

IBA CHATBOT FOR STUDENTS
BASED ON RAG

Group members :
Syed Muhammad Irtiza Zaid (23973)
Muhammad Usman (24454)
Azizullah Khan (24911)

Table of Contents

Introduction	3
Application of RAG Architecture for IBA University Chatbot	3
Data	3
Test queries.....	4
Data cleaning	5
<i>Data through PDF</i>	<i>5</i>
<i>Tabular Data.....</i>	<i>5</i>
<i>Data through API.....</i>	<i>6</i>
Methodology for Implementing RAG Model for IBA Chatbot	7
Vector DB Setup stage	8
<i>Approach 1.....</i>	<i>8</i>
1. RAG Read Docs From the Source	8
2. Send Docs to Chunk Function	8
3. Send Chunks to Embedding Function.....	9
4. Store Vectors in DB.....	9
Results:.....	9
<i>Approach 2.....</i>	<i>9</i>
1. RAG Read Docs From the Source	9
2. Send Docs to Chunk Function	10
3. Send Chunks to Embedding Function.....	10
4. Store Vectors in DB.....	10
Results	10
<i>Approach 3.....</i>	<i>10</i>
1. RAG Read Docs From the Source	11
2. Send Docs to Chunk Function	11
3. Send Chunks to Embedding Function.....	11
4. Store Vectors in DB.....	11
Results:.....	11
Retrieval stage	11
<i>User Submits Query.....</i>	<i>11</i>
<i>Parrot</i>	<i>11</i>
<i>Search Similar Vectors in DB</i>	<i>13</i>
<i>BM25 Retriever.....</i>	<i>13</i>
<i>Results.....</i>	<i>13</i>
Generation stage.....	14
<i>Long-Context reordering</i>	<i>14</i>
<i>Combine The Query & The Documents</i>	<i>14</i>
<i>LLM used.....</i>	<i>14</i>
<i>Results.....</i>	<i>16</i>

Introduction

In recent years, the use of artificial intelligence (AI) has become increasingly prevalent in various industries, including education. One of the most promising AI technologies is Large Language Models (LLMs), which power intelligent chatbots and other natural language processing (NLP) applications. These chatbots have the potential to revolutionize the way students interact with their academic institutions, providing them with instant and personalized support.

However, the nature of LLM technology introduces unpredictability and inconsistency in the generated responses, which can be problematic in an educational setting. Moreover, the static nature of LLM training data implies that the knowledge possessed by the model is limited to a specific cut-off date, which may not be up-to-date with the latest academic research and findings.

To address these challenges, the Retrieve, Generate, and Grounded (RAG) approach is a potential solution. RAG redirects the LLM to retrieve pertinent information from pre-determined and authoritative knowledge sources, such as academic databases and institutional policies. This ensures that the generated responses are accurate, reliable, and up-to-date, providing students with the best possible support.

Application of RAG Architecture for IBA University Chatbot

The RAG model has demonstrated exceptional performance in scenarios where high-quality data is available. With this in mind, we aimed to develop a chatbot for IBA University that leverages the university's online available data and documents. The primary objective of this chatbot is to interact with students and provide them with accurate and timely responses to their queries, thereby reducing the need for them to email departments to ask about questions that are already addressed in these documents.

Our vision is for the chatbot to serve as the first point of contact for students in case of any issues or concerns. By using the RAG architecture, the chatbot can retrieve relevant information from authoritative and pre-determined knowledge sources, such as the university's policies and procedures, academic calendars, and course catalogs. This ensures that the generated responses are accurate, reliable, and up-to-date, providing students with the best possible support.

In the following sections, we will discuss the implementation of the RAG architecture for the IBA University chatbot in more detail, including the data sources used, the training and evaluation process, and the potential benefits and limitations of the system.

Data

To develop a chatbot for IBA University that can accurately and efficiently answer student queries, we utilized the following five data sources:

1. **Student Handbook:** This comprehensive document provides essential information for students, including academic policies, student services, and campus resources.
2. **Program Announcement PDF:** This document contains detailed information about the university's academic programs, including course descriptions, admission requirements, and degree requirements.
3. **Faculty Information through the Website:** We collected data on IBA University's faculty members, including their names, titles, and areas of expertise, to enable the chatbot to provide accurate information about faculty members.
4. **Data of IBA Program on Website:** We also collected data on the university's academic programs, including program descriptions, admission requirements, and course offerings, to ensure that the chatbot can provide up-to-date and accurate information about the university's programs.
5. **FAQs Available on IBA Site:** We utilized the frequently asked questions (FAQs) available on the university's website to train the chatbot to respond to common student queries accurately.

While we had access to various data sources, we found that the data in the student handbook and program announcement was sufficient and largely consisted of all the information students require. Therefore, instead of picking more data options for data redundancy, we focused on maximizing the cleaning and structuring of this data. We paid particular attention to tabular data, as it is often essential for accurately answering student queries. By doing so, we were able to ensure that the chatbot can provide accurate and reliable responses to student queries.

Test queries

To evaluate the performance of our IBA chatbot, we tested it with a set of questions that students commonly ask. These questions included:

- What are the eligibility criteria for candidates applying for the BSCS program?
- What is the pre-requisite for the Data Structures course?
- What is the email address of Sir Sajjad Haider?
- What is the email address of the IT help desk?
- Is it possible to exempt the IBA aptitude test?

These two questions were designed to further evaluate the pipeline's performance on these tasks.

- Write an email to Sir Sajjad Haider that I need extension for Project report submission.
- What are the admission criteria of LUMS?

We assessed the chatbot's responses based on their accuracy, relevance, and coherence. In cases where there were multiple possible answers, we used our judgment and the similarity/rank of the retrieved documents to select the best response.

Data cleaning

Platform used: local machine (MacOS M1)

Data through PDF

To clean and preprocess our PDF data, we employed some techniques similar to those used in the C4 data cleaning process. We only retained lines that ended with a terminal punctuation mark, such as a period, exclamation mark, question mark, or end quotation mark. Additionally, we discarded any pages that contained fewer than three sentences and only kept lines that had a minimum of five words.

To eliminate duplicates in the dataset, we removed all but one of any three-sentence spans that occurred more than once. This ensured that the data was unique and diverse, which is crucial for training a high-performing language model.

Overall, these preprocessing steps helped us to improve the quality of our PDF data and make it suitable for use in our language model training pipeline.

Code:

rag-2-gen.ipynb

Tabular Data

To extract tables from the splitted PDFs, we utilized an online tool called Nanonets. The link to this tool is: <https://nanonets.com/free-tools/extract-table-from-pdf>. After extracting the tables, we performed some preprocessing steps to ensure the accuracy and consistency of the data.

```
import pdfplumber

# Replace 'path_to_your_pdf.pdf' with the actual path to your PDF file
pdf_path = 'C:\\Users\\HP\\Downloads\\pa_pdfs\\split_1.pdf'

# Create an empty list to store extracted tables
tables = []

# Open the PDF file
with pdfplumber.open(pdf_path) as pdf:
    # Iterate through each page in the PDF
    for page in pdf.pages:
        # Extract tables from the page
        extracted_tables = page.extract_tables()
        if extracted_tables:
            # Extend the list of tables with tables extracted from the current page
            tables.extend(extracted_tables)

# Convert the list of tables into Pandas DataFrames
dfs = [pd.DataFrame(table) for table in tables]
```

Figure 1: code that shows how we extracted tabular data from pdfs

Firstly, we dropped the first column of tables that contained a semester-wise breakdown of courses, if necessary. This step was taken to avoid any confusion and ensure that the data was presented in a clear and concise manner.

Secondly, we concatenated tables that were splitted during CSV reading. This step was taken to ensure that all relevant data was included and presented in a comprehensive manner.

Lastly, we created metadata corresponding to each table in the sequence they appeared in the list. This step was taken to maintain the integrity of the data and ensure that it was properly structured for use in our chatbot.

By following these preprocessing steps, we were able to significantly improve the quality of our tabular data

```
metadata_list = [
    {'table_name': 'Master of Business Administration (MBA) Business Orientation Program', 'description': ''},
    {'table_name': 'Master of Business Administration (MBA) program structure', 'description': 'Time for completion is approximate only; actual duration Corporate internship is mandatory for students coming from non-BBA academic backgrounds.\nPersonal effectiveness program consists of 5 sessions delivered by industry professionals.'},
    {'table_name': 'Master of Business Administration (MBA) morning semester-wise sequence of courses', 'description': 'Semester 2'},
    {'table_name': 'Master of Business Administration (MBA) morning semester-wise sequence of courses', 'description': 'Semester 4'},
    {'table_name': 'Master of Business Administration (MBA) morning semester-wise sequence of courses', 'description': 'Semester 1'},
    {'table_name': 'Master of Business Administration (MBA) morning semester-wise sequence of courses', 'description': 'Semester 3'},
    {'table_name': 'Master of Business Administration (MBA) evening semester-wise sequence of courses', 'description': 'Semester 2'},
    {'table_name': 'Master of Business Administration (MBA) evening semester-wise sequence of courses', 'description': 'Semester 6'},
    {'table_name': 'Master of Business Administration (MBA) evening semester-wise sequence of courses', 'description': 'Semester 3'},
    {'table_name': 'Master of Business Administration (MBA) evening semester-wise sequence of courses', 'description': 'Semester 7'},
    {'table_name': 'Master of Business Administration (MBA) evening semester-wise sequence of courses', 'description': 'Semester 4'},
    {'table_name': 'Master of Business Administration (MBA) evening semester-wise sequence of courses', 'description': 'Semester 1'},
    {'table_name': 'Master of Business Administration (MBA) evening semester-wise sequence of courses', 'description': 'Semester 5'},
    {'table_name': 'Master of Business Administration (MBA) electives', 'description': 'Students can select 5 courses as electives. These may be selected from the list of courses offered by the School of Business Studies (SBS) that are not core courses for the MBA program. These elective courses, therefore, include courses offered by the Finance department (usually course codes beginning with FIN), Accounting and Law department (beginning with ACC and LAW), Marketing department (usually beginning with MKT) and the Management department (usually beginning with MGT, HRM, ENT, and SCM); subject to fulfillment of pre-requisites. The list of course offerings differ from semester to semester and students can find the current offerings on the website or with the relevant program offices.'},
]
```

Figure 2: meta data of all of the tables extracted

Code:
pdfScrapping.ipynb

Data through API

During the retrieval, we realized a crucial mistake in our data preprocessing process. Initially, we had converted the website data and faculty data obtained from the API directly from CSV to text format. However, this resulted in poor data retrieval performance, with both the vector database and BM25 retriever failing to provide satisfactory results. Moreover, the retrieved data was poorly structured, making it difficult to extract meaningful information.

To address this issue, we went back to the data processing step and wrote a script to transform the data into a properly structured format. This involved cleaning and formatting the data to ensure that it was consistent and easy to understand. For example, we ensured that all faculty names were formatted in the same way, with the first and last names. We also removed any unnecessary information, such as HTML tags and formatting, from the website data.

```

for faculty in faculty_data:
    # Scrape the webpage for the current URL
    faculty_soup = scrape_webpage(faculty["url"])

    # Find the <div> tag with class 'faculty-profile-info'
    faculty_info = faculty_soup.find('div', class_='faculty-profile-info')

    h3_tags = faculty_info.find_all('h3')

    # Extract the data from the webpage and add it to the dictionary
    faculty_name = faculty_info.h1.text.strip()
    faculty['name'] = faculty_name
    faculty['faculty_details_url'] = "For further information regarding " + faculty_name + ", kindly refer to the following webpage: " + faculty['url']
    faculty['identity'] = faculty_info.h1.text.strip() + " is a member of IBA Computer Science Faculty."
    faculty['position'] = faculty_name + " is a " + faculty_info.h2.text.strip()
    faculty['department'] = faculty_name + " works in the Department of " + h3_tags[0].get_text(strip=True).replace('Department:', '')
    faculty['specialization'] = faculty_name + " specializes in " + h3_tags[1].get_text(strip=True).replace('Specialization:', '')
    faculty['onboard_status'] = faculty_name + " is currently " + h3_tags[2].get_text(strip=True).replace('Onboard Status:', '')
    faculty['email'] = "You can reach " + faculty_name + " at " + h3_tags[3].get_text(strip=True).replace('Email:', '')

```

Figure 3: a part of script that transforms json file format into meaningful text

```

Dr. Sajjad Haider
For further information regarding Dr. Sajjad Haider kindly refer to the following webpage: https://www.iba.edu.
Dr. Sajjad Haider is a member of IBA Computer Science Faculty.
Dr. Sajjad Haider is a Professor
Dr. Sajjad Haider works in the Department of Computer Science
Dr. Sajjad Haider specializes in Information Technology
Dr. Sajjad Haider is currently Available
You can reach Dr. Sajjad Haider at sahaider@iba.edu.pk
Here is the list of Dr. Sajjad Haider's Work Experience: Professor at Institute of Business Administration Kara
Here is the list of Dr. Sajjad Haider's Qualifications: PhD (Information Technology) George Mason University -
Here is the list of courses taught by Dr. Sajjad Haider: ADVANCED TEXT ANALYTICSBIG DATA ANALYTICSBUSINESS ANAL
Here is the list of Dr. Sajjad Haider's research interest: Information Technology
Here is the list of Dr. Sajjad Haider's projects: List to come

```

Figure 4: this image shows how data was structured

This improvement in data preprocessing had a significant impact on the performance of our chatbot. The vector database and BM25 retriever were able to retrieve relevant information more accurately and efficiently, and the retrieved data was properly structured, making it easier to extract meaningful information.

Code:

Faculty_data_full_time.ipynb

Faculty_data_part_time.ipynb

Methodology for Implementing RAG Model for IBA Chatbot

In this section, we will explain how we implemented the RAG model to create an IBA University chatbot using three main stages:

1. Vector DB Setup Stage
2. Retrieval Stage
3. Generation Stage

By following these three stages, we were able to implement the RAG model for the IBA University chatbot, which can accurately and efficiently answer student queries using authoritative and pre-determined knowledge sources. In the following sections, we will provide more details on each stage and discuss the results of our implementation.

During the development of the IBA chatbot using the RAG model, we conducted various tests to evaluate the performance of the system and identify areas for improvement. Based on

the results of these tests, we made **several iterations** to the model to enhance its accuracy and efficiency.

Overall, the RAG model was improved over time through a series of tests, iterations, and evaluations, resulting in a highly accurate and efficient chatbot for IBA University.

Vector DB Setup stage

After extensive experimentation and trialing various approaches, we found that the most efficient and effective method was the third version of our implementation. Therefore, if you are interested in replicating our best methodology or recreating our results, we highly recommend focusing on and replicating only version 3.

Approach 1

Platform used: local machine (MacOS M1)

Data Source	Available	Cleaned & Structured	Original Format
PDF	Yes	No	Text
Tabular	No	No	CSV
Website	Yes	No	Json converted to Text

Figure 5: Data source and preprocessing done for approach 1.

1. RAG Read Docs From the Source

The RAG system begins the process by reading documents from the document source. In our case, we used all the PDFs of the data sources mentioned earlier, and we converted our API calls to text format that we obtained from the IBA website and faculty information. We then concatenated all this data into a single text file.

Important note: We made a crucial mistake here, which was turning the CSV files to a text file as it is. Refer it to data improvement section to see how we fixed it.

2. Send Docs to Chunk Function

The RAG system sends the documents to a chunking function. This step involves breaking down the documents into smaller pieces which are called “chunks”. We experimented with various chunking techniques, including character chunking, recursive splitting, and semantic splitting.

- Character Chunking: We tested different separators, chunk sizes (200, 600, 1000), and overlaps (100, 200).
- Recursive Split: We compared the performance of the recursive method with that of the character text splitter using the same chunk size and overlap. The only difference between the two methods was the list of separators used. Surprisingly, the character text splitter outperformed the recursive method in this comparison.

- Semantic Split: Semantic chunking was done on three breakpoint_threshold_types: 1. 'percentile' 2. 'standard deviation' 3. 'interquartile'. On average, it took more than 5 minutes to perform splits but did not outperform the character text splitter.

3. Send Chunks to Embedding Function

The RAG pipeline forwards the document chunks to an embedding function for further processing. In our experimentation, we consistently used GPT4AllEmbeddings. The primary function of the embedding process is to convert the chunks into numerical vectors, known as embeddings, that encapsulate the semantic meaning of the text.

4. Store Vectors in DB

Finally, these vectors are stored in a vector database.

Results:

The results were evaluated based on the number of test queries that successfully retrieved relevant information regarding to the query. The judgment was through **human evaluation**.

The results were not satisfactory.

Chunk_size, overlap	Character text splitter	Recursive text splitter Default separators ["\n\n", "\n", " ", ""]
1000, 200	2/6 with '.' 2/6 with '\n'	2/6
500, 100	2/6 with '\n' 3/6 with '.'	2/6
700, 200	2/6 with '.' 2/6 with '\n'	2/6

Figure 6: Retrieval results of approach 1.

Approach 2

Platform used: local machine (MacOS M1)

Data Source	Available	Cleaned & Structured	Original Format
PDF	Yes	No	Text
Tabular	Yes	Yes	CSV
Webiste	Yes	Yes	Json converted to text file

Figure 7: Data source and preprocessing done for approach 2.

1. RAG Read Docs From the Source

In approach 2, we followed the same process as in approach 1. However, we made a significant improvement in the preprocessing of website data. We transformed the data into a proper text script format to enhance the quality of the results. For more details on the data cleaning process, please refer to the Data Cleaning section,

specifically the subsection on Data through API. Finally, we concatenated all the data into a single text file.

2. Send Docs to Chunk Function

Same as version 1.

3. Send Chunks to Embedding Function

In Version 2, we continued to use GPT4AllEmbeddings for the embedding function, as we had in Version 1.

4. Store Vectors in DB

Finally, these vectors, along with associated metadata, are stored in a vector database (Document DB). This database is designed to facilitate efficient similarity search or retrieval operations. We tried several vector databases, including Chroma and Faiss.

Although we initially encountered compatibility issues with Chroma DB in the ensemble retriever, we analyzed most of our semantic searches on the Chroma DB. Ultimately, we decided to use Faiss as our primary vector database due to its compatibility with ensemble retriever and BM25 retriever.

Results

It was very apparent that if we increased the chunking size, the retrieval text chunk gave much more accurate results because even if the subject in the query was mentioned in a text, it gave the whole chunk size which eventually had its result in it. Hence, the LLM just needed to pick it accordingly.

The following table summarizes the results of our experiments:

Chunk_Size & Overlap	Character Split	Recursive Split	Semantic
1000 & 200	4/5 on "." & 3/5 on "/n"	3/5	3/5 for percentile 3/5 for std 3/5 for iqr
500 & 100	2/5 on "." & 1/5 on "/n"	2/5	2/5 for percentile 3/5 for std 2/5 for iqr
700 & 200	2/5 on "." 1/5 on "/n"	3/5	X

Figure 8: Retrieval results of approach 2.

Approach 3

Platform used: local machine (MacOS M1)

Data Source	Available	Cleaned & Structured	Original Format
PDF	Yes	Yes	Text
Tabular	Yes	Yes	CSV
Webste	Yes	Yes	Json converted to test file

Figure 9: Data source and preprocessing done for approach 3.

1. RAG Read Docs From the Source

In approach 3, we utilized the most cleaned and structured data available. One of the major improvements we made was in the cleaning of the PDF text files. We used a cleaning mechanism inspired by the C4 data set cleaning process. This resulted in the PDFs' extracted text files becoming more readable and properly formatted, which in turn improved the quality of our embeddings.

2. Send Docs to Chunk Function

The testing process for this version was conducted in a similar manner as it was for approach 1.

3. Send Chunks to Embedding Function

Similar to approach 1.

4. Store Vectors in DB

Finally, these vectors are stored in a vector database.

Results:

The results were better, and the data retrieved using this method was significantly cleaner than with the previous approach.

chunk_size & overlap	character split	recursive split	semantic
1000 & 200	4/5 on "." & 4/5 on "/n"	3/5	X
500 & 100	2/5 on "." & 2/5 on "/n"	2/5	X
700 & 200	3/5 on "." 2/5 on "/n"	3/5	X

Figure 10: Retrieval results of approach 3.

Semantic search was disregarded here because it was taking too much time.

Retrieval stage

User Submits Query

Platform used: local machine (MacOS M1)

The process begins with the user submitting a search query to the system.

Parrot

Platform used: Kaggle (local machine was taking too much time)

Here we also used the Parrot library to create similar paraphrased queries for better search and analysis. We integrated the Parrot paraphrasing tool to refine prompts and improve search effectiveness. We selected the top three paraphrased prompts with the highest relevance scores to the original query (using paraphrase-MiniLM-L6-v2 of Hugging Face).

```

model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
def calculate_similarity(sentence1, sentence2):
    # embeddings
    embeddings = model.encode([sentence1, sentence2], convert_to_tensor=True)
    cosine_similarity = util.pytorch_cos_sim(embeddings[0], embeddings[1]).item()
    return cosine_similarity

```

Figure 11: Code used for relevance between query and parrot phrases.

We conducted searches using all three prompts to increase the likelihood of retrieving relevant results.

```

Input Phrase: What is email of IT help desk?
Top 1 Paraphrase: what is an e-mail address of the it help desk?, Similarity: 0.8151271343231201
Top 2 Paraphrase: tell me the email address of the it help desk?, Similarity: 0.7678142786026001
Input Phrase: What are the eligibility criteria for candidates applying for BSCS program?
Top 1 Paraphrase: tell me the eligibility criteria for bscs program?, Similarity: 0.8950644731521606
Top 2 Paraphrase: tell me the eligibility criteria for the bcs program?, Similarity: 0.6597251892089844
Input Phrase: Is it possible to exempt IBA aptitude test?
Top 1 Paraphrase: is it possible to exempt an iba aptitude test?, Similarity: 0.9926439523696899
Input Phrase: What is the pre-requisite for Data structures?
Top 1 Paraphrase: what is a pre-requisite for data structures?, Similarity: 0.9973128437995911
Top 2 Paraphrase: what would be pre-requisite for data structure?, Similarity: 0.9460704922676086
Top 3 Paraphrase: tell me the pre-requisite for a data structure?, Similarity: 0.9059039354324341
Input Phrase: What is email of sir sajjad haider ?
Top 1 Paraphrase: what is sajjad haider's email address?, Similarity: 0.9311031103134155
Top 2 Paraphrase: what is sir sajjad haider's e-mail?, Similarity: 0.8933532238006592
Top 3 Paraphrase: tell me the email address of sri sajjad haider?, Similarity: 0.8884987831115723

```

Figure 12: Test questions and their parrot phrases ranked.

We needed reranking again because now each query had its own return chunks. We utilized the Coherence Rerankers API for reranking, which resulted in the reranker occasionally selecting the same query multiple times when the relevance score was very high. Although this reduced the diversity of documents, it improved the answering ability in our test cases. This was because the reranker often discarded uninformative documents in favor of higher-scoring duplicates, leading to a more relevant and informative document set.

```

try {
    const rerankedDocs = await coherenceRerank.compressDocuments(transformedDocs, origQuestion);
    console.log(`Original Question: ${origQuestion}`);
    console.log('Top 5 Documents:');

    writeStream.write(`Original Question: ${origQuestion}\n`);
    rerankedDocs.forEach((doc, index) => {
        console.log(`Document ${index + 1} Metadata:`, doc.metadata);
        writeStream.write(`Document ${index + 1} Page Content: ${doc.pageContent}\n`);
    });
}

```

Figure 13: Code snippet of coherence reranker.

We implemented a checking function to filter out any documents with a similarity score of less than 0.8. Consequently, when we attempted to search for criteria related to LUMS University, no documents were retrieved.

Script:
 Parrot similarity.ipynb
 coherenceRerank.js

Search Similar Vectors in DB

Platform used: local machine (MacOS M1)

The RAG system processes the user's query by converting it into an embedding vector using the same embedding function as before. This vector is then used to search for the most similar vectors in the document database (Vector DB). The system retrieves the top k results with the highest similarity scores to the user's query, where k is a configurable parameter. We experimented with different values of k and found that a value of 5 was optimal for our use case.

```
question = "What is the email of sir sajjad?"
#compute embeddings for the question
question_vector = GPT4AllEmbeddings().embed_query(question)
docs1 = char_db.similarity_search_with_score(question,k=5)
docs2 = char_db.similarity_search_with_relevance_scores(question,k=5)
docs3 = char_db.similarity_search_by_vector(question_vector,k=5)
docs4 = char_db.max_marginal_relevance_search(question,k=5)
```

Figure 14: Code snippet for search types.

BM25 Retriever

Platform used: local machine (MacOS M1)

In addition to the vector-based retrieval method, we also experimented with a BM25 retriever. Unlike the vector-based approach, BM25 does not rely on embeddings and instead searches for keywords within the documents.

For our test data, the BM25 retriever performed exceptionally well, particularly when the queries were specific and contained unique keywords or phrases. In fact, the **BM25 retriever often outperformed the vector-based retrieval** method in terms of accuracy and relevance. This can be attributed to the fact that BM25 is specifically designed to handle keyword-based searches and assigns higher relevance scores to documents that contain the exact query terms. As a result, it is able to quickly and efficiently identify the most relevant documents, even in large and complex datasets.

Results

The use of the Parrot library in our search process proved to be beneficial as it mapped keywords into similar wording, thus increasing the likelihood of retrieving the desired chunk even if the user did not input the exact keyword. However, this came at the cost of increased complexity and a longer pipeline running time.

After conducting extensive testing and experimentation, we found that a combination of both vector based and BM25 retrieval methods yielded the best results. This approach allowed us to capitalize on the strengths of both methods and ultimately improve the overall accuracy and efficiency of the system.

Furthermore, we experimented with different values of k for semantic search and found that a value of 5 was optimal for our specific use case.

Generation stage

The RAG system receives the reranked documents from the retrieval stage and limits them to the top k documents. In our experiments, we found that a value of 5 for k worked well for our use case.

Note: we had to keep the k documents <5 because the optimal chunking size was 1000 characters.

Long-Context reordering

The performance of our model is negatively impacted when it is presented with more than five long documents, each containing 1000 characters. This is because models tend to disregard the provided documents when attempting to extract relevant information from lengthy contexts. To avoid this issue, we reordered the documents after retrieval to improve the model's performance.

Combine The Query & The Documents

Platform used: Kaggle.

The RAG system then combines the original user query with the top k documents, using a specific format to ensure that the LLM can process the information correctly.

Finally, the combined query and documents are sent to a large language model (LLM), which generates an answer based on the information provided.

```
question = "What are the eligibility criteria for candidates applying for the BSCS program?"

prompt = f"""
Based on the following information:\n\n
{response_content}\n\n
Please provide a detailed answer to the question: {question}.
Your answer should integrate the essence of all the information, providing a unified answer that leverages the \
diverse perspectives or data points provided by all the information. \
Remember you are a chatbot for IBA university only.If the responses are irrelevant to the question then respond \
by saying that I couldn't find a good response to your query in the database.
"""
```

Figure 15: prompt used for text generation.

LLM used

Platform used: Kaggle.

Initially, we employed various language models (LLMs), including Mistral. However, we encountered issues running them due to recent updates in the gate token. Below are some of the LLMs that we utilized for generating results in Approach 3.

Outputs through Qwen/Qwen-1_8B-Chat :

Question	Result text	Time
What are the eligibility criteria for candidates applying for the BSCS program?	The eligibility criteria for candidates applying for the BSCS program vary depending on the university and its institution's specific requirements. However, in general, the eligibility criteria include: 1. Higher Secondary School Certificate (HSSC): All applicants must have completed their HSSC with a minimum of 60% marks. 2. Higher Secondary School Certificate (Pre-Engineering): This program is designed for students who wish to pursue engineering-related subjects after completing their HSSC. The program has different eligibility criteria based on the institution. Some institutions may require an English/写作 score for placement in English courses. 3. Higher Secondary School Certificate (General Group with Mathematics): This program is designed for students who wish to pursue mathematics-related subjects after completing their HSSC. The program has different eligibility criteria based on the institution. Some institutions may require an English/写作 score for placement in English courses. It's important to note that these eligibility criteria are subject to change and may vary from year to year, so it's best to check with the university directly for the most up-to-date information.	13.2622 seconds
What is the pre-requisite for the Data Structures course?	The pre-requisite for the Data Structures course is Object-Oriented Programming Techniques (CSE142), as it is mentioned in Document 1.	1.5297 seconds
What is the email address of Sir Sajjad Haider?	I apologize for any confusion caused by my previous answers. Based on the given information, the email address of Sir Sajjad Haider is sahaider@iba.edu.pk. This is also mentioned in Document 4, where it is stated that "Dr. Sajjad Haider works in the Department of Computer Science" and his email address is available.	3.2046 seconds
What is the email address of the IT help desk?	he email address of the IT help desk at IBA University is not explicitly mentioned in the given documents. However, based on the context, it can be inferred that Mr. Faraz Baig is the person responsible for IT help desk support for both Main Campus and City Campus. Therefore, the email address of the IT help desk for Main Campus could be "helpdeskmain@iba.edu.pk" and the email address of the IT help desk for City Campus could be "helpdesk@iba.edu.pk". The IT help desk team members available at both the campuses can be reached through these email addresses or through their respective phone numbers.	5.4149 seconds
Is it possible to exempt the IBA aptitude test?	Yes, it is possible to exempt the IBA aptitude test for some students. Exemption from the test does not mean that a candidate is automatically accepted into the IBA, but it provides them with an opportunity to showcase their skills and knowledge in a different way. The decision to exempt the test will depend on various factors such as the reason for exemption, the nature of the test, and the specific requirements of the candidate's program. It is recommended for candidates to carefully consider these factors before making any decisions.	4.3663 seconds

Figure 16: results for Qwen

Code:
Qwen.ipynb

Outputs through TheBloke/neural-chat-7B-v3-2-GPTQ:

Assistant:

The eligibility criteria for candidates applying for the BSCS program at IBA (Institute of Business Administration) vary slightly across different documents. In general, the main criteria include having completed their Higher Secondary School Certificate (Pre-Engineering or General Group with Mathematics) with a minimum of 60% marks in either option. Candidates should also pass the SAT I, ACT, or GRE tests. For the SAT I, they should have a score of 670 in Mathematics and 600 in Evidence-Based Reading and Writing, while for the ACT, they need a Composite Score of 28. Essay scores are not required, although they can be submitted for placement in English courses subsequent to admission. It's important to note that these criteria might differ slightly depending on the year or specific program.</s>

Time taken for generation: 206.17 seconds

Assistant:

The eligibility criteria for candidates applying for the BSCS program at IBA (Institute of Business Administration) vary slightly across different documents. In general, the main criteria include having completed their Higher Secondary School Certificate (Pre-Engineering or General Group with Mathematics) with a minimum of 60% marks in either option. Candidates should also pass the SAT I, ACT, or GRE tests. For the SAT I, they should have a score of 670 in Mathematics and 600 in Evidence-Based Reading and Writing, while for the ACT, they need a Composite Score of 28. Essay scores are not required, although they can be submitted for placement in English courses subsequent to admission. It's important to note that these criteria might differ slightly depending on the year or specific program.</s>

Time taken for generation: 206.17 seconds

Assistant:

Sir Sajjad Haider is a member of IBA Computer Science Faculty and works as a Professor in the Department of Computer Science. He specializes in Information Technology and is currently available. You can reach him at his email address, sahaider@iba.edu.pk.

Time taken for generation: 71.08 seconds

Assistant:

To reach the IT help desk, you can contact the following members based on their respective campuses.

Main Campus:

Mr. Faraz Baig

Email: helpdeskmain@iba.edu.pk and (or) mbaig@iba.edu.pk

Ext: 2101, 2120, 2114

City Campus:

Mr. Zeeshan Nasir

Email: helpdesk@iba.edu.pk and (or) znasir@iba.edu.pk

Ext: 1105, 1113

Please note that the Campus Management System is not mentioned in the provided information, so we couldn't find any specific help desk members for this.

Time taken for generation: 189.40 seconds

Assistant:

It is possible to exempt the IBA aptitude test under certain conditions. Students can avoid taking the exam if they have achieved certain minimum scores on standardized tests like SAT, GRE, or GMAT. For instance, for the undergraduate programs, students with SAT I scores of 670 in Mathematics and 600 in Evidence-Based Reading and Writing, or GRE scores of 750 in Quantitative and 600 in Verbal, can be exempted. Similarly, for the graduate programs, students with a GRE score of 310, GMAT score of 600, or 700 in the Analytical Writing Assessment can be exempted. Please note that this exemption is accompanied by additional requirements like the payment of the admission application processing fee, obtaining the admit card, and going through other admission stages like group discussion and interview. The information provided also highlights that the IBA aptitude test is not compulsory for students from BSCS program - SAT I, BSCS program - ACT, or BSCS program - SAT II. However, the key point is that it's possible to avoid the IBA aptitude test by meeting certain score requirements on other standardized tests.

Time taken for generation: 305.47 seconds

Results

If our retrieval and generation (RAG) model failed to find any relevant document, indicating no returned document or coherence, we prompted it to respond with "I don't have knowledge of such information." However, if documents were returned but lacked any required knowledge to answer query, the language model (LLM) indicated its lack of knowledge. Nevertheless, we observed instances where the model hallucinated answers, particularly when the required answer was long. However, for concise information, it performed admirably. This highlighted the effectiveness of our human evaluation process in providing relevant information overall.

Larger models exhibited improved overall performance with fewer hallucinations, but at the cost of increased time and memory requirements.

In our final analysis, our team have created a RAG model customized to cater to the specific needs of IBA students. The vision of this project is projected to decrease the burden of departmental email management, thereby enhancing operational efficiency. Furthermore, by facilitating prompt and effective responses to student inquiries, our model increases overall satisfaction levels and support within the IBA.

Drive Link for all codes

<https://drive.google.com/drive/folders/1U9ejlOy9OIL0u3ntDhTwSIfgxpO5nLTy>