applied sciences

MDPI

Article

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

Jaeyeon Byun [1], Bokyeong Kim [1], Kyung-Ae Cha [1,*] and Eunhyung Lee [2]

[1] Department of Artificial Intelligence, Daegu University, Gyeongsan 38453, Republic of Korea; 9722jayon@gmail.com (J.B.); qhrud418@gmail.com (B.K.)
[2] Textway Inc., A02 Unicorn Lab., 5th Fl, 111, Oksan-ro, Buk-gu, Daegu 41593, Republic of Korea; eh.lee@textway.net
* Correspondence: chaka@daegu.ac.kr; Tel.: +82-53-850-6641

Abstract: This study introduces a novel approach to personalized information retrieval by integrating retrieval augmentation generation (RAG) with a personalized database system. Recent advancements in large language models (LLMs) have shown impressive text generation capabilities but face limitations in knowledge accuracy and hallucinations. Our research addresses these challenges by combining LLMs with structured, personalized data to enhance search precision and relevance. By tagging keywords within personal documents and organizing information into context-based categories, users can conduct efficient searches within their data repositories. We conducted experiments using the GPT-3.5 and text-embedding-ada-002 models and evaluated the RAG assessment framework with five different language models and two embedding models. Our results indicate that the combination of GPT-3.5 and text-embedding-ada-002 is effective for a personalized database question-answering system, with potential for various language models depending on the application. Our approach offers improved accuracy, real-time data updates, and enhanced user experience, making a significant contribution to information retrieval by LLMs and impacting various artificial intelligence applications.

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

check for updates

Citation: Byun, J.; Kim, B.; Cha, K.-A.; Lee, E. Design and Implementation of an Interactive Question-Answering System with Retrieval-Augmented Generation for Personalized Databases. *Appl. Sci.* **2024**, *14*, 7995. https://doi.org/10.3390/app14177995

Academic Editor: Alessandro Di Nuovo

Received: 30 July 2024
Revised: 31 August 2024
Accepted: 5 September 2024
Published: 6 September 2024

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 LLMs, 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 expanded the model's capabilities, and GPT-3, with billions of parameters, further enhanced its ability to generate complex and diverse information [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].

Currently, LLM developers are addressing 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 where specific and up-to-date information is required, such as academic research, customer service, or content creation [14–18].

Modern data environments comprise vast amounts of information, and rapid and accurate searches are 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 facilitates retrieval of the necessary information without using sensitive data. Thus, RAG is the key to 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.

The main contributions of this study are as follows:

- We designed and developed a method to integrate an interactive QA system with the actual SQL database of a company that provides personalized semantic tagging services. This system leverages RAG to reflect user-specific database modifications in real time. By reusing previously embedded data, the system reduces costs by avoiding the need to embed data each time. The proposed system offers a personalized database experience that dynamically updates based on user interactions.
- We present the results of performance evaluation experiments applying various state-of-the-art LLMs, providing valuable reference data for the appropriate selection of LLMs in designing RAG architecture-based QA systems. According to our experimental results, as evaluated by the RAG framework, the combination of GPT-3.5-Turbo and our custom template applied to the system demonstrated the most impressive performance, highlighting the importance of an optimized RAG design.

The remainder of this paper is organized as follows. Section 2 discusses 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 discusses the study results and outlines 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 investigate this phenomenon, Hanna et al. [24] entered 54 prompts from different domains into ChatGPT, resulting in ChatGPT-3.5 and ChatGPT-4 achieving overall success rates of approximately 61% 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 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.

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-3.5 model.

2.2. Research on RAG

The typical process of an RAG and the differences between implementing LLM 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.

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 have 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. They 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.

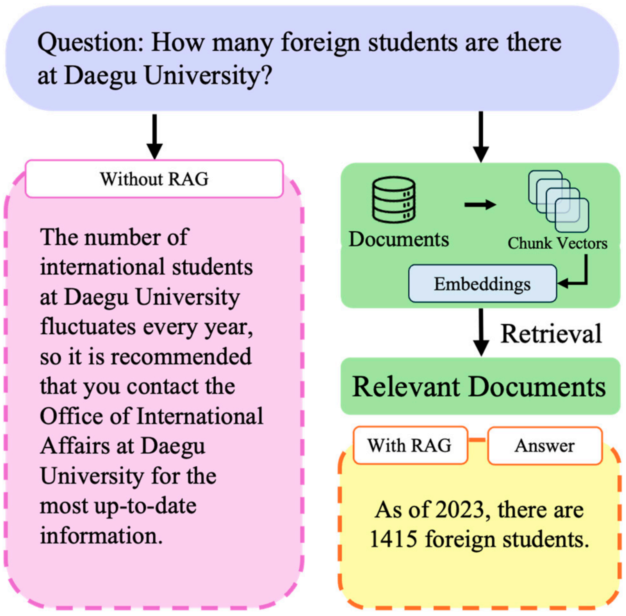

Question: How many foreign students are there at Daegu University?
Without RAG
The number of international students at Daegu University fluctuates every year, so it is recommended that you contact the Office of International Affairs at Daegu University for the most up-to-date information.
Documents
Chunk Vectors
Embeddings
Retrieval
Relevant Documents
With RAG
Answer
As of 2023, there are 1415 foreign students.

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

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, 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. Materials and Methods

3.1. System Overview

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

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 to employ the RAG pipeline for designing a personalized database construction and retrieval technique to develop an information-retrieval technique without the disadvantages of existing LLMs.

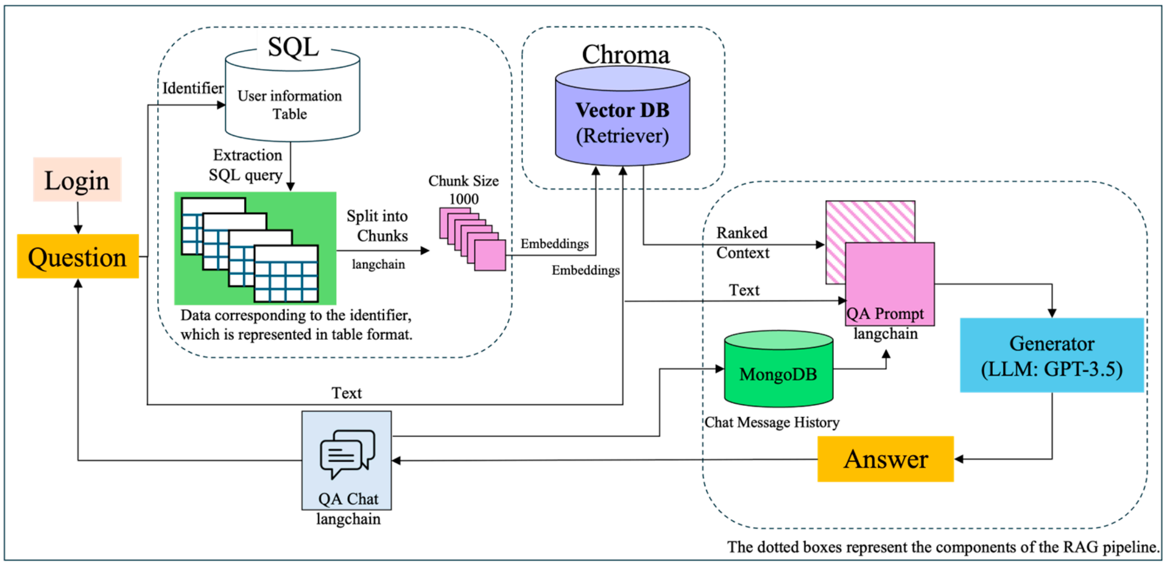


Figure 2. System architecture.

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 "TaggingBox" 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, the vector embedding of information allows the transformation of 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.

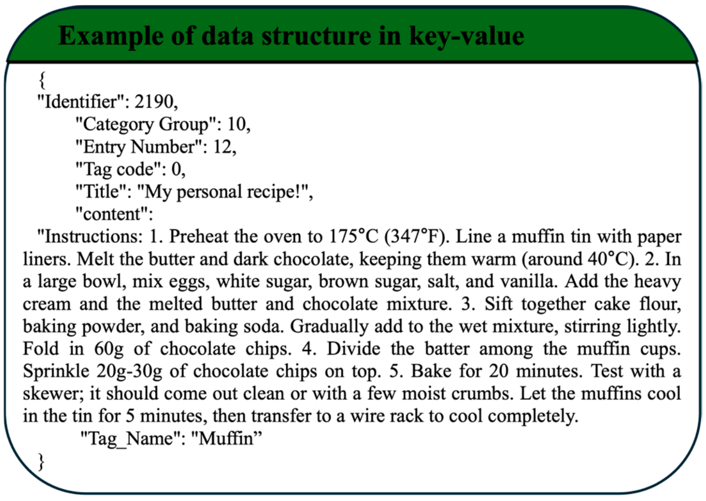
Finally, the QA generator utilizes an LLM, such as GPT-3.5-Turbo, to generate accurate and useful answers to questions. This process is performed based on the context selected from the vector database, and the final answer is returned to the user.

3.2. Data Extraction: LangChain Integration

An easy method for implementing RAG is to employ LangChain (0.2.8) [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 (0.2.8). 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: The extracted 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 (0.2.8) manages the flow used in this process.

LangChain (0.2.8) is a software development kit 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 "TB Search" 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.



Example of data structure in key-value

```
{
"Identifier": 2190,
      "Category Group": 10,
      "Entry Number": 12,
      "Tag code": 0,
      "Title": "My personal recipe!",
      "content":
"Instructions: 1. Preheat the oven to 175°C (347°F). Line a muffin tin with paper liners. Melt the butter and dark chocolate, keeping them warm (around 40°C). 2. In a large bowl, mix eggs, white sugar, brown sugar, salt, and vanilla. Add the heavy cream and the melted butter and chocolate mixture. 3. Sift together cake flour, baking powder, and baking soda. Gradually add to the wet mixture, stirring lightly. Fold in 60g of chocolate chips. 4. Divide the batter among the muffin cups. Sprinkle 20g-30g of chocolate chips on top. 5. Bake for 20 minutes. Test with a skewer; it should come out clean or with a few moist crumbs. Let the muffins cool in the tin for 5 minutes, then transfer to a wire rack to cool completely.
       "Tag_Name": "Muffin"
}
```

Figure 3. User data extracted in the JavaScript Object Notation JSON format in the case of UserID as 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 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.

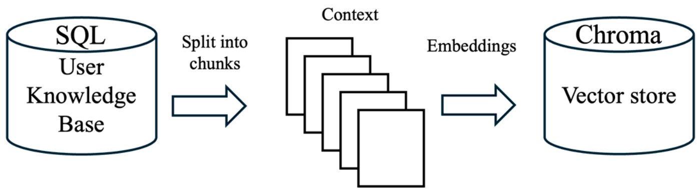


Figure 4. Document embedding workflow.

3.3. *Role 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.

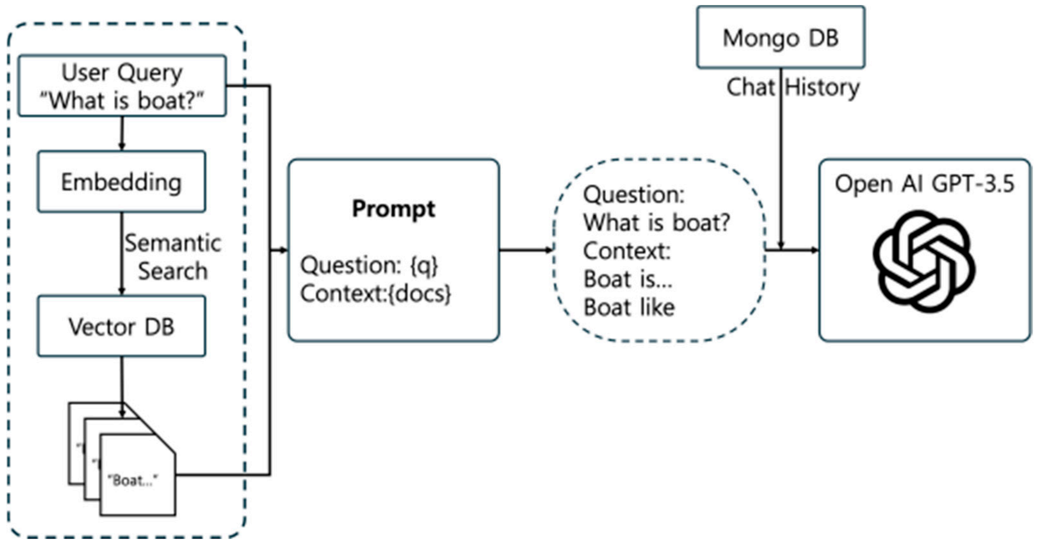


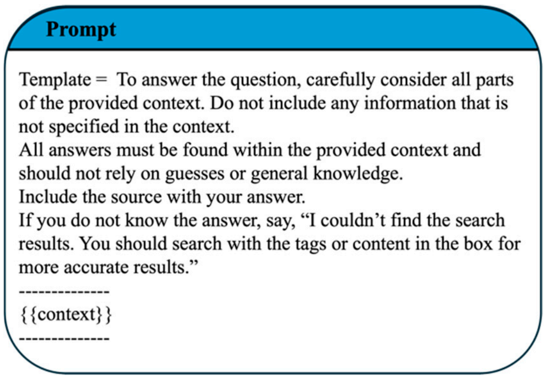
Figure 5. Process of handling user prompts and generating answers using LangChain (0.2.8) with MongoDB.

Prompts act as an interface between user questions and LLMs and are used as the basis for generating answers. The proposed system obtains a list of documents extracted via LangChain (0.2.8), 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 cannot 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.



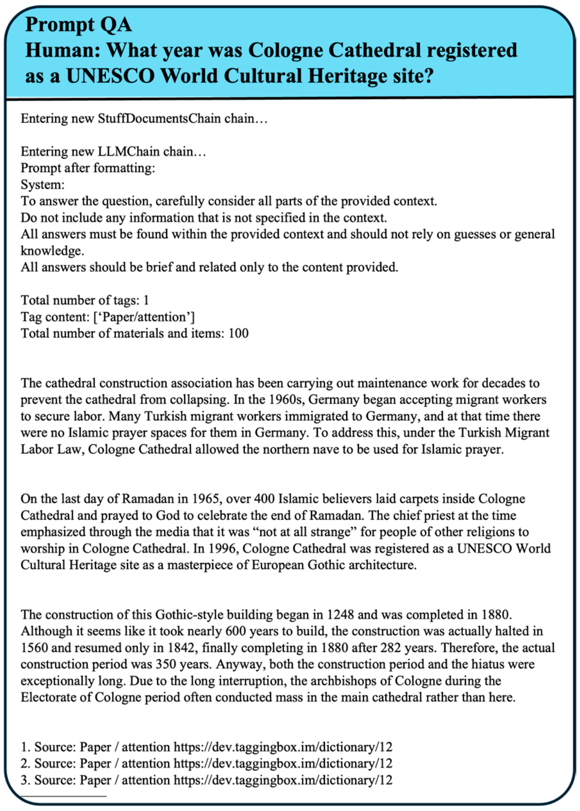
Prompt

Template = To answer the question, carefully consider all parts of the provided context. Do not include any information that is not specified in the context.
All answers must be found within the provided context and should not rely on guesses or general knowledge.
Include the source with your answer.
If you do not know the answer, say, "I couldn't find the search results. You should search with the tags or content in the box for more accurate results."

{{context}}

Figure 6. Custom prompt template.

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



Prompt QA
Human: What year was Cologne Cathedral registered as a UNESCO World Cultural Heritage site?

```
Entering new StuffDocumentsChain chain…

Entering new LLMChain chain…
Prompt after formatting:
System:
To answer the question, carefully consider all parts of the provided context.
Do not include any information that is not specified in the context.
All answers must be found within the provided context and should not rely on guesses or general
knowledge.
All answers should be brief and related only to the content provided.

Total number of tags: 1
Tag content: ['Paper/attention']
Total number of materials and items: 100


The cathedral construction association has been carrying out maintenance work for decades to
prevent the cathedral from collapsing. In the 1960s, Germany began accepting migrant workers
to secure labor. Many Turkish migrant workers immigrated to Germany, and at that time there
were no Islamic prayer spaces for them in Germany. To address this, under the Turkish Migrant
Labor Law, Cologne Cathedral allowed the northern nave to be used for Islamic prayer.


On the last day of Ramadan in 1965, over 400 Islamic believers laid carpets inside Cologne
Cathedral and prayed to God to celebrate the end of Ramadan. The chief priest at the time
emphasized through the media that it was "not at all strange" for people of other religions to
worship in Cologne Cathedral. In 1996, Cologne Cathedral was registered as a UNESCO World
Cultural Heritage site as a masterpiece of European Gothic architecture.


The construction of this Gothic-style building began in 1248 and was completed in 1880.
Although it seems like it took nearly 600 years to build, the construction was actually halted in
1560 and resumed only in 1842, finally completing in 1880 after 282 years. Therefore, the actual
construction period was 350 years. Anyway, both the construction period and the hiatus were
exceptionally long. Due to the long interruption, the archbishops of Cologne during the
Electorate of Cologne period often conducted mass in the main cathedral rather than here.


1. Source: Paper / attention https://dev.taggingbox.im/dictionary/12
2. Source: Paper / attention https://dev.taggingbox.im/dictionary/12
3. Source: Paper / attention https://dev.taggingbox.im/dictionary/12
```

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

3.4. History Management: MongoDB

The QA system developed in this study was implemented to manage 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 conversations from 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.

3.5. Personalized RAG-Based Responses

In this design, we developed an RAG-based question-answering (QA) system that generates accurate answers tailored to specific contexts by utilizing stored documents and tagged keywords extracted from an SQL database. When the RAG pipeline is applied to the QA system, users can receive answers based on the information they have previously built. In contrast, general ChatGPT services provide responses based on commonly learned information. This demonstrates the ability of the developed interactive QA system to reflect personalized information through RAG. An example highlighting the differences in responses before and after RAG was applied can be found in Figure A1 of Appendix A.

3.6. Evaluation: 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 8 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.

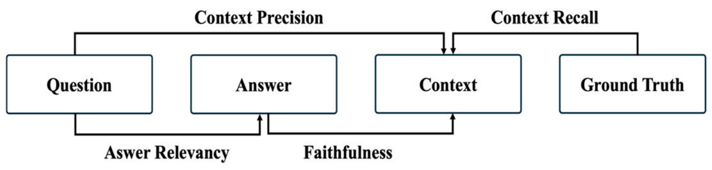


Figure 8. RAGAs evaluation.

The mathematical formulas for the evaluation metrics are summarized in Table 1. This table presents the formulas used to assess answer generation and document retrieval in the RAGAs framework.

Table 1. RAGAs evaluation metrics.

Metric	Equation
Faithfulness	$\text{Faithfulness score} = \frac{\lvert\text{Number of claims in the answer that can be inferred from the given context}\rvert}{\lvert\text{Total number of claims in the answer}\rvert}$
Answer Relevancy	$\text{answer relevancy} = \frac{1}{N}\sum_{i=1}^{N} cos\left(E_{g_i}, E_o\right)$ $\text{answer relevancy} = \frac{1}{N}\sum_{i=1}^{N} \frac{E_{g_i} \cdot E_o}{\lVert E_{g_i}\rVert \lVert E_o\rVert}$
Context Precision	$\text{Context Precision@K} = \frac{\sum_{k=1}^{K}(\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top } K \text{ results}}$ $\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k+false positives@k})}$
Context Recall	$\text{context recall} = \frac{\lvert\text{Ground Truth sentences that can be attributed to context}\rvert}{\lvert\text{Number of sentences in Ground Truth}\rvert}$

E_{g_i} is the embedding of generated question i, N is the total number of generated questions; E_o 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.

3.7. Experimental Setup

The data used in the experiment were evaluated using the commonsense dataset provided by AIHub [32] as the personalized database of "Tagging 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. To facilitate understanding of the role of each column, a specific example is provided in Appendix B, Table A1.

Based on this, we evaluated the search performance of the RAGAs framework. We used the NVIDIA RTX 3070 GPU (Santa Clara, CA, USA) and the Anaconda environment for the experimental setup and the Ollama (0.2.5) library [33] for experimentation. The versions of RAGA, Chroma DB [34], the vector database used in this study, and LangChain (0.2.8), which supports the RAG evaluation process, are detailed in Table 2.

Table 2. System environment employed for RAG development and evaluation.

Components	Version
CPU	AMD Ryzen 9 5900X 12-Core Processor (Advanced Micro Devices, Inc., Santa Clara, CA, USA)
RAM	32 GB
GPU	NVIDIA GeForce RTX 3070
Anaconda Python	3.9.19
Ollama	0.2.5
RAGA	0.1.9
Chroma DB	0.4.23
LangChain	0.2.8

3.8. Performance Analysis for Various Model Combinations

To implement the RAG, we used GPT-3.5-Turbo as the LLM 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 the 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.

4. Evaluation Results

We evaluated the performance of the QA system in our study using the two public data sources mentioned above. We used the RAGAs framework to analyze the results of the system's evaluation of contextual relevance, accuracy, and reliability of responses.

First, we present the evaluation results using the common knowledge dataset. This dataset consists of question–answer pairs on WIKI texts, where the question is related to the content of the WIKI text and the answer is the corresponding answer pair in the WIKI text. The information from this dataset was processed by the user's "TaggingBox" and stored in a personalized database, which we used to evaluate how well our QA system generates accurate and relevant answers to user-input questions.

Figure 9 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 show high accuracies of 0.46 and 0.45, respectively, suggesting that Llama-3-8B offers higher accuracy than the other models.

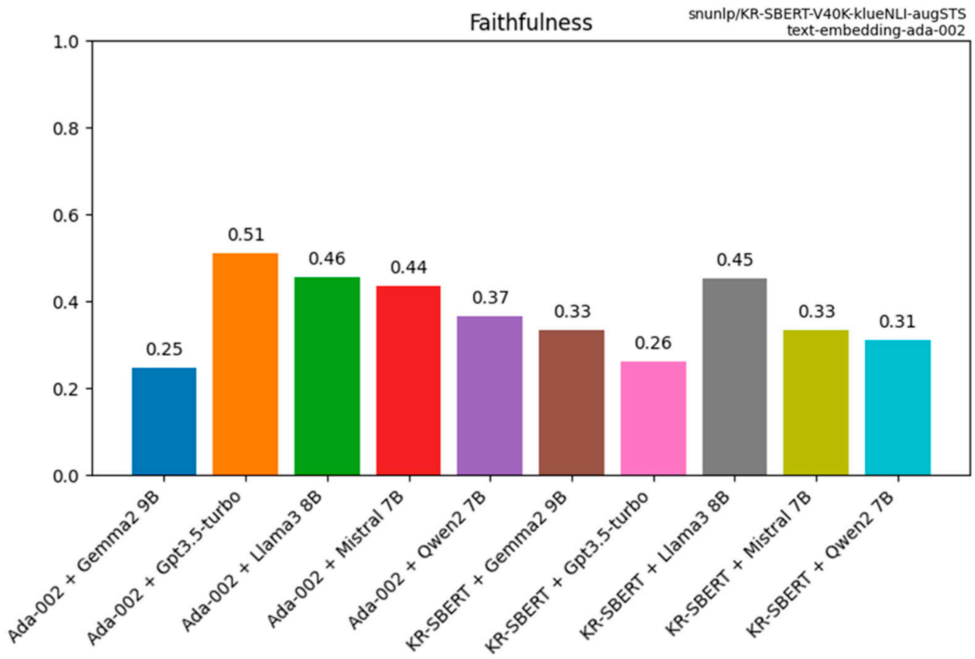


Figure 9. Faithfulness assessment results for the commonsense dataset.

Figure 10 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.

Figure 11 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 effectiveness in recalling information for a given context. By contrast, the Ada-002 + GPT-3.5-Turbo combination shows a relatively low recall score of 0.67, suggesting that this combination has weak information-recall capability.

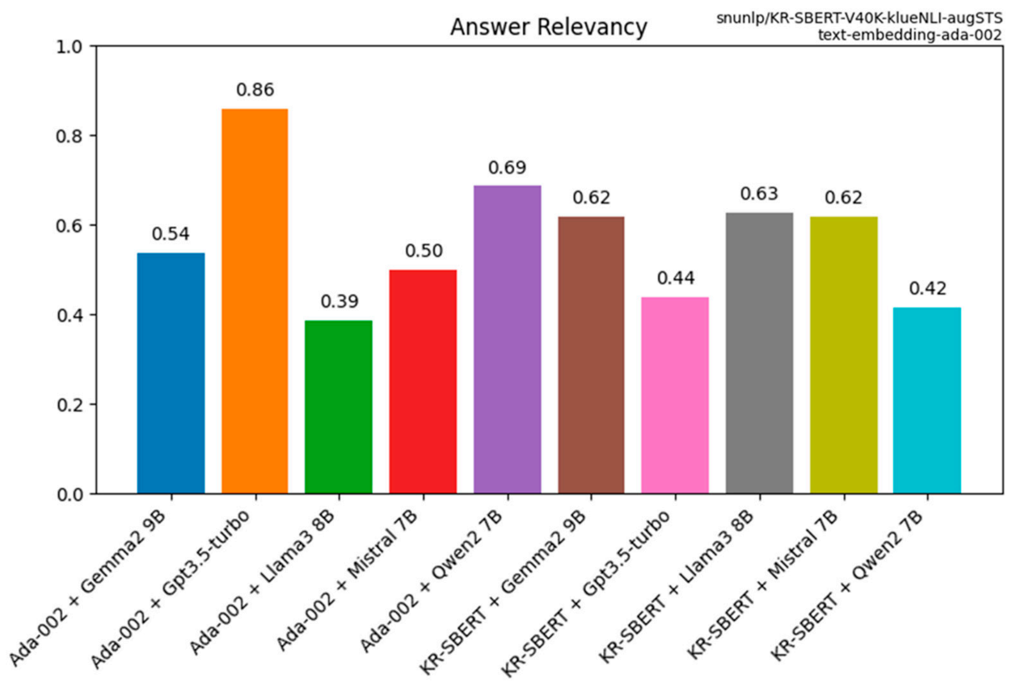


Figure 10. Answer relevance results for commonsense dataset.

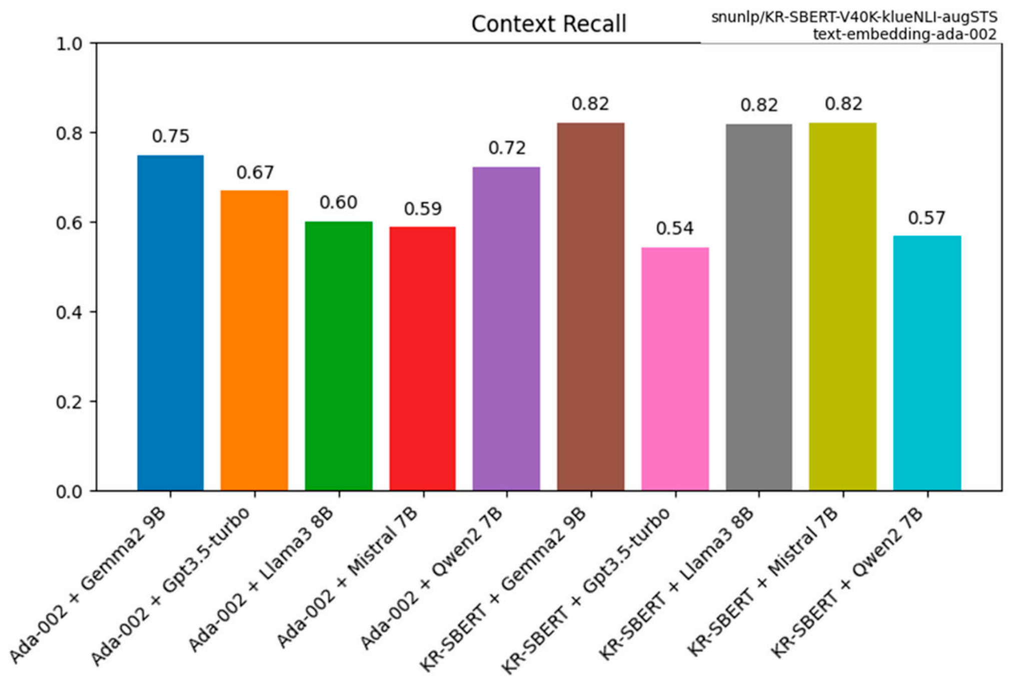


Figure 11. Context-recall results for the commonsense dataset.

Figure 12 shows the context-precision scores for all models, wherein the Ada-002 + GPT-3.5-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.

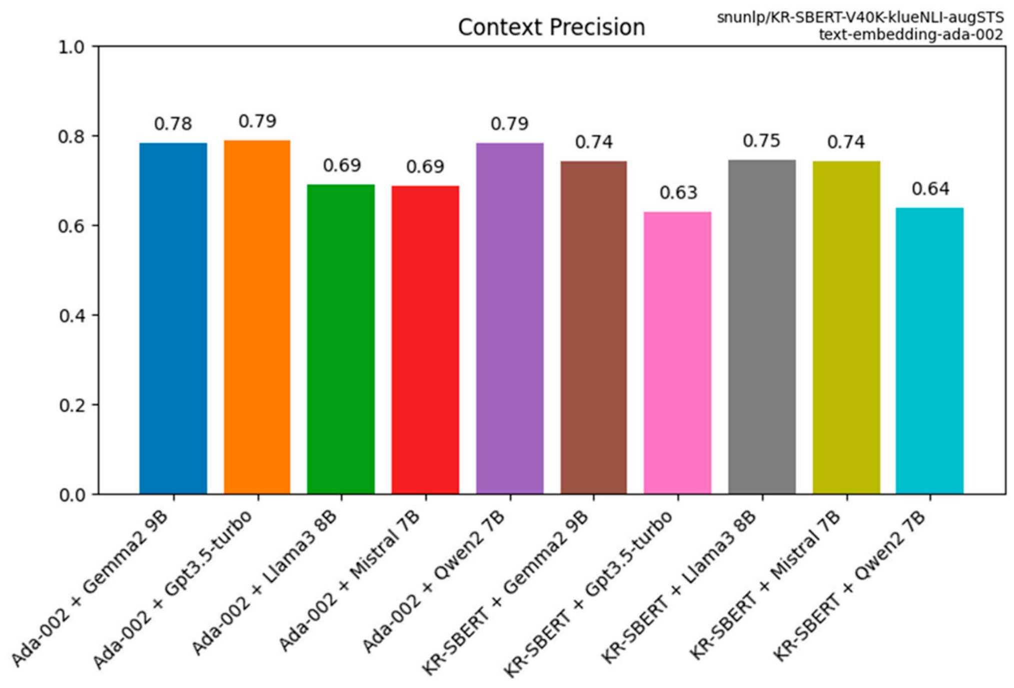


Figure 12. Context-precision results for the commonsense dataset.

Additionally, the response times of all combinations were analyzed. Figure 13 shows a comparison of the processing time per data item for each model combination. Among the KR-SBERT combinations, KR-SBERT + GPT-3.5-Turbo exhibits 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 exhibits 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 + GPT-3.5-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 requires API communication.

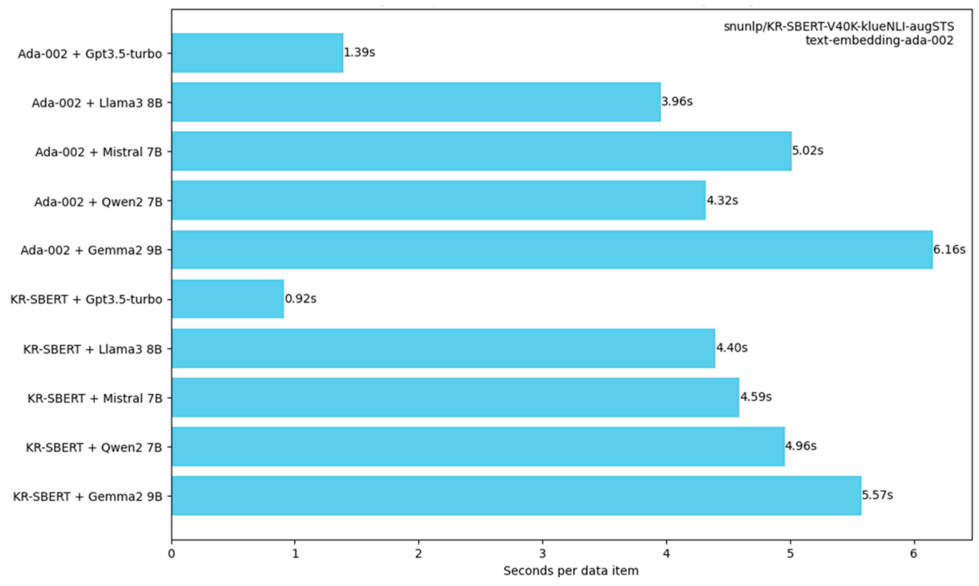


Figure 13. Processing time per data item for the various LLM and embedding combinations for the commonsense dataset.

Overall, Ada-002 + GPT-3.5-Turbo obtains the best performance, outperforming others across several performance metrics, including accuracy, answer relevance, and contextual precision, and exhibits the second lowest processing time. Moreover, the KR-SBERT + GPT-3.5-Turbo combination exhibits the lowest processing time, suggesting that it is a useful alternative.

Next, we conducted additional experiments and evaluated the performance using a news article dataset [40], alongside the existing open datasets. This news article dataset, consisting of 450,000 news articles, serves as a training set for developing machine reading comprehension systems. It includes articles across nine different categories and provides ground truth for questions, along with the context from which the answers were derived, enabling evaluation through RAGAs.

As shown in Figure 14, the Ada-002 + GPT-3.5-Turbo combination achieves the highest faithfulness score of 0.47. This indicates that this combination generates responses that are more faithful to the original information compared to other models. By contrast, the KR-SBERT + GPT-3.5-Turbo combination exhibits the lowest faithfulness score of 0.28.

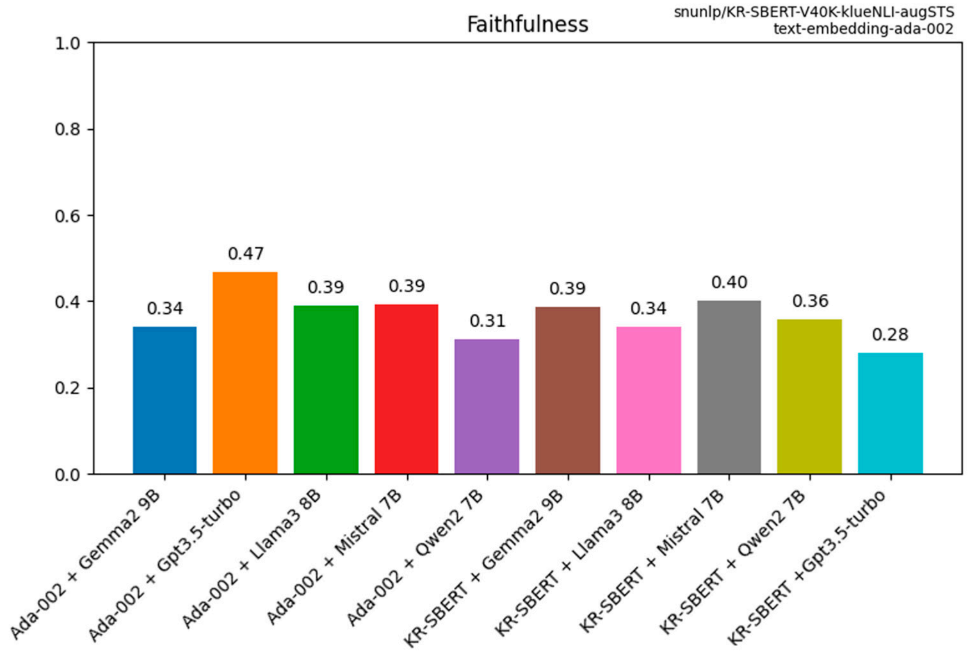


Figure 14. Faithfulness assessment results for the news article dataset.

In Figure 15, the Ada-002 + GPT-3.5-Turbo combination demonstrates the highest performance in answer relevance with a score of 0.82. This indicates a very strong alignment with the original question, suggesting that this combination can provide clear and accurate responses to user inquiries. Conversely, the KR-SBERT + Mistral 7B combination records the lowest score on this metric.

In Figure 16, contextual recall performance is evaluated, and the Ada-002 + GPT-3.5-Turbo combination shows lower performance with a score of 0.51. By contrast, the KR-SBERT + GPT-3.5-Turbo combination achieves the highest score of 0.68.

In Figure 17, contextual precision is evaluated, and the Ada-002 + GPT-3.5-Turbo combination scores 0.73. Although this combination demonstrates a sufficiently reliable level of precision in document retrieval, the KR-SBERT + Mistral2-7B combination achieves the highest precision with a score of 0.87.

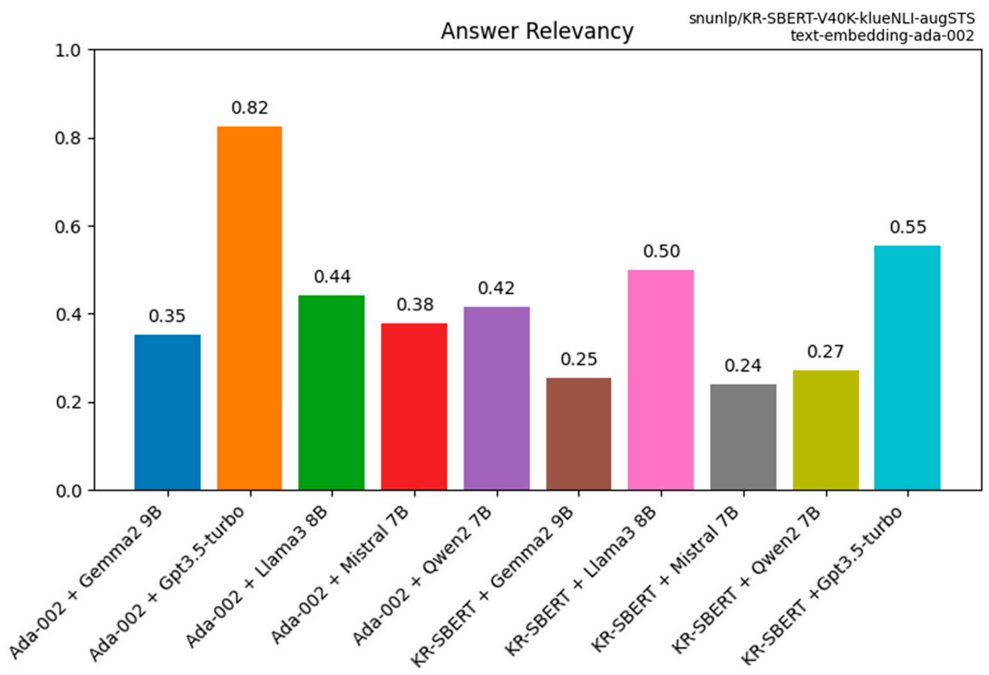


Figure 15. Answer relevance results for the news article dataset.

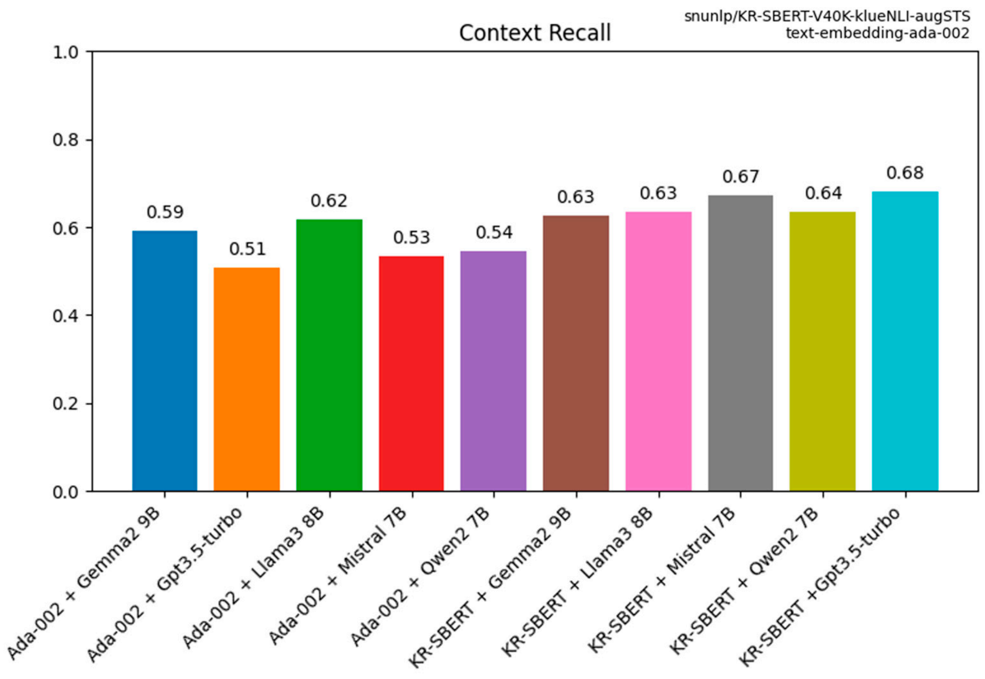


Figure 16. Context-recall results for the news article dataset.

Figure 18 visualizes the data-processing time for each embedding combination. The KR-SBERT + GPT-3.5-Turbo combination records the fastest processing time of 0.95 s, which is likely the result of running directly on the local machine. By contrast, the Ada-002 + GPT-3.5-Turbo combination took slightly longer at 1.56 s but maintained a very efficient processing speed while using API communication. This makes Ada-002 + GPT-3.5-Turbo a balanced choice for performance and speed. Although the Ada-002 + GPT-3.5-Turbo combination shows balanced results in terms of performance and processing speed, we performed additional experiments to examine the performance of the newer GPT-4 model. GPT-4 has several reported improvements, including better context handling, increased accuracy, and reduced hallucination rates.

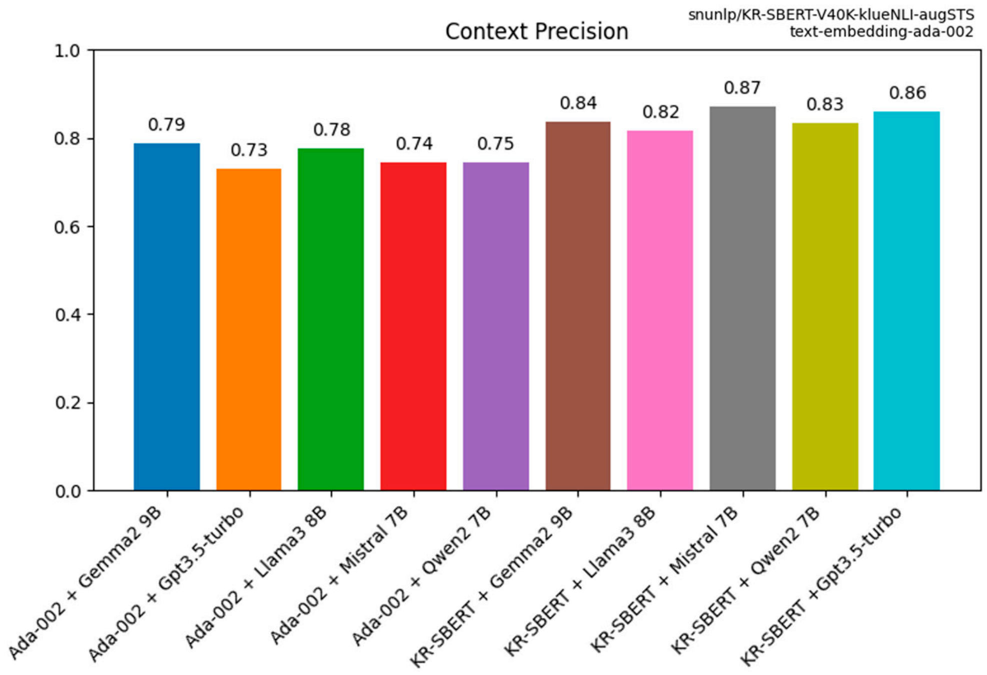


Figure 17. Context-precision results for the news article dataset.

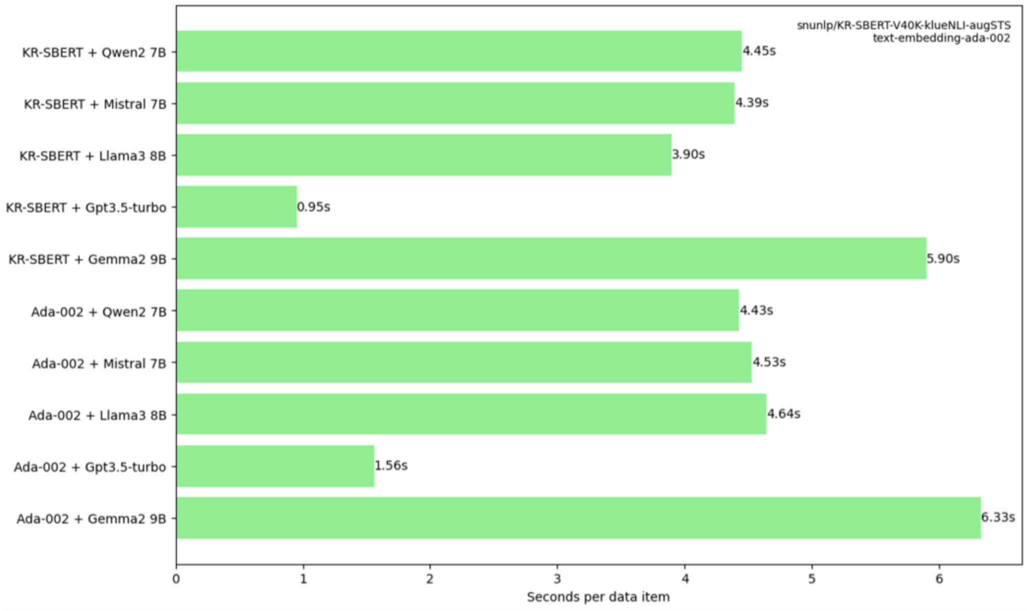


Figure 18. Processing time per data item for the various LLMs and embedding combinations for the news article dataset.

To evaluate the impact of these improvements on real-world performance, we conducted an experiment with GPT-4. The experiment aimed to compare the performance of GPT-4 and GPT-3.5-Turbo and to assess how the two models perform in a real-world production environment. According to the results shown in Figure 19, GPT-4 incurs an approximately 8137.5% higher cost than GPT-3.5-Turbo when processing 100 pieces of data. In addition, in terms of processing time, GPT-4 is about 111.5% slower than GPT-3.5-Turbo, where processing time refers to the time taken to process one piece of data. These results highlight that cost and time efficiency can be a big issue in real-world production environments where cost and time efficiency are critical.

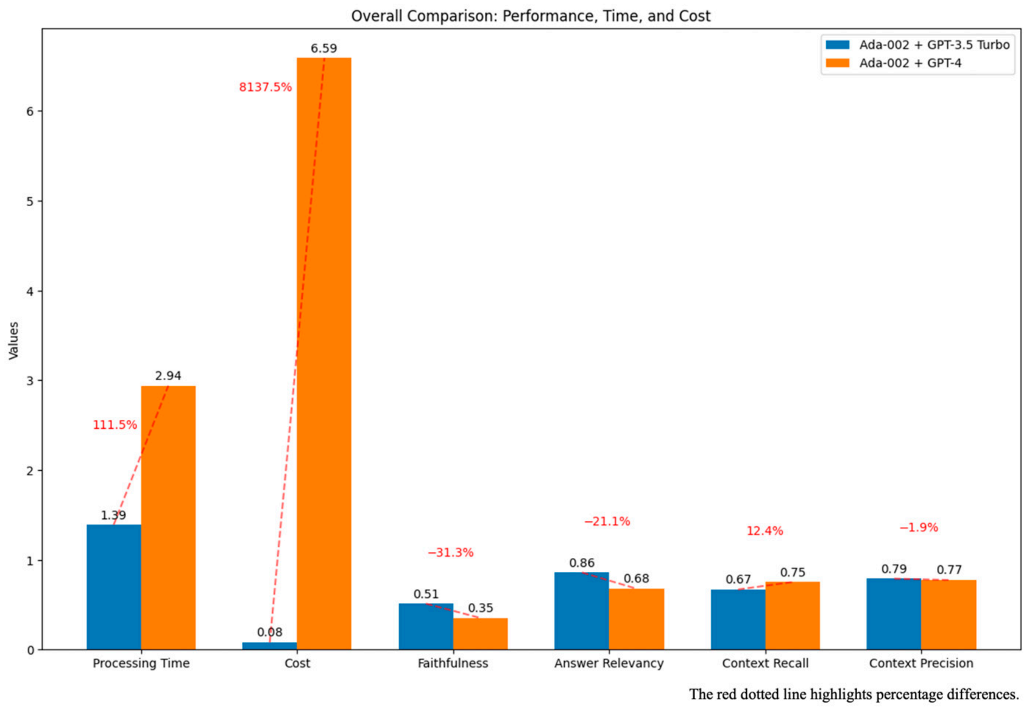


Figure 19. Comparative evaluation of processing time, cost, and performance metrics for Ada-002 + GPT-3.5-Turbo and Ada-002 + GPT-4.

In the RAGAs evaluation, GPT-3.5-Turbo performed 31.3%, 21.1%, and 1.9% better in terms of faithfulness, answer relevancy, and context precision, respectively. These results indicate that GPT-3.5-Turbo is better suited to meet research requirements where the consistency and accuracy of responses are important. Conversely, GPT-4 performed 12.4% better on Context Recall, but this advantage did not lead to a significant improvement in overall system performance. Thus, while GPT-4 offers improved performance in context handling and accuracy, cost-effectiveness and processing speed are important factors in real-world applications. It can be concluded that Ada-002 + GPT-3.5-Turbo may be more suitable in situations where cost and processing time efficiency are important.

5. Discussion

5.1. Results and Contribution

This study demonstrates that a QA system leveraging RAG can be customized for specific domains by indexing text corpora.

Additionally, because the system ensures consistent tracking and updating of information, including documents modified or deleted by individuals logged into the personalized database, there is no need to retrain or fine-tune it on sensitive personal and workplace information. To address the limitations of existing LLM-based QA systems, which often fail to record the context of conversations, we implemented a stable RAG system using MongoDB to provide contextual information to prompts, thereby offering a search experience that reflects individual search history.

Finally, experimental results using GPT-3.5-Turbo with custom templates for prompting showed that this combination outperforms traditional RAG setups. These findings indicate that an optimized RAG design plays a crucial role in delivering accurate answers to user queries and confirm that the proposed system can achieve high efficiency and effectiveness in real-world applications.

5.2. Limitations and Future Work

The RAG system developed in this study has limitations due to its reliance on the performance of the LLM. Additionally, the QA system extracts context based on the similarity of the question to generate responses. In cases where the context is not established in the database, the response is limited. Although this reduces hallucinations, it may be perceived as a limitation of the system from the user's perspective. This could potentially limit the generalization of performance across databases of varying sizes and structures. Therefore, this study aims to improve the QA system's performance by adjusting various thresholds.

Moreover, the data used in the experiment were selected specifically for research purposes from a customized database optimized for a particular domain. This implies that the same level of performance may not be guaranteed with datasets from other domains or with different structures, which could introduce potential bias in the study. Nonetheless, the domain used in our actual experiment is already a structured database from a company currently in operation. Consequently, multiple users in this domain can expand their personalized databases with their own information and use the RAG QA system to derive answers without hallucinations, which is a significant contribution of this study. In the future, we plan to leverage the accumulated information from this company's system to increase the potential for generalization.

In the future, we intend to focus our research on overcoming the limitations identified in this study by utilizing the accumulated information from the company's system. Specifically, we will work on enhancing the integration of data from various domains and improving the ability to process unstructured data, enabling the system to handle more complex and diverse queries.

Author Contributions: Conceptualization, J.B., B.K., K.-A.C. and E.L.; methodology, J.B., B.K. and K.-A.C.; software, J.B., B.K. and E.L.; validation, J.B. and B.K.; investigation, J.B., B.K., K.-A.C. and E.L.; data curation, J.B. and B.K.; writing—original draft preparation, J.B., B.K. and K.-A.C.; writing—review and editing, J.B., B.K. and K.-A.C.; visualization, J.B. and B.K.; supervision, K.-A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

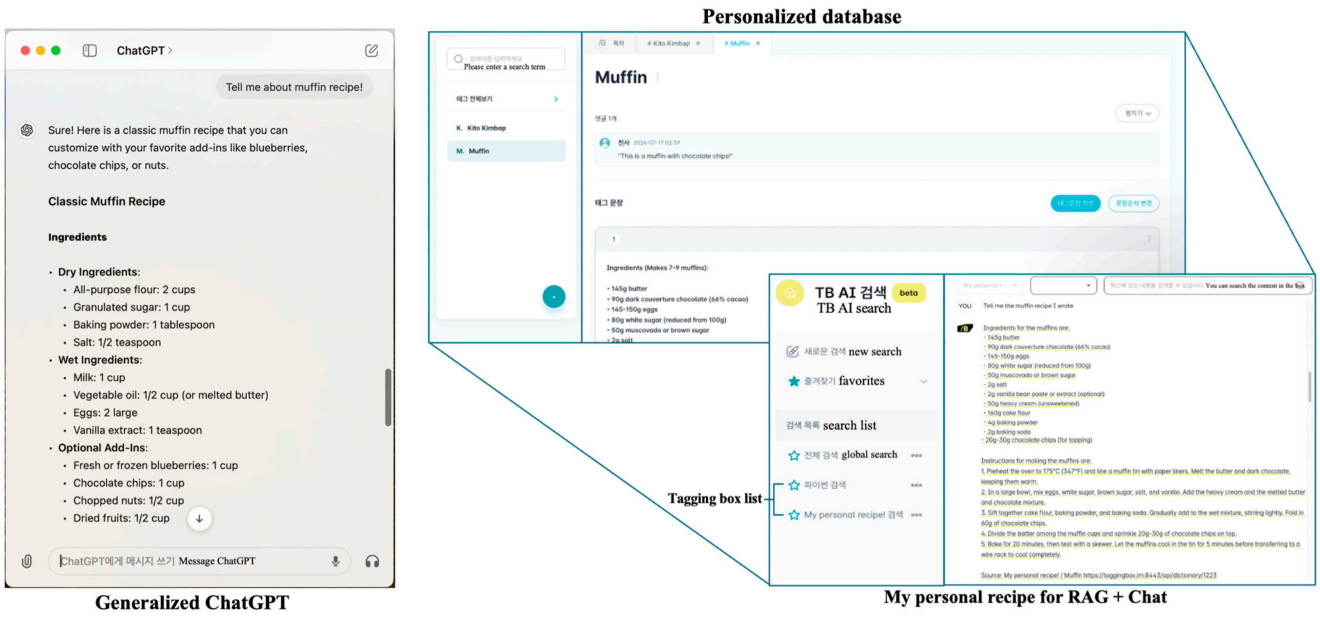
Informed Consent Statement: Not applicable.

Data Availability Statement: The data are contained within the article.

Conflicts of Interest: Author Eunhyung Lee was employed by the company Textway Inc. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A

Figure A1 compares the response differences between our RAG-based QA system and traditional LLMs, particularly GPTs. This comparison demonstrates the ability of RAG to reflect personalized information about user prompts in the interactive question-answering system we developed. We highlight the difference in responses before and after the application of RAG with an example of a company's application solution where the RAG pipeline was applied to a QA system. As shown in Figure A1, the user prompt for a muffin recipe and the output from ChatGPT-3.5 do not reflect the user's inclinations and intentions: the output does not reflect context because it is based on already-learned parameters, and the LLM simply analyzes the prompt and generates an answer. By contrast, as shown on the right in Figure A1, the contextualized response generated by RAG from the personalized database of "TaggingBox" implemented in this system outputs a muffin recipe created and updated by the individual, displaying contextual information relevant only to the individual.



Tell me about muffin recipe!

Sure! Here is a classic muffin recipe that you can customize with your favorite add-ins like blueberries, chocolate chips, or nuts.

Classic Muffin Recipe

Ingredients

- **Dry Ingredients**:
  - All-purpose flour: 2 cups
  - Granulated sugar: 1 cup
  - Baking powder: 1 tablespoon
  - Salt: 1/2 teaspoon
- **Wet Ingredients**:
  - Milk: 1 cup
  - Vegetable oil: 1/2 cup (or melted butter)
  - Eggs: 2 large
  - Vanilla extract: 1 teaspoon
- **Optional Add-Ins**:
  - Fresh or frozen blueberries: 1 cup
  - Chocolate chips: 1 cup
  - Chopped nuts: 1/2 cup
  - Dried fruits: 1/2 cup

천사 2024-07-17 02:39
"This is a muffin with chocolate chips!"

1

Ingredients (Makes 7-9 muffins):

- 145g butter
- 90g dark couverture chocolate (66% cacao)
- 145-150g eggs
- 80g white sugar (reduced from 100g)
- 50g muscovado or brown sugar
- 2g salt

YOU Tell me the muffin recipe I wrote

Ingredients for the muffins are:
- 145g butter
- 90g dark couverture chocolate (66% cacao)
- 145-150g eggs
- 80g white sugar (reduced from 100g)
- 50g muscovado or brown sugar
- 2g salt
- 2g vanilla bean paste or extract (optional)
- 50g heavy cream (unsweetened)
- 160g cake flour
- 4g baking powder
- 2g baking soda
- 20g-30g chocolate chips (for topping)

Instructions for making the muffins are:
1. Preheat the oven to 175°C (347°F) and line a muffin tin with paper liners. Melt the butter and dark chocolate, keeping them warm.
2. In a large bowl, mix eggs, white sugar, brown sugar, salt, and vanilla. Add the heavy cream and the melted butter and chocolate mixture.
3. Sift together cake flour, baking powder, and baking soda. Gradually add to the wet mixture, stirring lightly. Fold in 60g of chocolate chips.
4. Divide the batter among the muffin cups and sprinkle 20g-30g of chocolate chips on top.
5. Bake for 20 minutes, then test with a skewer. Let the muffins cool in the tin for 5 minutes before transferring to a wire rack to cool completely.

Source: My personal recipe! / Muffin https://taggingbox.im:8443/api/dictionary/1223

Figure A1. A comparison example between ChatGPT-3.5 and the RAG-based QA system's question and answer responses.

Appendix B

The contexts listed in Table A1 are examples of the dataset format used in a personalized document describing the ancient walls surrounding Jinju, a region in South Korea, taken from a public dataset. This document is stored in a personalized database.

Table A1. An example of dataset format in the evaluation experiment.

Question	Ground Truth	Context
What is the area of Ibanseong-myeon?	19.41 km^2	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 km^2, 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 1 January 2012, the population was 3233 (male: 1556, female: 1677) with 1413 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, 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.

References

1. Radford, A.; Narasimhan, K. Improving Language Understanding by Generative Pre-Training, OpenAI Blog 2018. Available online: https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf (accessed on 25 July 2024).
2. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models Are Unsupervised Multitask Learners, OpenAI Blog 2019. Available online: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 25 July 2024).
3. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Agarwal, S. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
4. Liu, J.; Shen, D.; Zhang, Y.; Dolan, B.; Carin, L.; Chen, W. What Makes Good In-Context Examples for GPT-3? *arXiv* **2021**, arXiv:2101.06804.

5. Bin-Nashwan, S.A.; Sadallah, M.; Bouteraa, M. Use of ChatGPT in academia: Academic integrity hangs in the balance. *Technol. Soc.* **2023**, *75*, 102370. [CrossRef]
6. Gao, C.A.; Howard, F.M.; Markov, N.S.; Dyer, E.C.; Ramesh, S.; Luo, Y.; Pearson, A.T. Comparing Scientific Abstracts Generated by ChatGPT to Real Abstracts with Detectors and Blinded Human Reviewers. *NPJ Digit. Med.* **2023**, *6*, 75. [CrossRef] [PubMed]
7. Ferreiro-Santamaria, G. Exploring the Role of ChatGPT in English Teaching within Higher Education Settings. *Int. J. Trends Dev. Educ.* **2024**, *4*, 44–58.
8. Surameery, N.M.S.; Shakor, M.Y. Use Chat GPT to Solve Programming Bugs. *Int. J. Inf. Technol. Comput. Eng. IJITC* **2023**, *3*, 17–22. [CrossRef]
9. Floridi, L.; Chiriatti, M. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds Mach.* **2020**, *30*, 681–694. [CrossRef]
10. Xu, Z.; Jain, S.; Kankanhalli, M. Hallucination is inevitable: An innate limitation of large language models. *arXiv* **2024**, arXiv:2401.11817.
11. Alkaissi, H.; McFarlane, S.I. Artificial hallucinations in ChatGPT: Implications in scientific writing. *Cureus* **2023**, *15*, e35179. [CrossRef]
12. Wang, B.; Chen, W.; Pei, H.; Xie, C.; Kang, M.; Zhang, C.; Xu, C.; Xiong, Z.; Dutta, R.; Schaeffer, R.; et al. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 31232–31339.
13. Lewis, P.S.H.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
14. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Guo, Q.; Wang, M.; et al. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv* **2023**, arXiv:2312.10997.
15. Chen, J.; Lin, H.; Han, X.; Sun, L. Benchmarking Large language models in retrieval-augmented generation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, Canada, 20–27 February 2024; Volume 38, pp. 17754–17762.
16. Zhao, P.; Zhang, H.; Yu, Q.; Wang, Z.; Geng, Y.; Fu, F.; Yang, L.; Zhang, W.; Jiang, J.; Cui, B. Retrieval-Augmented Generation for AI-Generated Content: A Survey. *arXiv* **2024**, arXiv:2402.19473.
17. Xia, Y.; Huang, Y.; Qiu, Q.; Zhang, X.; Miao, L.; Chen, Y. A Question and Answering Service of Typhoon Disasters Based on the T5 Large Language Model. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 165. [CrossRef]
18. Li, H.; Su, Y.; Cai, D.; Wang, Y.; Liu, L. A survey on retrieval-augmented text generation. *arXiv* **2022**, arXiv:2202.01110.
19. Es, S.; James, J.; Espinosa-Anke, L.; Schockaert, S. RAGAS: Automated Evaluation of Retrieval Augmented Generation. *arXiv* **2023**, arXiv:2309.15217.
20. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* **2023**, *55*, 248. [CrossRef]
21. Zhang, Y.; Li, Y.; Cui, L.; Cai, D.; Liu, L.; Fu, T.; Huang, X.; Zhao, E.; Zhang, Y.; Chen, Y.; et al. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *arXiv* **2023**, arXiv:2309.01219.
22. Liu, F.; Liu, Y.; Shi, L.; Huang, H.; Wang, R.; Yang, Z.; Zhang, L.; Li, Z.; Ma, Y. Exploring and Evaluating Hallucinations in LLM-Powered Code Generation. *arXiv* **2024**, arXiv:2404.00971.
23. Galitsky, B.; Chernyavskiy, A.; Ilvovsky, D. Truth-o-meter: Handling multiple inconsistent sources repairing LLM hallucinations. In Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, Washington, DC, USA, 14–18 July 2024; pp. 2817–2821.
24. Hanna, E.; Levic, A. Comparative Analysis of Language Models: Hallucinations in ChatGPT: Prompt Study, DIVA 2023. Available online: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1764165&dswid=6109 (accessed on 25 July 2023).
25. Chen, Y.; Fu, Q.; Yuan, Y.; Wen, Z.; Fan, G.; Liu, D.; Zhang, D.; Li, Z.; Xiao, Y. Hallucination detection: Robustly discerning reliable answers in large language models. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, ACM, Birmingham, UK, 21–25 October 2023; pp. 245–255.
26. Yao, J.Y.; Ning, K.P.; Liu, Z.H.; Ning, M.N.; Yuan, L. LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples. *arXiv* **2023**, arXiv:2310.01469.
27. Zhang, M.; Press, O.; Merrill, W.; Liu, A.; Smith, N.A. How language model hallucinations can snowball. *arXiv* **2023**, arXiv:2305.13534.
28. Varshney, N.; Raj, S.; Mishra, V.; Chatterjee, A.; Sarkar, R.; Saeidi, A.; Baral, C. Investigating and Addressing Hallucinations of LLMs in Tasks Involving Negation. *arXiv* **2024**, arXiv:2406.05494.
29. TaggingBox. Available online: https://taggingbox.im/ (accessed on 24 July 2024).
30. MongoDB: The Developer Data Platform. Available online: https://www.mongodb.com/ (accessed on 25 July 2024).
31. LangChain. Available online: https://github.com/langchain-ai/langchain (accessed on 25 July 2024).
32. AI Hub. General Knowledge Dataset. Available online: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=106 (accessed on 25 July 2024).
33. Ollama. Available online: https://github.com/ollama/ollama (accessed on 25 July 2024).
34. Chroma. Available online: https://www.trychroma.com/ (accessed on 25 July 2024).
35. Gemma Team. Gemma2: Improving Open Language Models at a Practical Size. *arXiv* **2024**, arXiv:2408.00118.
36. Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. The Llama 3 Herd of Models. *arXiv* **2024**, arXiv:2407.21783.

37. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D.S.; Casas, D.D.L.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. Mistral 7B. *arXiv* **2023**, arXiv:2310.06825.
38. Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; et al. Qwen2 Technical Report. *arXiv* **2024**, arXiv:2407.10671.
39. snunlp. KR-SBERT-V40K-klueNLI-augSTS. Available online: https://huggingface.co/snunlp/KR-SBERT-V40K-klueNLI-augSTS (accessed on 24 July 2024).
40. AI Hub. Machine Reading Dataset. Available online: https://www.aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=89 (accessed on 19 August 2024).