HSERAG:
Optimizing University Administration with Retrieval-Augmented Generation

Ilya Shakhov

December 20, 2024

Abstract

This article explores how a RAG system can be used to create a chatbot that helps with university administration tasks. Using data from the Higher School of Economics, the chatbot is designed to answer questions accurately and provide useful information for students. The study compares different ways of finding relevant data, such as BM25, TF-IDF, BERT and combinations of these methods. It also looks at how multihop RAG performs compared to standard RAG, especially for answering tough questions. By analyzing how well the chatbot retrieves data, how efficient it is and how accurate its answers are, the study finds the best way to build a reliable chatbot for university use. These insights could improve how conversational AI is used in education, making information easier to access for everyone.

Introduction

This article examines the potential of using retrieval-augmented generation or (as they called) RAG systems to create chatbots tailored for university administration tasks. With a focus on the Higher School of Economics, the research explores how RAG can efficiently answer student questions, automate repetitive tasks and improve communication between students and administrative staff. The study evaluates various retrieval techniques, including BM25, TF-IDF and BERT, as well as their combinations, to determine the most effective approach for handling unstructured data. Additionally, it compares the performance of multihop RAG, which synthesizes multiple pieces of information, against standard RAG in addressing complex questions. By analyzing retrieval quality, efficiency and accuracy, this research aims to identify practical methods for building reliable and secure chatbot systems that can enhance university operations and adapt to increasing student demands.

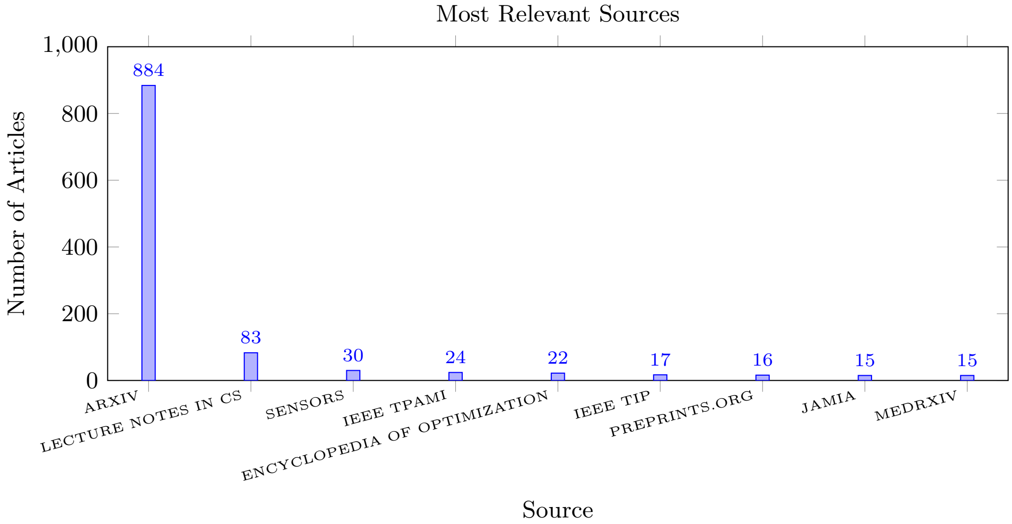
In this article, we will explore the following:

1. Review of existing research: we will look at current studies on chatbots and retrieval-augmented generation.
2. Gathering data for the knowledge base: we will review methods for collecting and organizing the information that the chatbot will use to answer questions.
3. Ways to search for information: we will analyze different retrieval methods, including bm25, tf-idf and bert-based approaches and compare their strengths and weaknesses.
4. Setup query classification mechanism.
5. Multihop approaches: We will discuss how multihop RAG can handle complex questions that require connecting multiple pieces of information and analyze which LLM can fit our task.

This research focuses on understanding how to build the best chatbot for answering university-related questions, with the ultimate goal of providing a practical and efficient tool for academic administration.

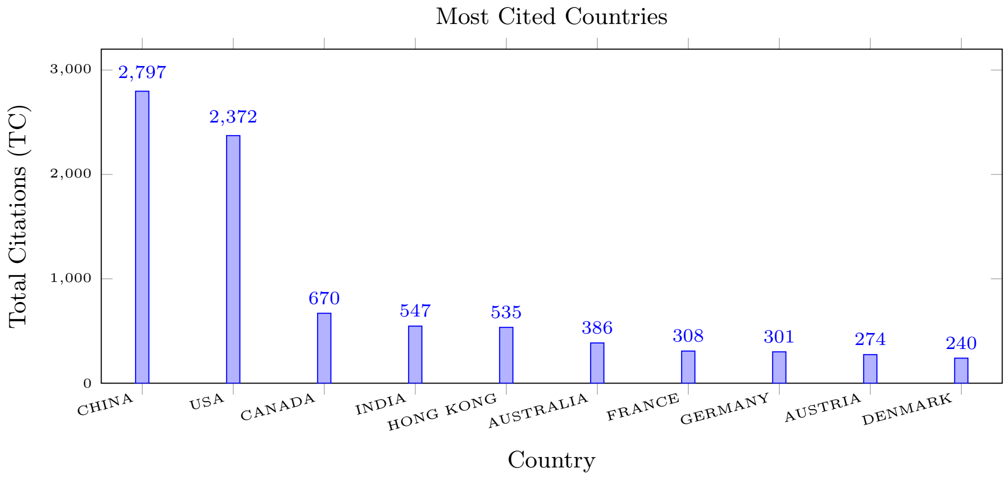
1 Literature analysis

For this project, we collected and combined many articles related to RAG. We used the Dimensions website and an R tool called Bibliometrix Biblioshiny [3] to gather data. Since Dimensions only allows downloading 500 articles at a time, we used Python (with pandas) to merge the tables. In total, we analyzed 2,081 articles.



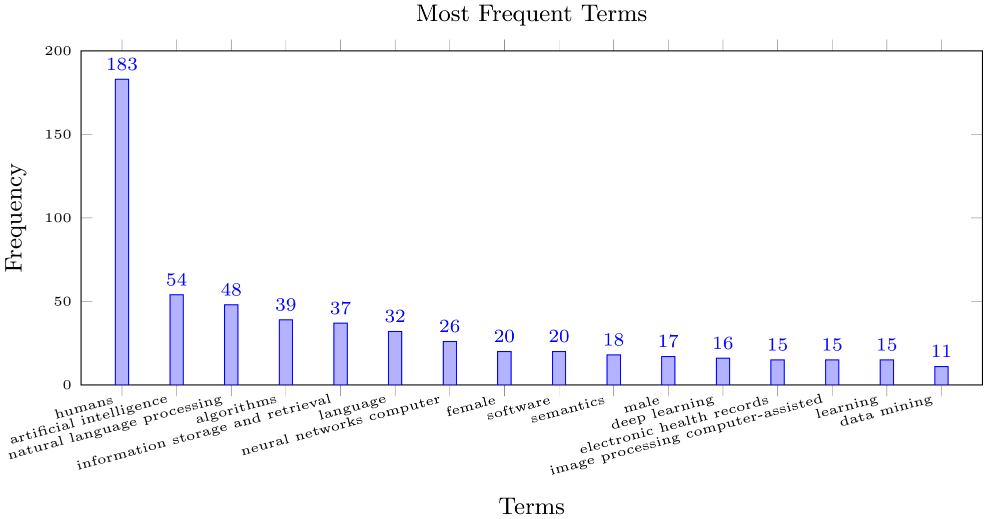
The data suggests that the most significant contributions to the field are hosted on repositories like arXiv, which dominates the references with a substantial number of documents - 884.The "Lecture Notes in Computer Science" series ranks second in relevance, emphasizing the academic and technical interest in RAG methodologies within the computer science community. Specialized sources like the "Journal of the American Medical Informatics Association" and "MedRxiv" suggest exploratory applications of RAG in niche domains such as healthcare and we found many RAG papers in the medical field, from conversation agent to a doctor assistant. This points to the flexibility of RAG techniques and their potential for adaptation beyond conventional chatbot use cases.

Diversity of sources in these references demonstrates that RAG research is rapidly evolving and spans multiple domains, from academic theory to real-world applications.



The most cited countries are China and the United States, with 2,797 and 2,372 citations respectively underscoring their dominant role in advancing artificial intelligence. Other notable contributors include Canada, India and Hong Kong, which collectively demonstrate strong engagement from both North America and Asia.

No Russian publications were identified in the dataset, suggesting a gap in local contributions to RAG research. Given the growing importance of AI-driven systems for administrative purposes, this absence highlights an opportunity for Russian researchers and institutions to engage more actively in this rapidly developing field.



"Humans" occupies the largest segment, reflecting the emphasis on user-centric applications and human-computer interaction within RAG-related studies. "Artificial intelligence" and "natural language processing" follow as prominent themes as a technological backbone driving advancements in retrieval and generation models.

Topics like "information storage and retrieval", "algorithms" and "neural networks" underscore the technical focus on optimizing the retrieval mechanisms and generative capabilities. Smaller segments such as "language" and "computer networks" illustrate the interdisciplinary scope of the research, integrating linguistic and computational perspectives.

Despite the growing interest in RAG, there is a surprising lack of research papers focusing in the area of educational and university-specific applications. Our analysis of multiple studies, including the work by Neupane et al. [11] and Nam et al. [10], highlights the scarcity of resources exploring advanced RAG techniques tailored to university environments. Even more striking is the near-complete absence of studies on multihop RAG approaches within this domain.

Multihop RAG, which involves chaining multiple retrieval steps to generate more contextually informed and accurate responses, holds immense potential for addressing complex, layered queries often encountered in academic contexts. However, current research seems to focus primarily on single-hop retrieval methods, leaving multihop strategies largely unexplored. This presents a unique opportunity for researchers to advance both the academic and practical applications of RAG technology.

2 Implementing RAG

2.1 Data collection

Data collection plays a crucial role in the effectiveness of RAG systems[9]. In such system, the model retrieves relevant documents from an external corpus and uses this information to produce responses that are contextually grounded.

The information required for RAG systems can often be obtained online through web scraping techniques. Many universities publish their documents directly on their official websites. This information is typically available in formats like *.html*, *.pdf* or plaintext.

For our implementation, we used Beautiful Soup, a Python library that simplifies web scraping by parsing *.html* documents [14]. Beautiful Soup helps in navigating the structure of a web page, makes it easier to find and extract specific content such as links or text. This makes it an effective choice for projects where continuous access to new data is necessary.

We made use of the large collection of regulatory documents available for download at `http://hse.ru/docs`. These documents provide detailed information about various university rules and processes. However we did not download all of them. Instead we selected only the most relevant documents by manually reviewing the content. This approach allowed us to keep our database clean as we needed to avoid the inclusion of unnecessary information.

We fully scraped the information available on the HSE Academic Handbook website at `https://www.hse.ru/studyspravka/`. This resource is a centralized guide containing summaries of all key academic processes and student-related activities at HSE. Using Beautiful Soup, we wrote a script to iterate through each link on the website and extract the content of each page. The main content of these pages is stored under the `post__text` class, which we targeted to retrieve only the relevant data.

Even after extracting the data, we faced challenges with text formatting. Many pages included special characters, unnecessary symbols that could disrupt our database. To address this, we normalized all the text to UTF-8 encoding. This process cleaned up the data by removing unwanted characters and ensuring it was uniform.

By combining manual selection for regulatory documents with automated scraping for the academic handbook, we were able to create a reliable and focused database.

The collected data was saved in two formats based on its source. Official documents were downloaded and saved in *.docx* format to maintain the original formatting and structure. Meanwhile, content scraped from webpages was normalized and stored in *.txt* format.

This method is quite simple and can be used by almost any university. It doesn't require advanced tools, just some basic knowledge of web scraping and data handling. We also used information for prospective students, which was created by university staff in *.docx* format. This information is not available on public websites, making it unique to our project.

We reviewed the content carefully to select only the most important documents. By doing this, we avoided including unnecessary files that could make the database harder to use. In the end our database contained 71 files.

This approach shows that with some effort and planning, universities can create focused and useful data collections for projects like this.

Effective data collection is critical for ensuring the success of RAG systems. The principle of "garbage in, garbage out" is particularly relevant here - if the data fed into the system is irrelevant, incomplete or poorly structured, the quality f generated responses will inevitably suffer.

2.2 Chunking

The quality of information retrieval directly impacts the relevance of generated responses. Good retrieval is important because it guarantees that the text generation is grounded in relevant data. Studies have shown that retrieval-augmented systems perform significantly better when the input is highly relevant and contextually appropriate [9].

To search through documents effectively, you first need to break them into smaller pieces and then compare a query to each piece of text. This makes it easier for the system to find the right information. Splitting documents into chunks also minimizes the risk of missing context due to the length constraints of retrieval or generation models [8]. We used a basic method where we split the documents into chunks of size n, with an overlap of k between them. Overlap ensures that boundary information is retained and improves the quality of retrieval.

The formula we used to determine the starting point of each chunk is as follows:

$$\text{Start of each chunk} = i \cdot (n - k) \quad \text{for } i = 0, 1, 2, \dots$$

Where:

- n = size of each chunk
- k = overlap between chunks
- i = chunk index (0, 1, 2, ...)

For our implementation, we set $n = 1000$ and $k = 100$. After splitting the documents using this method, the total number of chunks generated was 818. This chunking process helps the system handle large documents more efficiently and ensures that each query has access to all relevant pieces of text without missing important information due to document length.

Additionally, chunking methods like this are commonly used in dense retrieval systems, which rely on embedding-based approaches to compare queries with chunks in vector space [8]. By keeping chunks smaller and overlapping, the retrieval model can better align query vectors with relevant content, ultimately improving downstream text generation quality.

2.3 Retrieval

2.3.1 Embedding models

The most advanced way for information retrieval is using embedding models by converting our query text and documents to a vector representation and then computing similarity score (e.g.

using cosine similarity) to retrieve the most relevant documents. Embedding-based retrieval has gained popularity due to its ability to understand context and semantics. For example, if a student searches for "class schedules" the system can use an embedding model to find documents that talk about "timetables" or "course timings".

Several models can be used for embedding-based search, such as different variations of BERT [5], which is great at understanding the context of words in a query. It considers how each word relates to others in the sentence, making it ideal for searches where the meaning is important. Another model is Sentence-BERT [12], which extends BERT to handle entire sentences or paragraphs. In our study, we used ai-forever/sbert_large_nlu_ru [2], sergeyzh/LaBSE-ru-turbo [16] and cointegrated/rubert-tiny2 [4], which are specialized for the Russian language.

2.3.2 Classical methods of information retrieval

TF-IDF and BM25 Okapi are classical methods of information retrieval which are still being used. These methods work by measuring the importance of specific words in a document compared to their commonality across all documents.

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, assigns a weight to each term that increases with the term's frequency in a document but decreases with how common it is across all documents [17]. The TF-IDF formula is used to determine the importance of a term t in a document d relative to a collection of documents. It is defined as:

$$\text{TF-IDF}(t, d) = \frac{\text{Number of occurrences of } t \text{ in } d}{\text{Total number of terms in } d} \times \log\left(\frac{N}{\text{DF}(t)}\right)$$

- N Total number of documents in the collection
- $\text{DF}(t)$ Number of documents that contain the term t

BM25 Okapi builds on the principles of TF-IDF by incorporating the length of each document into its ranking process [15].

$$\text{BM25}(q, d) = \sum_{t \in q} \frac{\text{IDF}(t) \cdot f(t, d) \cdot (k_1 + 1)}{f(t, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)}$$

Where:

- q query, d document, t term in the query
- $f(t, d)$ frequency of term t in document d
- $|d|$ length of document d
- avgdl average document length in the collection
- k_1 term frequency scaling parameter
- b parameter for document length normalization
- $\text{IDF}(t)$ inverse document frequency of term t

While TF-IDF is valued for its simplicity and ease of implementation, BM25 Okapi offers more sophisticated handling document length and many other aspects, making it a preferred choice for many search engines.

2.3.3 Combining methods

In our study, we experimented with combining classical methods like BM25 Okapi with advanced embedding models. The goal was to use the strengths of both. BM25 Okapi provided precision and speed, while the embedding models offered a better understanding of context.

We combined their scores using weights to balance each method's contribution. Specifically, we used a weight of 0.6 for the embedding model score and 0.4 for the BM25 Okapi score.

Why did we try this approach? Because the text contained many "unknown" words for the embedding model, such as "маголего", "ЭПП"", "ДОЦ" and others. These are terms that the embedding model might not understand well due to their domain-specific or uncommon nature. By leveraging BM25 Okapi, which performs well with exact term matching, we aimed to mitigate some of the errors of the embedding model and improve overall retrieval performance.

2.4 Measuring retrieval

The main goal was to see how good each system was at finding useful answers to questions. To test this, we used a set of *60* questions that the student might ask, covering a range of topics. For each question, the system would give k answers. For our purposes k value of *3* suits us more. To measure the performance of each system, we used the F1 score. However, calculating recall for F1 score is difficult as we do not truly know how many relevant instances exist in the whole documentation.

Through our testing, we found that some systems were better than others at finding useful answers. The system that combined traditional keyword methods like BM25 Okapi with modern embedding models performed the best. They seemed to balance finding relevant keywords and understanding the deeper meaning of the questions and answers.

Information Retrieval Method	**F1 Score**
TF-IDF	0.4833
BM25 Okapi	0.5333
cointegrated/rubert-tiny2	0.3500
ai-forever/sbert_large_nlu_ru	0.3722
sergeyzh/LaBSE-ru-turbo	0.6889
sergeyzh/LaBSE-ru-turbo with BM25 Okapi (0.6/0.4 weights)	0.8000

Table 1: F1 Score for different information retrieval methods

2.5 Generating RAG Responses

2.5.1 Classifying user queries

We implemented a classification mechanism to distinguish between two types of queries those requiring RAG and those that can be answered directly. This classification is performed before the query enters the RAG pipeline, ensuring faster processing.

The classification is performed using a lightweight prompt-based approach with LLM. The prompt instructs the LLM to categorize the query into one of two categories - general or specific. "General" queries include greetings, small talk or any inputs that do not require access to external knowledge, such as "Привет" or "Как дела?". Furthermore, the mechanism treats potentially malicious queries, such as those containing inappropriate language or prompt injection attempts (e.g., <system> or <user> tags), as "general". Queries classified as "specific" require detailed answers based on external data, particularly those related to studying at HSE.

The classification process is designed to be robust and fast. The LLM receives a clear and concise prompt that defines the criteria for each category and explicitly guards against prompt injection by ignoring any suspicious tags in the query. The response from the LLM is stripped of unnecessary formatting and normalized to lowercase for comparison. If the LLM fails to produce a valid classification, the query defaults to "general" to prevent unnecessary RAG usage.

For queries classified as "general", the system generates a direct response without engaging the RAG mechanism. These responses are crafted using a separate prompt that ensures a friendly tone and reinforces system security. The prompt explicitly instructs the LLM to address the user's query politely and if offensive language is detected, to respond appropriately by informing the user that the system cannot process such requests. The prompt also reminds the user that the system can assist with questions related to studying at HSE.

This approach ensures that simple or potentially harmful queries are addressed quickly and securely while reserving the computationally intensive RAG pipeline for queries that genuinely require detailed information retrieval. This classification step significantly enhances system efficiency and security.

2.5.2 Multihop RAG

As classic RAG has been implemented many times, we decided to experiment with our version of multihop RAG.

In our implementation of multihop RAG, we introduced the concepts of steps and paths to make the system more dynamic and capable of exploring topics in depth and breadth. This approach allows for iterative reasoning and parallel exploration of a user's query, resulting in a more comprehensive and nuanced final answer, as we theorized.

Steps represent a sequential, iterative process. When a user asks a question, such as "Как отписаться от маголего?", the system retrieves relevant information for this query and generates an initial answer using LLM. If the parameter max_steps is set to a value greater than one, the system takes this initial answer and allows the LLM to generate a follow-up question aimed at deepening the discussion. For example, after answering the first query, the LLM might ask "Что полезного предоставляет блок маголего для студентов?". This follow-up question then triggers another round of the retrieval process, followed by another answer generation step. This cycle can continue indefinitely, depending on the value of max_steps, creating a chain of questions and answers that progressively explore the topic. After all the steps are completed, the system compiles all the questions and answers into a final prompt, along with the user's initial query, to generate a comprehensive response.

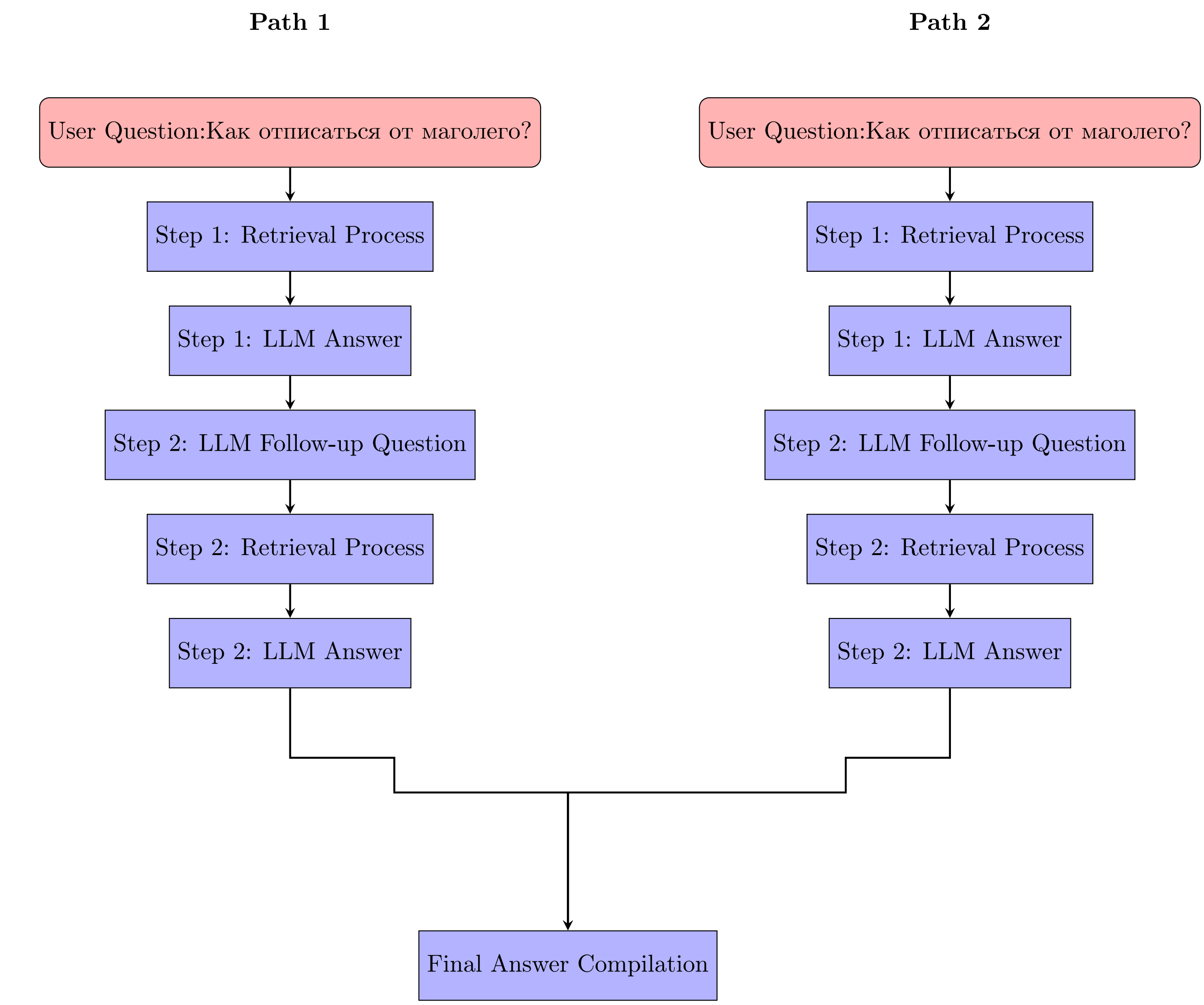
Paths, on the other hand, introduce parallelism into the process. Instead of focusing on one sequence of retrieval and answering, paths allow the system to explore the user query in different contexts. For instance, if max_paths is set to two, the user's initial question, "Как отписаться от маголего?", is processed twice. Each path starts with the same question but proceeds independently. The first path may generate one sequence of follow-up questions and answers, such as exploring the utility of the "маголего" block for students, while the second path might delve into whether this block is mandatory for students. Each path conducts its own retrieval and reasoning steps and the contexts from all paths are aggregated at the end. This parallel exploration helps uncover different aspects of the same queryt.

By combining steps and paths, our system achieves both depth and breadth in its analysis. For example, with steps set to 2 and paths set to 2, the system processes the initial user query in two parallel paths, each with two iterative steps. In the first path, after answering the initial query, the system may generate a follow-up question related to the benefits of "маголего". In the second path, the process begins again with the initial query but proceeds with a different follow-up question, such as whether the "маголего" block is mandatory. Both paths contribute their sequences of questions and answers to the final compilation step.

At the end of this multihop process, all the questions and answers generated across all steps and paths are included in a final prompt. This final prompt serves as the input for the LLM, which synthesizes all the information into a comprehensive answer to the user's original question.

Our multihop RAG system thus leverages the complementary strengths of sequential reasoning (steps) and parallel exploration (paths) to deliver high-quality, contextually rich answers. This architecture is particularly well-suited for complex, open-ended queries, where a single round of retrieval and answering might not meet the user's needs.

Scheme of RAG setup with 2 paths and 2 steps



2.5.3 Evaluation dataset

To test our theory, we introduced a custom dataset inspired by the MMLU format. Each entry in this dataset consists of a single question, four possible answers and one correct answer. Additionally, each question is assigned a complexity category ranging from 1 to 4. This complexity category is somewhat subjective, as it is determined by the individual creating the dataset. It reflects the perceived number of steps required to arrive at the correct answer. For example, a complexity of 1 might correspond to a simple query that can be resolved with a single google search, while a complexity of 4 would indicate a more challenging question requiring multiple steps to find the answer.

Example of evaluation dataset

```
[
  {
    "question": "Что такое маголего?",
    "options": [
      "Маголего - это система коммуникации для магистрантов",
      "Маголего - это система обязательных курсов для магистрантов",
      "Маголего - это система необязательных курсов для магистрантов",
      "Маголего - это факультативы для бакалавров и магистрантов ВШЭ"
    ],
    "answer": 1,
    " complexity": 1
  },
  {
```

```
    "question": "Как студент ВШЭ может уйти в академический отпуск?",
    "options": [
      "Необходимо подать заявление в учебный офис и предоставить необходимые документы",
      "Необходимо прекратить посещение занятий без уведомления",
      "Необходимо отправить электронное письмо преподавателю с просьбой о отпуске",
      "Необходимо запросить отпуск через социальные сети университета"
    ],
    "answer": 0,
    " complexity": 1
  },
  {
    "question": "Что такое ИУП в контексте ВШЭ?",
    "options": [
      "ИУП - Институт управления проектами",
      "ИУП - Индивидуальный учебный план",
      "ИУП - Информационно-управленческая платформа",
      "ИУП - Инициатива университетской поддержки"
    ],
    "answer": 1,
    " complexity": 1
  }
]
```

Using this dataset and slightly modifying the original RAG implementation to fit the changing test environment, we were able to evaluate how selected LLMs perform under varying parameters. In our testing environment, we could alter the number of steps and paths to observe their effects on performance.

However, our ability to conduct extensive research was limited by the computational time required. Our dataset contains 113 questions, a total of 38 questions for complexity of 1, 43 questions for complexity 2, 24 questions for complexity of 3 and 8 questions for complexity 4 and even with the simplest configuration of 1 step and 1 path, it takes approximately 20–30 seconds to generate an answer for a single question. This translates to at least 40 minutes to evaluate one set of parameters. Increasing the number of steps or paths significantly amplifies this time. For example, using 2 steps and 1 path doubles the runtime, while 2 steps and 2 paths quadruples it.

The testing loop begins by loading the dataset, which contains questions, their respective options, the correct answer index and complexity. For each parameter combination, every question is processed sequentially. The query is formatted to include the question text, all answer options and specific instructions for the LLM to respond only with the correct answer's index. This query is passed to the RAG.

The final answer is extracted using regular expressions to identify the number corresponding to the selected option. This answer is then compared to the correct index to determine accuracy. Additionally, the framework tracks performance across different complexity levels by recording how many questions of each level were answered correctly.

2.5.4 LLMs and testing

Our choice was to use any proficient language model capable of understanding and generating Russian while being lightweight enough to run on consumer-grade hardware. Specifically, we focused on models with fewer than 12 billion parameters to ensure feasibility for universities like HSE, which may not have access to high-end enterprise hardware. Given that HSE is known to have at least some RTX graphics cards, it is entirely possible to run these models without requiring significant additional investment in hardware. This approach makes it practical for academic institutions to adopt advanced language models without incurring excessive costs.

We tested two advanced language models to evaluate their performance in a Russian language environment. The models included Vikhr-Llama3.1-8B-Instruct-R-21-09-24 [19] and Saiga Gemma2 10B [7]. Each of these models is well-suited for Russian-language tasks and has been optimized for efficient comprehension and response generation within this domain.

The Vikhr-Llama3.1-8B-Instruct model [19] is based on the Llama3.1 architecture [1]. Saiga Gemma2 10B [7] builds on Gemma2's[13] architecture. Both models were run as quantized *.gguf* files, gemma with Q_6_K quantization and llama with Q_8.

Steps	Paths	Complexity 1	Complexity 2	Complexity 3	Complexity 4	Total Accuracy
1	1	73.68%	60.47%	66.67%	50%	65.49%
2	1	63.16%	55.81%	75%	37.5%	61.06%
1	2	86.84%	67.44%	75%	75%	76.11%
2	2	63.16%	48.84%	95.83%	50%	63.72%

Table 2: Accuracy results for Saiga_Gemma2_10B at different steps and paths with sbert_large_nlu_ru embedder and 0.6 coefficient for embeddings, 0.4 for BM25.

Steps	Paths	Complexity 1	Complexity 2	Complexity 3	Complexity 4	Total Accuracy
1	1	67.26%	65.12%	75.0%	50%	67.26%
2	1	60.53%	62.79%	79.17%	75.0%	66.37%
1	2	84.84%	67.44%	83.33%	87.5%	77.88%
2	2	60.53%	55.81%	79.17%	50%	61.95%

Table 3: Accuracy results for Saiga_Gemma2_10B at different steps and paths with sergeyzh/LaBSE-ru-turbo embedder and 0.6 coefficient for embeddings, 0.4 for BM25.

Gemma From our observations, the Gemma model proved to be the most stable in terms of following prompts and avoiding hallucination or introducing unrelated information. While this observation is subjective and could be influenced by the quality of our prompts, the Gemma model consistently stayed grounded in the context of the given questions. However, it occasionally made peculiar decisions, such as reasoning correctly about the answer but ultimately selecting a different option in its final response. There were also instances where it attempted to provide multiple answers instead of choosing just one. Despite these quirks, its stability and adherence to the task made it stand out among the models tested. The accuracy results for Saiga_Gemma2_10B at different steps and paths reveal insights into the impact of these parameters on model performance. Increasing the number of steps appears to introduce more hallucinations and reduces overall accuracy. For instance, with 2 steps and 1 path, the total accuracy drops to 61.06%, compared to 65.49% with 1 step and 1 path. This trend suggests that while additional steps provide the model with an opportunity to refine its answers, they also increase the likelihood of the model generating irrelevant or incorrect information.

On the other hand, increasing the number of paths appears to stabilize the model's performance and reduce hallucinations. With 1 step and 2 paths, the total accuracy significantly improves to 76.11%, the highest among all configurations. This indicates that parallel retrieval processes enhance the model's ability to ground its responses in relevant context, leading to better overall performance. Even with 2 steps and 2 paths, where steps introduce some instability, the model demonstrates better performance (63.72%) than with 2 steps and 1 path.

These findings suggest that increasing paths is more beneficial for this model, as it helps to stabilize and reduce the amount of hallucinated information, whereas increasing steps may degrade accuracy due to the additional complexity and opportunity for errors.

Also, the configuration with 2 steps and 2 paths showed an interesting pattern in its performance. It struggled with basic questions, particularly those of lower complexity but excelled at answering more challenging ones, such as those with complexity 3. This suggests that a dynamic approach, where the number of steps and paths is adjusted based on the perceived difficulty of the question, could potentially optimize performance. However, implementing such an approach is challenging, as accurately assessing the hardness of a question in real-time remains a difficult task.

Sometimes, the LLM provided the correct answer but failed to include the required keyword "Финальный ответ:", which meant the answer could not be parsed correctly and was therefore marked as incorrect. The LLM occasionally changed the format of the answer options from 0, 1, 2, 3 to 1, 2, 3, 4, or even to formats like a, b, c, d, or other variations. While such discrepancies could be dismissed and counted as correct with significant manual checking, this approach introduces two major issues.

First, it highlights that the LLM does not strictly follow the instructions provided in the prompt, which is problematic. For our conversational HSE University chatbot, adhering to prompts is essential to ensure both security and effectiveness in its responses. Second, the need for manual verification undermines the scalability and efficiency of the system. As a result, answers that deviate from the expected format are counted as incorrect, even if they are factually correct.

We can clearly observe the use of chain of thought reasoning by the Gemma model which was not detected in Llama model. The model provides a structured explanation, breaking down the question into smaller logical steps to reach a conclusion. This approach not only demonstrates the model's ability to understand and analyze the context but also highlights its capability to trace the reasoning process systematically.

The chain of thought reasoning is particularly evident in how the model first explores the context of the question, then verifies the relevant information and finally evaluates the consequences of each choice. This structured methodology ensures that the model's final answer aligns with the given instructions and is grounded in the provided context.

Here is an example of the model's work:

Вопрос:Можно ли студент ВШЭ не записываться на факультатив по английскому языку? Варианты ответов:

0. Да, можно не записываться, но независимый экзамен по английскому языку нужно сдать.

1. Нет, факультатив обязателен для всех студентов.

2. Можно, если студент владеет другим иностранным языком.

3. Да, можно не записываться, независимый экзамен по английскому языку в таком случае сдавать не нужно.

Выбери ТОЛЬКО ОДИН вариант ответа, выбирать несколько нельзя и ответь только номером правильного ответа после слов Финальный ответ:.

Ответ: 1. Изучаем контекст: "Студент может вовсе не выбирать бесплатный факультатив по английскому языку и выбрать путь самостоятельного изучения". 2. Проверяем, есть ли в тексте информация о том, что это возможно или нет: "в платных языковых школах, самостоятельно с помощью онлайновых ресурсов, на курсах дополнительного профессионального обучения в НИУ ВШЭ." 3. Учитываем последствия выбора такого пути: "В этом случае он в обязательном порядке должен будет подтвердить свою готовность учиться в дальнейшем на английском языке, а именно сдать промежуточные и внутренние экзамены на первом курсе и внешний независимый экзамен на втором курсе."

Финальный ответ: 0.

This example illustrates the reasoning depth of the Gemma model and its ability to adhere to the provided instructions.

We also conducted experiments with the this model using two different embedders: sbert_large_nlu_ru and sergeyzh/LaBSE-ru-turbo. Both embedders were paired with a 0.6 coefficient for embeddings and a 0.4 coefficient for BM25 to balance the contributions of dense and sparse retrieval methods.

The results, as presented in the tables, show that the performance was nearly identical across most configurations. While LaBSE-ru-turbo demonstrated a slight advantage in overall accuracy, particularly in the more challenging categories such as complexity 4, the difference was marginal. For instance, with 1 step and 2 paths, the total accuracy for LaBSE-ru-turbo reached , compared to 76.11% with sbert_large_nlu_ru. This aligns with findings from our previous benchmarks, where `LaBSE` consistently showed a better performance.

Despite this small advantage, the overall results indicate that both embedders are robust and effective for RAG tasks within this setup. However, the marginal improvement seen with LaBSE-ru-turbo may make it a preferred choice in scenarios where precision is critical.

Steps	Paths	Complexity 1	Complexity 2	Complexity 3	Complexity 4	Total Accuracy
1	1	57.89%	60.47%	83.33%	50%	63.72%
2	1	52.63%	55.81%	41.67%	37.5%	50.44%
1	2	84.21%	55.81%	66.67%	37.5%	66.37%
2	2	55.26%	46.51%	58.33%	25%	50.44%

Table 4: Accuracy results for Vikhr-Llama3.1-8B-Instruct-R-21-09-24 at different steps and paths.

Llama Llama 3.1 was bad at creating questions for itself for the steps. Usually, it was not really a question but something akin to a prompt to make a question or thoughts about what needs to be found in order to answer the question or it filled the questions with unnecessary information like comments on the context and so on. Llama 3.1 felt more wordy but we didn't change any parameters. It loved to repeat the prompt information and, as if to prompt itself, which felt out of place. Some of the strange comments also included emojis, as the question generated by the LLM somehow had emojis. While its wordiness comes into play when it should find the proper answer, it was worse to use than Gemma as a chat agent for this specific task.

Another notable issue with Llama 3.1 is its inability to reliably generate an EOS token, which is a common problem in Llama 3.1 fine-tunes using SFT. This limitation prevents the model from predicting the loss of the eot_id token. To address this, we opted to use '- - -' as a custom stop token. This issue is described in detail in discussions on Llama.cpp's and Unsloth's repositories [6] [18].

Additionally, the model occasionally switches to English during its outputs, which can disrupt the consistency of responses in a Russian-language environment. Such inconsistencies further highlight challenges in fine-tuned versions of Llama 3.1 when applied to specific tasks requiring strict adherence to instructions and localized language contexts.

The performance of Vikhr Llama 3.1 across different configurations of steps and paths highlights notable trends in its ability to handle questions of varying complexity. With 1 step and 1 path, the model achieves a total accuracy of 63.72%, performing particularly well on complexity 3 questions 83.33% but struggling with complexity 4 50%. This setup shows that a simpler configuration allows the model to maintain focus on moderately challenging tasks, although its performance declines for higher complexity levels.

When increasing to 2 steps with 1 path, the total accuracy drops significantly to 50.44%. This setup introduces notable hallucinations, as the model struggles with properly formulating its own questions. The effect of these hallucinations is most evident in the accuracy for complexity 3 and 4 questions, which fall to 41.67% and 37.5% respectively. The additional step appears to increase the model's inability to follow prompts and it generates responses that deviate from the required structure.

Adding a second path, as seen in the configuration with 1 step and 2 paths, improves the total accuracy to 66.37%, the highest among all configurations. The improvement in complexity 1 questions 84.21% suggests that multiple paths stabilize the model's performance on simpler queries. However, the model still struggles with complexity 4 questions, where accuracy remains low at 37.5%. This indicates that while paths help reduce hallucinations, they do not fully address the challenges of handling more complex questions.

With 2 steps and 2 paths, the total accuracy again falls to 50.44%, with complexity 3 and 4 accuracies at 58.33% and 25%, respectively. This configuration combines the destabilizing effect of steps with the stabilizing effect of paths, resulting in inconsistent performance. The increased number of steps further amplifies the model's tendency to generate irrelevant or incorrect intermediate questions, diminishing its effectiveness.

These results show that steps introduce significant hallucinations for this model, especially due to its inability to correctly formulate intermediate questions. For this model, the configuration with 1 step and 2 paths offers the best balance between simplicity and accuracy.

Conclusions Overall, we preferred the Saiga_Gemma2_10B model with the sergeyzh/LaBSE-ru-turbo embedder over Vikhr Llama 3.1. The combination of Gemma and LaBSE provided better consistency and slightly higher accuracy, making it a more reliable choice for our tasks. However, we plan to test more models in the future to explore additional improvements.

Using paths instead of steps generally increased consistency and improved the accuracy of the answers. For example, with 2 paths, the model often demonstrated a more grounded approach to retrieval and generation. That said, increasing the number of paths to 3 in the Gemma model did not yield further improvements, indicating a limit to the benefits of parallel processes in this setup.

We believe that steps could still be viable if better grounding from the previous prompts is incorporated to reduce confusion. In the current setup, steps sometimes created conflicting situations, where the model generated inconsistent intermediate answers. Addressing this issue by ensuring clearer context between steps may unlock their potential for more complex multi-hop reasoning tasks.

3 Limitations

While this research provides valuable insights into the development of a RAG chatbot tailored for university administrative tasks, it is not without limitations. First, the computational resources available for testing were limited, which restricted the scale of experimentation. The testing process, especially for configurations with multiple steps and paths, was time-consuming and constrained our ability to explore a broader range of parameters. This also impacted the ability to evaluate certain advanced configurations that may have further improved the system's performance or test other LLMs.

Another limitation lies in the dataset used for evaluation. While the dataset included diverse questions relevant to the university context, its size and complexity categories were subjective and might not fully represent the list of queries that users would encounter in real-world applications.

Also the reliance on manually curated data sources and web scraping methods as a RAG database introduced potential biases.

The performance of the LLMs used in this study was another limiting factor. Despite their overall effectiveness, inconsistencies in following prompts and formatting answers occasionally led to errors even when the reasoning was correct. Addressing these issues requires either refining the prompt design or developing custom fine-tuned models both of wich were beyond the scope of this project.

The study focused primarily on retrieval and response generation but did not fully explore user experience factors such as response time, usability and scalability under high demand. These aspects are critical for real-world deployment and should be investigated in future research to ensure the practical applicability of the system.

4 Conclusion

This study explored the potential of RAG systems for building an effective chatbot tailored to university administrative tasks. By combining traditional information retrieval methods, such as BM25 and TF-IDF, with advanced embedding models and incorporating multihop strategies, we sought to create a system capable of handling both simple and complex queries in a dynamic academic environment.

Our findings demonstrated that while both steps and paths have their place in multihop RAG systems, paths significantly outperform steps in enhancing performance. The introduction of multiple paths allowed the system to explore same or different contexts for a given query, leading to more stable and accurate results. In contrast, increasing the number of steps often introduced additional complexity and a higher likelihood of hallucinations or irrelevant information, reducing overall accuracy. This highlights the importance of parallel exploration over iterative reasoning in contexts where retrieving relevant, grounded responses is critical.

The results also underscored the value of combining classical and modern retrieval methods. Pairing BM25 with embedding models provided a balanced approach, leveraging BM25's precision for domain-specific terminology and the embedding models' contextual understanding. This combination proved particularly effective in mitigating the challenges posed by uncommon or ambiguous terms.

Despite these successes, the study revealed several areas requiring further refinement. Issues such as the computational cost of complex configurations, better utilization of setp mechanism, the occasional misalignment between prompts and outputs and the limited scalability of the current system underline the need for continued research and development.

In conclusion, this research not only validates the promise of RAG systems for educational applications but also provides a roadmap for future advancements.

References

[1] Meta AI. Llama-3.1-8b-instruct, 2024. URL https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct. Accessed: 2024-10-23.

[2] AI-Forever. ai-forever/sbert_large_nlu_ru, 2024. URL https://huggingface.co/ai-forever/sbert_large_nlu_ru.

[3] Massimo Aria and Corrado Cuccurullo. Bibliometrix: An r-tool for comprehensive science mapping analysis. *Journal of Informetrics*, 11(4):959–975, 2017.

[4] Cointegrated. cointegrated/rubert-tiny2, 2024. URL https://huggingface.co/cointegrated/rubert-tiny2.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019. URL https://arxiv.org/abs/1810.04805.

[6] ggerganov. Llama.cpp discussion on eos token issues, 2024. URL https://github.com/ggerganov/llama.cpp/discussions/8176. Accessed: 2024-12-19.

[7] Ilya Gusev. saiga_gemma2_9b, 2024. URL https://huggingface.co/IlyaGusev/saiga_gemma2_9b.

[8] Vladimir Karpukhin et al. Dense passage retrieval for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[9] Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[10] Yoonji Nam, TaeWoong Seo, Gyeongcheol Shin, Sangji Lee, and JaeEun Im. Novi: Chatbot system for university novice with bert and llms. *arXiv preprint arXiv:2409.06192*, 2024. URL https://arxiv.org/abs/2409.06192.

[11] Subash Neupane, Elias Hossain, Jason Keith, Himanshu Tripathi, Farbod Ghiasi, Noorbakhsh Amiri Golilarz, Amin Amirlatifi, Sudip Mittal, and Shahram Rahimi. From questions to insightful answers: Building an informed chatbot for university resources. *arXiv preprint arXiv:2405.08120*, 2024. URL https://arxiv.org/abs/2405.08120.

[12] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019. URL https://arxiv.org/abs/1908.10084.

[13] Google Research. Gemma-2-9b-it, 2024. URL https://huggingface.co/google/gemma-2-9b-it. Accessed: 2024-10-23.

[14] Leonard Richardson. *Beautiful Soup Documentation*, 2024. URL https://www.crummy.com/software/BeautifulSoup/.

[15] Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu. Okapi at trec-3. In *TREC*, 1995.

[16] sergeyzh. sergeyzh/labse-ru-turbo, 2024. URL https://huggingface.co/sergeyzh/LaBSE-ru-turbo.

[17] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1), 1972.

[18] unslothai. Github issue discussing model behavior, 2024. URL https://github.com/unslothai/unsloth/issues/416. Accessed: 2024-12-19.

[19] Vikhr. Vikhr-llama3.1-8b-instruct-r-21-09-24, 2024. URL https://huggingface.co/Vikhrmodels/Vikhr-Llama3.1-8B-Instruct-R-21-09-24.