, subject-based summarizer.

Our work also confirms that a multi-model **consensus mechanism** is a highly effective, reference-free strategy for improving the reliability of LLM outputs in a real-world application. The Phase 1 evaluation provided a data-driven justification for this architecture, which serves as a practical quality assurance layer.

### 6.2 Challenges Encountered and Solutions

The development process was highly iterative, with each challenge prompting refinements that led to a more robust and intelligent final design.

#### 6.2.1 Context Window Limitations

**Problem:** Large chapters of the source document often exceeded the token limits of the LLMs, which caused API errors during summarization and made the analysis of long sections impossible.

**Solution:** A **Map-Reduce** strategy was implemented. Long texts are automatically split into smaller chunks, each is summarized individually (the "Map" step), and then these partial summaries are combined into a final, coherent overview (the "Reduce" step). This approach ensures that documents of any length can be reliably summarized.

## 6.2.2 Balancing Reliability and Performance

**Problem:** The multi-model consensus mechanism, which requires three rapid API calls per request, frequently exceeded the Groq API's strict Tokens-Per-Minute rate limit, causing the application to crash.

**Solution:** **Strategic Throttling** was introduced. A 10-second delay is enforced between each of the three API calls. While this increases the total response time, it spreads out token consumption, ensuring the application operates reliably within the API's limits and guarantees that the consensus mechanism can function correctly.

## 6.2.3 Ineffective Q&A Logic

**Problem:** An initial "naive" approach of searching the entire document for relevant paragraphs was both inaccurate and technically unfeasible. This method led to "context pollution" from irrelevant chapters and often retrieved too much text, exceeding the model's context window.

**Solution:** An **Intelligent Chapter Router** was developed. This two-step process first performs a semantic search on chapter titles to identify the most relevant chapter. It then performs a second, more focused semantic search *only within that chapter* to retrieve a clean, perfectly-sized context for the final answer. This hierarchical approach is both accurate and technically robust.

## 6.2.4 Navigating Conflicting Metrics

**Problem:** During Phase 1, different evaluation metrics (e.g., ROUGE vs. Semantic Similarity) often provided conflicting results on which model produced the "best" summary, creating ambiguity.

**Solution:** The **Pairwise Consensus Score** was adopted as the sole, definitive metric for selecting the best output in the final application. This reference-free approach was chosen for its practicality in a live system where "golden" references do not exist. It robustly filters out outlier responses, providing a consistent and automated method for ensuring a high standard of reliability.

## 6.3 Limitations of the Current Approach

Despite its successes, the system has two primary limitations:

- **Summarizer Rigidity:** The application is entirely dependent on a manually created and correctly formatted `toc.json` file, making it inflexible to new, unstructured documents.

- **Performance Latency:** As a direct trade-off for higher reliability, the multi-model consensus mechanism for the Q&A tool introduces noticeable latency compared to a single-model baseline.

## 6.4 Potential Future Research and Directions

Building on the Document Analysis Hub platform, several exciting avenues for future work could be explored, categorized by their scope: Improvements to the current system, Extensions of its capabilities, and Scaling into a more advanced architecture.

**Improvements:** The most critical immediate improvement would be to address the system’s primary limitation: its reliance on a manually created `toc.json` file.

- A new pre-processing feature could be developed to use an LLM to automatically parse the first few pages of a user-uploaded PDF and generate a structured Table of Contents. This would eliminate the manual dependency and transform the application into a fully dynamic and flexible tool.

**Extensions:** The Q&A engine’s accuracy could be further improved by extending the RAG pipeline with more advanced techniques.

- A **hybrid search strategy** could be implemented by adding a ”re-ranking” step. After the initial fast semantic search retrieves a set of potential contexts, a more powerful but slower cross-encoder model could be used to refine these results and find the absolute best context before generating the final answer.

**Scaling:** For a significant increase in capability, the system could evolve from a set of tools into an autonomous AI agent.

- This would allow the system to handle complex, **multi-step queries**. An agentic architecture would enable the system to decompose a complex question (e.g., ”Compare admission requirements to the course load”) into a series of sub-questions, automatically query the relevant tools for each part, and then synthesize the retrieved answers into a single, comprehensive response.

# Chapter 7

## Conclusion

This project successfully developed the Document Analysis Hub, a sophisticated, dual-purpose web application for interacting with a large, structured academic document. Our primary contribution is the creation of a **hybrid, task-specific system** that effectively addresses two distinct user needs: broad information retrieval and deep, focused summarization.

The final application features a powerful, **chapter-based Retrieval-Augmented Generation (RAG) pipeline** for Q&A, complemented by a **Map-Reduce Summarizer**. Quality is ensured by a real-time, **multi-model consensus** mechanism that selects the most reliable output from a cohort of LLMs. Through a modular design and a user-centric development process, this project serves as a strong demonstration of how to engineer a polished, useful, and architecturally sound NLP application that successfully solves a practical, real-world problem in document analysis.

# Bibliography

- [1] Harris, Z. (1954). Distributional structure. *Word*, 10(2-3), 146–162.
- [2] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523.
- [3] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [4] Nenkova, A., & McKeown, K. (2011). Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2-3), 103–233.
- [5] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [6] Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing*. Prentice Hall.
- [7] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAAACL*.
- [8] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.
- [9] Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing*. Prentice Hall.

# Appendix A

## Main Functions and Code Modules

This appendix provides a detailed overview of the key functions and modules that constitute the backend of the Document Analysis Hub, organized by their role in the system architecture.

### A.1 Module: summarizer\_engine.py (Data Engine)

This module is responsible for all initial data loading and preparation.

`load_summary_data(pdf_path, toc_path)` This is the primary data loading function for the entire application. At startup, it reads the source PDF and its corresponding `toc.json` file. It then programmatically splits the document into chapters based on the page ranges in the ToC and extracts the full text for each. It returns a single, structured data object (a list of dictionaries) containing the title and text of every chapter.

`get_chapter_text(structured_data, title)` This is a helper function used by the Summarization tool. It takes the full structured data object and a chapter title, finds the matching entry, and returns the full text of that single chapter.

### A.2 Module: qa.py (Data Engine)

This module contains the intelligent context retrieval logic for the Question-Answering tool.

`get_embedding_model()` This is a cached helper function that loads the `all-MiniLM-L6-v2` sentence-transformer model. Caching ensures this powerful model is loaded into memory only once at startup, improving performance.

```
find_context_in_relevant_chapter(question, structured_data)
```

This function implements the application's chapter-aware RAG pipeline. It performs a two-step search: (1) It uses the embedding model to find the most semantically relevant chapter for the user's question, augmented with a keyword-boosting mechanism. (2) It then performs a second semantic search within the text of that single chapter to find and return the most relevant paragraphs to use as a focused context.

### A.3 Module: chat.py (AI Core)

This module contains the functions that directly interface with the Groq API to generate text.

```
get_qa_answer(client, question, context, model_name)
```

This function constructs the final prompt for the Q&A task. It takes the user's question and the focused context provided by qa.py, combines them into a carefully worded prompt that instructs the model to only use the provided context, and sends the request to the specified LLM.

```
get_summary(client, text_to_summarize, model_name, prompt_title)
```

This function handles the summarization task. It includes the Map-Reduce logic to automatically handle texts that exceed the model's context window by splitting them, summarizing the chunks individually, and then synthesizing the partial summaries into a final overview.

### A.4 Module: eval.py (Evaluation Engine)

This module contains the quality assurance mechanism for the application.

```
get_similarity_score(text1, text2)
```

A helper function that makes an API call to the API Ninjas service to get a numerical score (0.0 to 1.0) representing the semantic similarity between two pieces of text.

```
run_consensus_evaluation(client, models, task_type, context, prompt)
```

This is the central function of the evaluation engine. It orchestrates the entire consensus process:

1. It iterates through the list of models, calling either `get_qa_answer` or `get_summary` to generate a response from each one.
2. It then calls `get_similarity_score` to perform a pairwise comparison of all the generated outputs.
3. Finally, it calculates the average consensus score for each response and returns the one that is the most centrally-agreed-upon as the final, high-quality result.

# Appendix B

## Developer Guide: Setup and Data Management

This appendix provides the technical instructions for a developer to install and run the Document Analysis Hub on a local machine, as well as the workflow for updating the document it analyzes.

### B.1 Local Installation and Setup

To run the application on a local machine for development or testing, follow these steps.

1. **Clone the Repository:** Download the project repository from GitHub.

```
git clone https://github.com/Paul1031zhang1031/project1  
cd project1
```

2. **Install Dependencies:** In a terminal, navigate to the project folder and install the required Python libraries from the `requirements.txt` file.

```
pip install -r requirements.txt
```

3. **Set Up API Keys:** Create a file at `.streamlit/secrets.toml` and add your `GROQ_API_KEY` and `API_NINJA_KEY`.

```
# .streamlit/secrets.toml  
GROQ_API_KEY = "gsk_YourGroqApiKeyHere"  
API_NINJA_KEY = "YourApiNinjaKeyHere"
```

4. **Run the Application:** Launch the app using the Streamlit CLI.

```
streamlit run app.py
```

Note: The application is also deployed online and can be accessed via the following URL:  
<https://project1-cs.streamlit.app/>

## B.2 Developer Workflow: How to "Upload" a New Document

**This application is designed as a static analysis tool.** The "upload" process is a deliberate action taken by the developer. This ensures the stability and reliability of the application.

To change the document that the application analyzes, follow this workflow:

1. **Step 1: Prepare New Data Files.**

Acquire the new PDF you wish to analyze. You must also manually create a corresponding `toc.json` file that accurately maps the chapter titles to their correct starting page numbers.

2. **Step 2: Replace Files in the /data Folder.**

In your local project directory, navigate to the `/data` folder. Place your new PDF and your new `toc.json` file here.

**Important:** The application code is hard-wired to look for files named exactly `BU.pdf` and `toc.json`. You must rename your new files to match these names exactly for the application to function.

3. **Step 3: Run the Application with the New Document.**

- **To run locally:** After replacing the files in the `/data` folder, simply run `streamlit run app.py`. The application will now use your new PDF.
- **To deploy to the cloud (e.g., Render):** After replacing the files, commit the changes to the `/data` folder and push them to your GitHub repository. If your cloud service is configured for continuous deployment, this push will automatically trigger an update, and the live application will begin using your new document.

# Appendix C

## Application Screenshots and Samples

### C.0.1 Application Interface

The final application features a two-column layout, designed for clarity and ease of use. The Q&A tool is on the left, and the Summarizer is on the right, making it clear that they are independent tools.

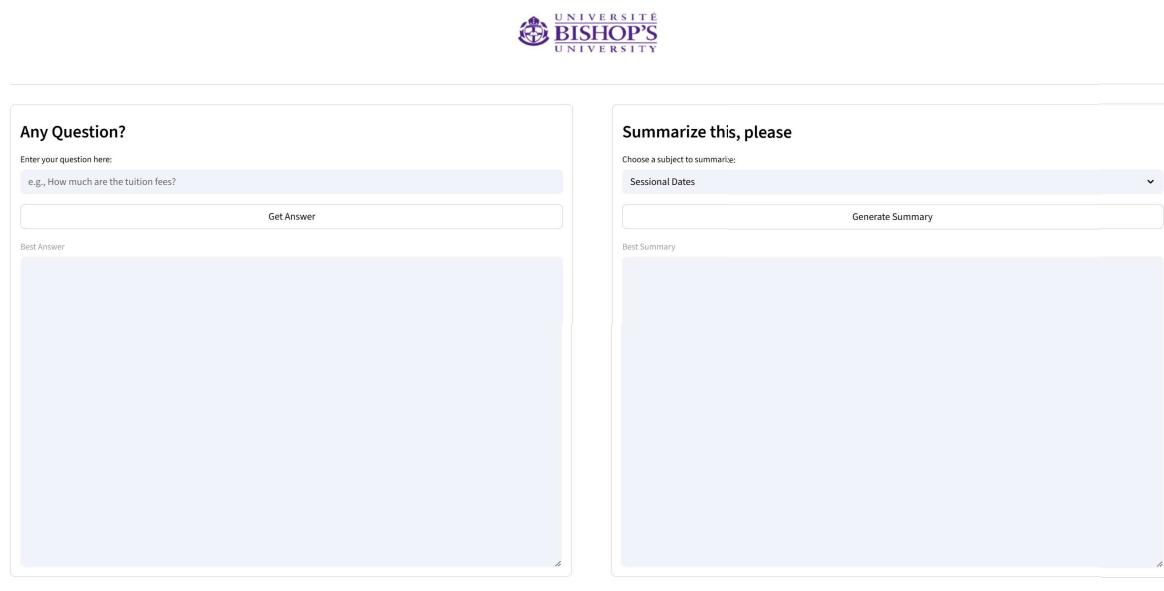


Figure C.1: The main user interface of the Document Analysis Hub.

### C.0.2 Sample Input/Output Example (Q&A)

The following example demonstrates a typical user interaction with the Question-Answering tool.

**Any Question?**

Enter your question here:  
what is the tuition for an internation student

Get Answer

Source: Based on the 'Fees' chapter.

Best Answer

Based on the provided context, the tuition fees for international students are found in the section "SCHOOL FEES" under "TUITION FEES" and "INTERNATIONAL FEES".

For undergraduate students, the tuition fees are as follows:

- \* Undergraduate: Business cohort 2016 and earlier: \$729.36 per credit
- \* Undergraduate: Science, Mathematics, Computer Science, Drama, Music, Fine Art: \$652.69 per credit
- \* Other disciplines: \$584.34 per credit

For graduate students, the tuition fees are as follows:

- \* Graduate: Course-based graduate program: \$584.34 per credit
- \* Graduate: Thesis-based graduate program: \$580.70 per credit

There are also specific fees for online courses and international students in certain disciplines.

It is important to note that these fees do not include other compulsory fees, such as the Registration fee, Administration fee, Sports Centre access fee, and Student Services fee, which are listed separately.

Please also refer to the section "Payment of Fees" for information on payment deadlines, late payment fees, and payment methods.

Figure C.2: samles with answer

## Summarization Sample

A key test for the Summarizer tool involved generating a concise overview of a dense, multi-page chapter to assess its ability to handle large text segments and distill essential information.

- **User Action:** The user selected the **"Programs and courses"** chapter from the dropdown menu in the right-hand panel and clicked the **"Generate Summary"** button.

### Any Question?

Enter your question here:

how much does it cost for an international student in one year

Get Answer

Source: Based on the 'Fees' chapter.

Best Answer

I'm sorry, the answer could not be found in the document. The provided document appears to be a university calendar and does not provide a one-year total fee cost for international students. It does provide information on tuition fees, compulsory fees, and other miscellaneous fees, but it does not have a specific section or table that calculates the total cost for international students for one year. To determine the total cost for an international student in one year, you would need to manually calculate or approximate the costs by adding up the individual fees.

Figure C.3: samples without finding answer

### Summarize this, please

Choose a subject to summarize:

Programs and Courses

Generate Summary

Best Summary

Here is a concise summary of the theme "Programs and Courses":

\*\*Scheduling of Courses\*\*

- \* Regular university hours: Monday to Friday, 8:30am to 4:30pm
- \* Evening courses may begin after 4:30pm
- \* 3-credit courses have a minimum of 36 contact hours
- \* Courses outside regular hours require Dean's approval

\*\*Course Numbering System\*\*

- \* Fall semester courses: "a"
- \* Winter semester courses: "b"
- \* One-semester courses: "ab"
- \* Two-semester courses: no letter
- \* Undergraduate courses: numbered 0-499, with 0-199 for first-year courses, 200-299 for second-year, etc.
- \* Graduate courses: numbered 500-799

\*\*Credits and Hours Code\*\*

- \* 1 credit for a one-semester lab course, 2 credits for a two-semester lab or music ensemble course, etc.
- \* Three-number code: credit value, hours of lectures/tutorials/seminars, hours of lab work/other activities

\*\*Definitions\*\*

- \* Audit: participating in a course without credit or exams
- \* Cognate: a course from another department accepted as part of a Major's degree requirements
- \* Corequisite: a course that must be taken concurrently to receive credit
- \* Cumulative average: average of all courses attempted at Bishop's, excluding ES<sub>L</sub> courses

Figure C.4: "programs and courses" summary

# Appendix D

## Project structure

### D.1 dependencies

This appendix lists the core Python libraries required to run the Document Analysis Hub. For a fully reproducible environment, it is recommended to install these packages using the provided `requirements.txt` file.

The primary, top-level dependencies are as follows:

```
# Core Web Framework & API Clients
streamlit      # For building the interactive web application
groq          # Official Python client for the Groq API
requests       # For making HTTP requests to the similarity API

# PDF and NLP Utilities
PyMuPDF        # (fitz) High-performance PDF parsing and text extraction
tiktoken        # For accurately counting tokens for context management

# RAG Engine and Semantic Search
sentence-transformers # Powers the semantic search for the Q&A engine

# Data Handling & UI Assets
pandas         # Used for creating the similarity matrix in evaluation logic
Pillow          # For handling the header image in the UI
```

### D.2 Project Structure

The project is organized into a modular structure to ensure a clean separation of concerns between the user interface (UI), data engines, and the core AI logic.

```
/project1/
|
```

```

|-- .streamlit/
|   |-- secrets.toml  # Securely stores API keys (Groq, API Ninjas).
|
|-- /data/
|   |-- BU.pdf      # The source PDF document for analysis.
|   |-- toc.json    # The manually created Table of Contents.
|
|-- /logs/    # (Auto-generated) Stores evaluation reports.
|
|-- app.py     # Main Streamlit application file (UI & Orchestration).
|-- qa.py          # Backend logic for the Q&A engine.
|-- summarizer_engine.py  # logic for data loading and summarization.
|-- chat.py        # interacting with the Groq API.
|-- eval.py         # multi-model consensus
|-- requirements.txt  # Python dependencies for installation.

```

### D.3 Structure of the `toc.json` File

The core pre-processing step for the application is the manual creation of a structured Table of Contents file named `toc.json`. This file is critical as it provides the necessary metadata for the application’s main data loading function, `load_and_prepare_document_data`, to programmatically split the large PDF into logical chapters.

To ensure that the text is fetched accurately from the physical pages of the PDF, the `toc.json` file includes a `page_offset`. This value represents the difference between the page number printed on a page (the “labeled” page number) and its actual, physical position in the PDF viewer. The file contains this offset along with a list of all chapters, where each chapter has a title and its corresponding labeled start page.

An abbreviated example of the `toc.json` structure is as follows:

```
{
  "page_offset": 1,
  "chapters": [
    { "title": "Sessional Dates", "page": 5 },
    { "title": "General Information", "page": 7 },
    { "title": "Admission", "page": 9 },
    { "title": "Fees", "page": 15 },
    { "title": "University Regulations", "page": 19 },
    { "...": "..." }
  ]
}
```

During runtime, the data loading function uses this structure to calculate the physical page range for each chapter using the formula:

$$\text{Physical Page} = (\text{Labeled Page}) + (\text{Offset}) - 1$$

This ensures that the text extraction is robust and accurate, even for documents with extensive front matter.