Article

Design and Implementation of an Interactive Question-Answering System with Retrieval-Augmented Generation for Personalized Databases

Firstname Lastname 1, Firstname Lastname 2 and Firstname Lastname 2,\*

|  |
| --- |
| **Citation:** To be added by editorial staff during production.  Academic Editor: Firstname Lastname  Received: date  Revised: date  Accepted: date  Published: date    **Copyright:** © 2024 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/). |

1 Affiliation 1; e-mail@e-mail.com

2 Affiliation 2; e-mail@e-mail.com

**\*** Correspondence: e-mail@e-mail.com; Tel.: (optional; include country code; if there are multiple corresponding authors, add author initials)

**Featured Application: Authors are encouraged to provide a concise description of the specific application or a potential application of the work. This section is not mandatory.**

**Abstract:** A single paragraph of about 200 words maximum.

**Keywords:** Retrieval-augmented generation (RAG); GPT; large language model (LLM); personalized knowledge database

1. Introduction

In recent years, large language models (LLMs) and natural language processing have revolutionized the artificial-intelligence field by leveraging large datasets and powerful computing resources. OpenAI's generative pretrained transformer (GPT) model series is one of the most prominent LLM models, with the first version, GPT-1, released in 2018, demonstrating high performance in natural language processing and generation tasks using the transformer architecture and transfer-learning techniques. Subsequently, GPT-2 significantly scaled up the model and amount of training data, expanding its contextual understanding and sentence-generation capabilities. GPT-3, released in 2019, was trained on billions of parameters and large datasets, and demonstrated the ability to understand and generate considerably more complex and diverse information than previous models [1–3].

These developments have had a profound impact on various natural language processing applications and LLMs are now being used for complex tasks such as automatic translation, question-answering, document summarization, and content generation in diverse fields, including healthcare, education, and science [4–8]. Although these pretrained LLMs can produce increasingly realistic text, their ability to access and accurately manipulate knowledge remains limited [9]. Additionally, they cannot clarify their decision-making process, which is known as the black-box problem. Therefore, the accuracy and authenticity of their results are unknown and updating them with new data remains challenging. Thus, although LLMs offer convenience to users, they may generate inappropriate or erroneous responses in certain situations [10–12].

Lately, LLM developers are aiming to address issues such as hallucinations, lack of updates, and lack of answer transparency through retrieval-augmented generation (RAG) [13]. This technique combines knowledge from the field of natural language processing and LLMs with external knowledge databases to enhance the quality and relevance of their responses. RAG is particularly useful in scenarios wherein specific and up-to-date information is required, such as academic research, customer service, or content creation [14–18].

RAG offers the following advantages for information-retrieval applications. Modern data environments comprise vast amounts of information, and fast and accurate search is essential to utilize the data effectively. Additionally, the demand for retrieving accurate and relevant information is increasing. Therefore, RAG can be used for faster updates and personalized searches in LLMs, where information is stored in parameters. RAG allows retrieving the necessary information without using sensitive data. Thus, RAG is key for in providing personalized search capabilities in information-retrieval services not only for companies and institutions but also for individuals.

In this study, a personalized database system was implemented using search augmentation, and keywords set by individuals through question-answering (QA) were tagged based on their context. This is similar to structuring information into categories, such as date and topic, based on context. Individuals using this search-enhancement system are provided a personalized database of keywords and contextual layers. However, a simple implementation of LLM prompts and outputs is insufficient to use this system. Therefore, we applied an RAG process for context-based search enhancement and implemented a NoSQL database to continuously update the search histories of users. Thus, we implemented a personalized database and QA system, and verified its performance through the retrieval augmented generation assessment (RAGAs) platform [19].

This study focuses on Internet web services that tag keywords within personal documents and use this information to search for personal documents in a personalized database (i.e., searching through notes in a document). On this platform, the text entered by logged-in individuals or referenced from external documents is stored in a personalization database with relevant keywords. Thus, the documents that an individual or team wants to retain are structured and tagged with specific keywords and updated in the database. This facilitates an interactive search for embedded personal information by understanding which documents are relevant and quickly analyzing the content within them.

To this end, RAG is applied to identify the context of personalized knowledge, such as vector databases, and generate accurate information from a personal database. By generating results through an interactive search across documents created and stored by an individual, we demonstrate the feasibility of a personalized database system using RAG.

Because the RAG retrieves a variety of information from its knowledge base, it does not require additional model training or fine-tuning. Therefore, the RAG-based personalized search system developed in this study can generate accurate answers through accumulated information, regardless of the training data. It can also reflect the latest data in real-time, which can reduce costs and use resources more efficiently. These features offer users a more intuitive and efficient search experience, increase their satisfaction, and allow them to easily obtain the required information.

The remainder of this paper is organized as follows. Section 2 presents related work, Section 3 describes the development of the RAG implemented in this study, Section 4 presents the practical service methods and shows how they can be used in conjunction with LLMs for designing applications that provide the desired results and applies the RAGAs framework to evaluate their performances and response times. Finally, Section 5 concludes the study and presents future research directions.

2. Related Work

2.1. Hallucinations in LLMs

Although LLMs have induced innovative developments in the field of natural language processing, they suffer from the problem of hallucinations, which refers to producing false information, logical errors, and non-existent data that are coherent but not based on actual facts. For example, they can provide information about a person or event that does not exist.

Hallucinations include actual hallucinations, which include non-factual information; logical hallucinations, which include logically inconsistent or senseless answers; and structural hallucinations, which include incorrectly or inappropriately formatted responses. Recently, this issue has become more prominent owing to ChatGPT's tendency to generate plausible-sounding false information. Hallucinations can have serious implications because they contribute to the spread of misinformation and undermine the credibility of the content generated by LLMs [20–24].

To test this phenomenon, a study [24] entered 54 prompts from different domains into ChatGPT, wherein ChatGPT 3.5 and 4 achieved overall success rates of approximately 61.3 and 72%, respectively, indicating hallucination rates of 39 and 28 %, respectively. Although these results were obtained through prompt-dependent experiments, they indicate the hallucination rates of LLMs.

The causes of hallucinations vary, and the following factors have been cited as major contributors [20–28]:

* **Training-data bias:** Most LLMs are trained on large amounts of textual data collected from the internet. These data inevitably contain incomplete or biased information, increasing the likelihood of the models learning wrong patterns.
* **How the model makes inferences:** LLMs work by predicting the next word; thus, they attempt to generate the most plausible word for a given context, which may lead them to generate out-of-context of untrue information.
* **Interaction limitations:** LLMs generate information through interactions with human users, which may be limited; therefore, the responses generated based on such information may be inaccurate and obtaining accurate information may be challenging.

Thus, by abstracting information from prompts, LLMs can output hallucinations by overlooking important details, and distortions caused by bad training data can result in unexpected answers. Therefore, considerable research has been conducted to investigate and mitigate the causes of LLM hallucinations.

The language generation capabilities of LLMs have also been used for programming, and psychedelics have been studied in software development on code generation using LLMs.

Liu et al. [22] conducted a thematic and systematic analysis of LLM-generated codes to summarize and classify the hallucinations and their distribution through HALLUCODE, a benchmark for evaluating hallucinations in LLM code generation, and showed that existing LLMs can barely mitigate hallucinations. They suggested hallucinations can be mitigated by implementing additional tools or enhancements during the generation process of an LLM.

Chen et al. [25] proposed an evaluation model, called ReID, for detecting trustworthy answers generated by LLMs and successfully detected hallucinations in various LLM-generated responses. In this study, we propose conviction detection to analyze and understand the types of hallucinations in LLM-generated answers.

We still cannot fully trust their answers because LLMs suffer from hallucinations that manipulate non-existent facts to trick you without realizing it.

Yao et al. [26] demonstrated that meaningless prompts comprising random tokens could induce LLMs to respond to hallucinations. By making small changes to the input data, the model can be induced to make incorrect predictions and recall that hallucinations may be another perspective for adversarial cases. We propose a strategy to evaluate the robustness of LLMs and suggest that it is necessary to design defenses before practical applications.

Zhang et al. [27] showed that LLM hallucinations do not occur only because of lack of knowledge but can also occur when the model tries to justify its initial mistakes, suggesting that a more comprehensive understanding of LLM hallucinations is required to develop effective mitigation strategies.

Varshney et al. [28] focused on the effects of “negation” on LLM hallucinations and conducted experiments on LLama-2-Chat, Vicuna, and Orca-2 using negation representations, such as false premise completion and constrained fact generation, and found that these models produce significant hallucinations in tasks involving negation. Hallucinations associated with negative representations are a major drawback of LLMs, and further research is required to address them.

The primary cause of hallucination in LLMs that, unlike databases and search engines, LLMs generate text by making inferences from the input prompts. Their primary purpose is to generate answers based on the information they have already been trained on; therefore, they are unable to use new data and may provide incorrect answers. Lack of ability to accurately cite sources. In this study, we implemented search augmentation in a personalized database service and used the RAG to address the problems of hallucinations, lack of updating, and unclear sources in the existing ChatGPT model.

2.2. Research on RAG

The typical process of an RAG and the differences between implementing LLMs with and without RAG are shown in Figure 1 [13–15]. Consider a scenario wherein a user asks ChatGPT about the number of foreign students at a university. Because ChatGPT relies on pretrained data, it initially lacks the ability to provide the most recent data. As shown on the left side of Figure 1, it cannot provide the exact number and only outputs abstract information. By contrast, RAG allows it to retrieve knowledge from external databases; therefore, it can refer to the university database and provide the exact number. Thus, RAG improves the search performance of ChatGPT, similar to providing it with a personalized textbook for information retrieval.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**Figure 1.** Difference between the processing of a large language model (LLM) with and without retrieval augmented generation (RAG).

Chen et al. [15] analyzed the various components of RAG. Recently, modular RAG techniques that increase the flexibility of RAG by incorporating new modules and other techniques such as fine-tuning has emerged. For example, the implementation and advantages and disadvantages of naive, advanced, and modular RAG models were analyzed for RAG applications.

Zhao et al. [16] introduced practical RAG applications and benchmarks by categorizing the fundamental processes of RAG and investigating various search- and generator-augmentation methodologies. They investigated the impact of search augmentation on LLMs and analyzed the RAG performance in terms of noise, negative rejection, information integration, and semantic robustness. We experimented with four separate testbeds based on the aforementioned basic capabilities for evaluating RAG in English and Chinese. The results showed that LLMs exhibit a certain level of noise robustness but still suffer from significant difficulties in negative rejection, information integration, and handling false information. These results suggest the need for designing applications more deeply and using databases to enhance the benefits of RAG. In this study, we implemented an application system that can fully leverage RAG by building and retrieving personalized information.

Xia et al. [17] integrated domain refinement and RAG to implement a QA method for providing accurate information on typhoon disasters. They trained the T5 LLM on typhoon disaster information from open-source databases, such as Baidu Encyclopedia and Wikipedia. The RAG module was employed to enhance answers to user prompts by retrieving semantically similar phrases from external knowledge bases. They evaluated the typhoon agent (Typhoon-T5) using a similarity-matching approach and laid the foundation for integrating LLMs and disaster information. Thus, the employing RAG in conjunction with LLMs can effectively improve the search performance for a specific topic.

Li et al [18] reviewed search-augmented text-generation and other notable approaches for various text-generation tasks, including dialog-response generation and machine translation. They summarized the various components of search-augmented text-generation, including search metrics, search sources, and integration paradigms, and provided useful information for developing application-specific topics during RAG.

Thus, the efficacy of compensating for the shortcomings of LLMs, such as hallucinations, and improving search efficiency is being actively investigated. This study demonstrates that RAG can be effectively implemented to build a personalized database that provides differentiated search and QA services, along with improved search accuracy.

3. Implementation of RAG Pipeline

3.1. System Overview

This section describes the methods used in the proposed system, as illustrated in Figure 2.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**Figure 2.** System architecture.

The proposed system was designed to help users effectively manage personalized text documents. It offers users the ability to select and save specific sentences from a document, along with the associated tags and links. This is accomplished through a system called the tagging box [29], which allows users to map the text from a part of a document of interest to tagged keywords and save it as a personal archive. This represents a customized knowledge base categorized for the specific purposes of individuals or teams. This is part of a large-scale knowledge management engine, wherein many documents required by individuals or teams are organized and stored and the keywords used to tag specific sentences act as notes; this technology was developed by the authors of this paper. The aim was is to employ the RAG pipeline for designing a personalized database construction and retrieval technique to develop an information retrieval technique that without the disadvantages of existing LLMs.

A well-established semantic space is required for employing RAG to compensate for LLM hallucinations and ensure that appropriate and accurate answers are retrieved by the LLM during QA. Tagging services attempt to make RAGs more active and build a personalized semantic space for successful answer generation.

The proposed RAG pipeline effectively processes user questions and generates relevant answers using three main components. The first is an SQL database that stores personalized documents and information based on the identity of an individual. The user-information table in this SQL database manages the information of registered users through a “Tag Box” system. When a question is entered into the user interface of the QA system, the SQL database extracts information from the table that matches the user ID and splits it into chunks for further processing.

The second component is a vector database that takes the data chunks extracted from the SQL database and converts them into embedding vectors to reconstruct information. By storing data as vectors, vector databases can be used to handle unstructured data. To process user queries, vector databases use a similarity search between vector data, which offers the advantage of returning results more flexibly than using exact matches to queries. That is, vector embedding of information allow transforming data from a high-dimensional space to a low-dimensional vector. Although the data dimensions are reduced, the important information and data patterns are preserved. This allows computers to effectively analyze data and identify similarities or patterns between vectors.

We also implemented MongoDB [30] to store the chat history of QAs and generate answers that users can use later. As an LLM does not store the state, it does not remember the previous messages in a conversation. The developer is responsible for maintaining the history and providing context for the LLM. Prior contextual information can be stored in a persistent database and used to restore the context in new conversations, allowing scenarios wherein the questions and answers of users can be summarized and tracked back in history.

Finally, the QA generator utilizes an LLM, such as GPT3.5 Turbo, to generate accurate and useful answers to questions. This process is performed based on the context selected from the vector database, and the final answer is returned to the user.

3.2. Data Extraction: RAG and LangChain Integration

An easy method for implementing RAG is to employ LangChain [31], a powerful framework that integrates external tools to create an environment. This subsection details the implementation of RAG and a data-extraction technique that incorporates LangChain. The process involves the following steps:

* **Extract data from the SQL database:** In this step, relevant data are extracted from the user-information table through queries. This includes information related to documents, such as those tagged by the user.
* **Chunking and embedding:** Theextracted data are partitioned into chunks using LangChain's integrated framework and sent to a vector database (retriever), where embeddings are created. These embeddings convert the document content into high-dimensional vectors, improving information retrieval and matching.
* **Generating answers:** When a user inputs a prompt into the system, the relevant context is retrieved from the stored vector database and used as input to generate the optimal answer. LangChain manages the flow used in this process.

LangChain is a software development kit (SDK) that simplifies the integration of LLMs and their applications and is becoming increasingly important as the use of LLMs is increasing. It can segment, combine, and filter documents. The data are collected from an established SQL database through an API, returned in the JavaScript Object Notation (JSON) format, and structured as key-value pairs, as illustrated in Figure 3. The unique identifier number corresponds to a particular SQL database table and generates user information in the form of titles and tag names. The main information used to build a personalized database is the number of entries that contain tagging information, such as keywords, set by individuals to categorize documents. We call this a "tagbox,” and, as mentioned earlier, it is available as an Internet web service. In practice, a tagging box is implemented as a hyperlink to reference personalization information.

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

**Figure 3. U**ser data extracted in the JavaScript Object Notation (JSON) format (User ID: 2190).

Figure 4 shows that Context, which stores the contexts created or referenced by individuals and extracted from the SQL database, contains most of the document content, and based on this information, it sets the maximum length of the document to 1000 characters and splits the document for processing. The JSON-based TextSplitter takes directly takes the extracted data as input, selectively extracts and combines the required data, and returns them in the JSON format. The split document is divided into chunks of a certain size, each designed to be processed independently. After splitting, the documents are embedded via the text-embedding-ada-002 model of the OpenAI API and stored in a vector database. These data help place the most relevant information at the front of the QA prompt after a search.

Dembedding w

3.3. Role and Importance of Prompting Instructions

In the proposed system, user prompts are crucial as they directly affect its ability to respond effectively to user questions. This section explains the importance of providing appropriate prompts. Figure 5 illustrates the process of handling questions and generating responses. Based on the retrieved questions, the context of each question was established and organized as a “prompt” template.

텍스트, 도표, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명

**Figure 5.** Process of handling user prompts and generating answers using LangChain with Mongo DB.

Prompts act as an interface between user questions and LLMs and used as the basis for generating answers. The proposed system obtains a list of documents extracted via LangChain, formats them into prompts, and passes them to the LLM. The ability of an LLM to generate contextually appropriate answers is highly dependent on the quality and structure of the prompts provided. They play a pivotal role in ensuring that the answers are not only accurate but also relevant to the questions. Figure 6 shows an example of the prompt template developed in this study. When prompting, strictly adhere to the following principles.

* **Analyze context:** The entire context provided in the prompt must be considered to generate the answer. This ensures the accuracy of the answer, minimizes the transmission of misinformation, and aligns the response with the intent of the question.
* **Limit information:** any information that is not specified in context must not be included in the answer. This ensures that the answers are generated based solely on the entered data and prevents data leakage as the system can access tables that are not relevant to the question.
* **Cite sources:** State the source of information for every answer to allow the user to understand the origin or refer back to the information in the tagging box.
* **Recognize uncertainty:** If the answer is not available, inform the user that they should search for tags or content in the specified box to obtain more accurate results.

This creates a prompt, as shown in Figure 6, and the prompt generated for use in LLMs is shown in Figure 7.

스크린샷, 멀티미디어, 스마트폰이(가) 표시된 사진

자동 생성된 설명

**Figure 6.** Custom prompt template.

스크린샷, 텍스트, 멀티미디어, 전자 기기이(가) 표시된 사진

자동 생성된 설명

**Figure 7.** Example of a custom prompt generated for the QA task.

3.4. History Management: Mongo DB

The QA system developed in this study was implemented to manage the chat transcripts using MongoDB, which is a NoSQL-based database that stores key-value data in the JSON format. As it does not have a fixed schema, it can handle different types of data quickly and flexibly. The system takes the questions, retrieves the histories of five recent conversation from the MongoDB, and inputs them into the LLM along with QA prompts. Subsequently, the data are extracted using a predefined SQL query. The LLM analyzes the conversation history and generates contextually appropriate responses, which are then delivered to the user and stored in MongoDB. The flexible data-processing capabilities of MongoDB allow the system to store the history of user interactions and generate customized responses based on them.

4. Experiments

텍스트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

**Figure 8.** Typical ChatGPT and personalized RAG-based responses.

The left side of Figure 8 shows a user prompt for a muffin recipe and the output of ChatGPT 3.5, which is does not reflect the personality and intentions of the user. Because the output is based on already learned parameters, it does reflect the context; therefore, the LLM simply generates the answer by analyzing the prompt. By contrast, as shown in the right side of Figure 8, the context-based response generated through RAG from the personalized database of the “tag box” implemented in this system, that is, the muffin recipe created and updated by the individual is output, and the context information that is only relevant to the individual is displayed. To validate the differentiation of this system from traditional LLMs, we used the RAG performance evaluation metrics to verify the results.

4.1. RAGAs Framework

We used the RAGAs framework [19], which is a framework that focuses on evaluating the retrieval and generation capabilities of RAG systems, to evaluate the performance of the proposed system. The evaluation of each component of the RAG pipeline can be divided into two parts: answer generation and document retrieval.

The generation process shown in Figure 9 comprises two metrics: faithfulness, which evaluates the relevance between the retrieved documents and generated answers, and answer relevance, which evaluates the relevance of the generated answers to the questions. In the search process, the documents retrieved for a question are evaluated based on context precision and recall.

스크린샷, 블랙, 디자인이(가) 표시된 사진

자동 생성된 설명

**Figure 9.** RAGAs evaluation.

**Table 1.** RAGAs evaluation metrics.

|  |  |
| --- | --- |
| **Metric** | **Equation** |
| Faithfulness |  |
| Answer Relevancy |  |
| Context Precision |  |
| Context Recall |  |

is the embedding of generated question , is the total number of generated questions, is the embedding value for the original question, and k is the total number of chunks.

The faithfulness metric evaluates the consistency of the generated answers for providing relevant information based on the given context. It is calculated by comparing the generated answer with the context, with a higher probability indicating a more reliable answer. To calculate faithfulness, a set of assertions in the generated answer is first identified and then each assertion is crosschecked against the given context to determine whether the assertions can be inferred from the context.

Answer relevance indicates the relevance of the generated answer to an initially posed question. It evaluates the extent to which the answer meets the requirements of the question, with a high score indicating a complete and clear answer to a given question without redundant or unnecessary information. To calculate this, we used the LLM to generate multiple appropriate questions for the generated answers and evaluated their relevance by measuring the average cosine similarity between the generated and original questions.

Context precision indicates the percentage of retrieved documents containing content relevant to a question. It is used to evaluate the accuracy with which a search system curates and presents relevant documents to users. Context recall comprehensively evaluates whether the retrieved documents contain the information required to formulate the answer to a given question. It evaluates the performance of the search system during document retrieval by assessing whether the document contains sufficient background information and the correct answer to the question.

4.2. Experimental Setup

The data used in the experiment were evaluated using the commonsense dataset provided by AIHub [32] as the personalized database of 'Tag Box.” This dataset comprises 100 “Question,” “Ground\_Truth,” and “Context” items and is organized into three columns with these headings. The “Question” column contains the questions generated for the experiment, whereas the “Ground\_Truth” column contains the actual fact-based correct answers to each question. The “Context” column comprises textual data stored by users and serves as the basis for the questions and answers. The contexts listed in Table 2 are examples of those in a personalized document describing the ancient walls surrounding Jinju; this document is stored in a personalized database. Based on this, we evaluated the search performance of the RAGAs framework. We used the NVIDIA RTX 3070 GPU and Anaconda environment for LLM development and the Ollama library [33] for experimentation. The versions of RAGAs, Chroma DB [34], the vector database used in this study, and LangChain, which supports the RAG evaluation process, are detailed in Table 3.

**Table 2.** Data classification in the evaluation experiment.

|  |  |  |
| --- | --- | --- |
| **Question** | **Ground Truth** | **Context** |
| What is the area of Ibanseong-myeon? | 19.41 km² | Ibanseong-myeon is the center of transportation, culture, education, and commerce for the five eastern towns, and it has long been commercially developed, with the Banseong traditional market thriving day by day. The area is 19.41 km², making it the smallest of the 16 towns and districts in Jinju City. However, the nearby Gyeongnam Forest Environment Research Institute attracts tourists who pass through Ibanseong-myeon. As of January 1, 2012, the population was 3,233 (male: 1,556, female: 1,677) with 1,413 households, comprising 6 legal districts, 19 natural villages, and 31 neighborhoods. |
| Who discovered Cape Verde? | Portuguese navigators | The Republic of Cape Verde (Cabo Verde), also known as Cape Verde in English, is a country located in the Atlantic Ocean off the west coast of Africa and discovered by Portuguese navigators. Although it was uninhabited at the time of discovery, the islands had been visited since ancient times by Phoenicians, Arabs, Moors, and nearby West African tribes. Subsequently, Portuguese settlers began moving to this rare tropical region and establishing settlements. |

**Table 3.** System environment employed for RAG development and evaluation.

|  |  |
| --- | --- |
| **Computing** | **Version** |
| CPU | AMD Ryzen 9 5900X 12-Core Processor |
| RAM | 32 GB |
| GPU | NVIDIA GeForce RTX 3070 |
| Anaconda Python | 3.9.19 |
| Ollama | 0.2.5 |
| RAGAs | 0.1.9 |
| Chroma DB | 0.4.23 |
| LangChain | 0.2.8 |

4.3. Performance Analysis for Various Model Combinations

To implement the RAG, we used Gpt3.5-Turbo as the LLM model and text-embedding-ada-002 as the embedding model to verify the entire process, from user query to answer generation. In this section, we describe the reconstruction of the RAG process using various combinations of five LLMs and two embedding models to verify the scalability of RAG systems across different LLMs. Specifically, we used Gpt-3.5-Turbo, Gemma-2-9b [35], Llama-3-8B [36], Mistral-7B [37], and Qwen2-7B [38] LLMs, and OpenAI's “text-embedding-ada-002” and the local Korean “snunlp/KR-SBERT-V40K-klueNLI-augSTS” [39] embedding models. These models were analyzed using ten performance metrics.

Figure 10 shows the accuracies of the generated results for each LLM and embedding module combination, wherein Ada-002 + Gpt-3.5-Turbo exhibits the highest accuracy of 0.51. This is significantly higher than those of the other model combinations, indicating its reliability. The combinations using Llama-3-8B, Ada-002 + Llama-3-8B and KR-SBERT + Llama-3-8B, also showed high accuracies of 0.46 and 0.45, respectively, suggesting that Llama-3-8B offers a higher accuracy than the other models.

-

Figure 11 illustrates the response relevance. Similar to the accuracy results, the Ada-002 + Gpt-3.5-Turbo combination exhibits the highest score of 0.86, demonstrating its superiority to other model combinations in terms of relevance. By contrast, the Ada-002 + Llama-3-8B and KR-SBERT + Qwen2-7B combinations exhibit lower relevance scores of 0.39 and 0.42, respectively, suggesting that these combinations require improvements in terms of answer relevance. Although most model combinations obtained relevance scores of approximately 0.5, they performed relatively poorly.

-re

Figure 12 shows the context-recall scores for each model combination, wherein the KR\_SBERT + Gemma-2-9b, KR-SBERT + Llama-3-8B, and KR\_SBERT + Mistral-7B combinations exhibit the highest recall scores of 0.82, indicating their effectives in recalling information for a given context. By contrast, the Ada-002 + Gpt3.5-Turbo combination shows a relatively low recall score of 0.67, suggesting that this combination has weak information-recall capability.

-r

Figure 13 shows the context-precision scores for all models, wherein the Ada-002 + Gpt3.5-Turbo and Ada-002 + Qwen2-7B combinations exhibit the highest precision of 0.79, indicating that these combinations can provide highly accurate information for a given context. By contrast, the KR-SBERT + Mistral-7B combination shows a relatively low precision of 0.63, suggesting that this combination requires improvements in terms of contextual precision.

-

Additionally,

**t**he response times of all combinations were analyzed. Figure 14 shows a comparison of the processing time per data item for each model combination. Among the KR-SBERT combinations, KR-SBERT + Gpt3.5-Turbo exhibited the fastest processing time of 0.92 s. As KR-SBERT performs inference locally, it has a higher processing speed. By contrast, KR-SBERT + Gemma-2-9b exhibited the slowest processing time of 5.57 s, which may have been caused by the large size of Gemma-2-9b. Among the Ada-002 combinations, Ada-002 + Gpt3.5-Turbo exhibits a relatively fast processing time of 1.39 s, suggesting that this combination is capable of fast processing even under API communication. By contrast, Ada-002 + Gemma-2-9b shows the slowest processing time of 6.16 s. This difference is attributed to the fact that KR-SBERT is employed locally, whereas Ada-002 is requires API communication.

tdithe various LLM and embedding combinations

Overall, Ada-002 + Gpt3.5-Turbo obtained the best performance, outperforming others across several performance metrics, including accuracy, answer relevance, and contextual precision, and exhibited the second lowest processing time. Moreover, the KR-SBERT + Gpt3.5-Turbo combination exhibited the lowest processing time, suggesting that it a useful alternative. These results provide important information for selecting the optimal model by comparing the performances and processing times of different model combinations, and can help identify the model combination that can simultaneously offer optimal performance and efficiency.

4. Discussion

5. Conclusions

In this study, we developed an information-retrieval system for LLMs from personalized datasets using an efficient RAG approach. RAG combines the parameterized knowledge of LLMs with non-parameterized external knowledge to mitigate the hallucination problem and cite sources to increase output transparency and user trust. RAG can be customized for specific domains by indexing the relevant text corpora.

Additionally, we built a personalized database and designed an RAG approach to optimize data privacy and accuracy while minimizing costs. The latest personal information was updated in the database without retraining the LLMs and the search history was retained to develop a personalized QA system.

Existing databases can be leveraged as extensions of LLM knowledge, eliminating the need to retrain or fine-tune them on sensitive personal and workplace information. The database is updated in real-time and includes documents that can be modified or deleted by individuals, thereby ensuring consistent tracking and update of information.

The proposed RAG-based system demonstrated good performance in terms of accuracy, response relevance, and context relevance within the RAGAs framework. However, context-recall score was rather low, which will be addressed in future studies. Thus, the experiment verified the deficiencies of LLM response systems combined with the RAG pipeline, demonstrating the difficulty of using contextual information. However, this limitation can be overcome by using generalized hallucinations and updating the models with the latest information.

6. Patents

This section is not mandatory but may be added if there are patents resulting from the work reported in this manuscript.

**Supplementary Materials:** The following supporting information can be downloaded at: www.mdpi.com/xxx/s1, Figure S1: title; Table S1: title; Video S1: title.

**Author Contributions:** For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used “Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.” Please turn to the [CRediT taxonomy](https://img.mdpi.org/data/contributor-role-instruction.pdf) for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

**Funding:** Please add: “This research received no external funding” or “This research was funded by NAME OF FUNDER, grant number XXX” and “The APC was funded by XXX”. Check carefully that the details given are accurate and use the standard spelling of funding agency names at https://search.crossref.org/funding. Any errors may affect your future funding.

**Institutional Review Board Statement:** In this section, you should add the Institutional Review Board Statement and approval number, if relevant to your study. You might choose to exclude this statement if the study did not require ethical approval. Please note that the Editorial Office might ask you for further information. Please add “The study was conducted in accordance with the Declaration of Helsinki, and approved by the Institutional Review Board (or Ethics Committee) of NAME OF INSTITUTE (protocol code XXX and date of approval).” for studies involving humans. OR “The animal study protocol was approved by the Institutional Review Board (or Ethics Committee) of NAME OF INSTITUTE (protocol code XXX and date of approval).” for studies involving animals. OR “Ethical review and approval were waived for this study due to REASON (please provide a detailed justification).” OR “Not applicable” for studies not involving humans or animals.

**Informed Consent Statement:** Any research article describing a study involving humans should contain this statement. Please add “Informed consent was obtained from all subjects involved in the study.” OR “Patient consent was waived due to REASON (please provide a detailed justification).” OR “Not applicable.” for studies not involving humans. You might also choose to exclude this statement if the study did not involve humans.

Written informed consent for publication must be obtained from participating patients who can be identified (including by the patients themselves). Please state “Written informed consent has been obtained from the patient(s) to publish this paper” if applicable.

**Data Availability Statement:** We encourage all authors of articles published in MDPI journals to share their research data. In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Where no new data were created, or where data is unavailable due to privacy or ethical restrictions, a statement is still required. Suggested Data Availability Statements are available in section “MDPI Research Data Policies” at https://www.mdpi.com/ethics.

**Acknowledgments:** In this section, you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments).

**Conflicts of Interest:** Declare conflicts of interest or state “The authors declare no conflicts of interest.” Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. Any role of the funders in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results must be declared in this section. If there is no role, please state “The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results”.

**Appendix A**

The appendix is an optional section that can contain details and data supplemental to the main text—for example, explanations of experimental details that would disrupt the flow of the main text but nonetheless remain crucial to understanding and reproducing the research shown; figures of replicates for experiments of which representative data is shown in the main text can be added here if brief, or as Supplementary data. Mathematical proofs of results not central to the paper can be added as an appendix.

**Appendix B**

All appendix sections must be cited in the main text. In the appendices, Figures, Tables, etc. should be labeled starting with “A”—e.g., Figure A1, Figure A2, etc.

References

1. Author 1, A.B.; Author 2, C.D. Title of the article. *Abbreviated Journal Name* **Year**, *Volume*, page range.
2. Author 1, A.; Author 2, B. Title of the chapter. In *Book Title*, 2nd ed.; Editor 1, A., Editor 2, B., Eds.; Publisher: Publisher Location, Country, 2007; Volume 3, pp. 154–196.
3. Author 1, A.; Author 2, B. *Book Title*, 3rd ed.; Publisher: Publisher Location, Country, 2008; pp. 154–196.
4. Author 1, A.B.; Author 2, C. Title of Unpublished Work. *Abbreviated Journal Name* year, *phrase indicating stage of publication (submitted; accepted; in press)*.
5. Author 1, A.B. (University, City, State, Country); Author 2, C. (Institute, City, State, Country). Personal communication, 2012.
6. Author 1, A.B.; Author 2, C.D.; Author 3, E.F. Title of Presentation. In Proceedings of the Name of the Conference, Location of Conference, Country, Date of Conference (Day Month Year).
7. Author 1, A.B. Title of Thesis. Level of Thesis, Degree-Granting University, Location of University, Date of Completion.
8. Title of Site. Available online: URL (accessed on Day Month Year).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.