2024 18th International Conference on Advanced Computing and Analytics (ACOMPA) | 979-8-3315-4246-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ACOMPA64883.2024.00021

The enhanced context for AI-generated learning advisors with Advanced RAG

1st Anh Nguyen Thi Dieu
Industrial University of Ho Chi Minh City
Van Hien University
Ho Chi Minh City, Viet Nam
anhntd2221@pgr.iuh.edu.vn
anhntd@vhu.edu.vn

2nd Hien T. Nguyen
Ho Chi Minh University of Banking
Ho Chi Minh City, Viet Nam
nthien@hub.edu.vn

3rd Chien Ta Duy Cong
Industrial University of Ho Chi Minh City
Ho Chi Minh City, Viet Nam
taduycongchien@iuh.edu.vn

***Abstract*—In Van Hien University, which serves tens of thousands of students, efficiently addressing student inquiries related to training programs, tuition fees, graduation conditions, and output standards has become a critical and urgent challenge. Traditional advisory methods are increasingly strained by the volume of queries and the complexity of providing accurate, context-specific guidance. This research introduces an AI Agent leveraging the Advanced Retrieval-Augmented Generation (RAG) model to solve this problem. The AI Agent is designed to dynamically retrieve relevant information from extensive institutional documentation and provide precise, context-aware advice to students. By integrating cutting-edge language models with a robust retrieval mechanism, the proposed system aims to enhance the accuracy, relevance, and scalability of student advisory services. This approach not only improves the efficiency of responding to student needs but also ensures that the guidance provided aligns closely with the university's standards and policies, ultimately contributing to a more streamlined and effective student support system.**

***Index Terms*—RAG, advanced RAG, large language models, AI agents.**

I. INTRODUCTION

The rapid advancement of artificial intelligence (AI) has been significantly propelled by the development of large language models (LLMs) [2]. These models, such as GPT-3 and its successors, have demonstrated remarkable capabilities in understanding and generating human-like text. LLMs are built on vast neural networks trained on diverse and extensive datasets, enabling them to perform various natural language processing (NLP) tasks, from language translation to complex question answering [30].

One notable application of LLMs is in creating generative AI agents, such as ChatGPT, which have proven useful across various domains [17, 33]. Despite their versatility, these agents often encounter limitations when dealing with specific, contextual knowledge related to particular entities, organizations, or individuals [8]. This shortfall stems from the models' reliance on broad training data, which may not encompass the detailed, niche information [9] required for precise and accurate responses in specialized contexts.

Researchers have increasingly focused on integrating Retrieval-Augmented Generation (RAG) techniques with LLMs to address these limitations [5, 6, 11, 21]. RAG combines the strengths of retrieval-based and generative models, enabling the AI to access and utilize external knowledge sources dynamically ([8]. This approach enhances the AI's ability to provide more accurate and contextually relevant answers by retrieving pertinent information from vast databases or knowledge repositories and then generating responses based on this specific data.

Recent studies have explored various implementations of RAG techniques, demonstrating their effectiveness across multiple applications. For instance, RAG has been utilized in the development of AI agents for customer support, where precise and context-aware responses are crucial [14, 29]. Using RAG, these agents can query extensive product databases and provide users with detailed and accurate information, improving user satisfaction and efficiency.

In the context of higher education, the potential of RAG-enhanced AI agents is particularly promising [3, 7]. These agents can offer personalized academic support, addressing university students' unique needs and inquiries. By integrating advanced NLP models with specialized academic databases, RAG-enhanced AI agents can deliver tailored advice and information, enhancing the learning experience and supporting better educational outcomes. Figure 1 below shows the interface of the AI-generated learning advisors of Van Hien University, responding to the question: 'What can the Van Hien AI Assistant support?' The answers include information on higher education regulations, student learning outcome assessment guidelines, the graduation process, and requirements for graduation output standards.

Fig. 1. AI Agent mentors Van Hien students

979-8-3315-4246-7/24/$31.00 ©2024 IEEE
DOI 10.1109/ACOMPA64883.2024.00021

This paper explores applying advanced RAG techniques in developing AI agents for academic advising in higher education. By integrating modern LLMs with a contextual knowledge database, enriching metadata, utilizing the ChromaDB data storage engine, and enhancing security through ZeroTier, we demonstrate the improvements in responsiveness and accuracy that RAG offers over traditional methods. The findings underscore the transformative potential of advanced AI technologies in revolutionizing higher education and highlight the need for continued research in this dynamic field.

II. RELATED WORK

A. Retrieval-Augmented Generation (RAG) Basic Concept

Retrieval-augmented generation (RAG) [15] is a technique that combines retrieval-based and generative models to enhance the performance of large language models (LLMs) [20]. The basic idea is to retrieve relevant documents or pieces of information from an external knowledge base and use this retrieved information to guide the generation of responses[16]. This approach addresses the limitations of purely generative models, which may not have access to specific or updated information.

Key Components:

Retriever: This component searches a large database or knowledge base to find relevant documents based on the input query [31].

Generator: The generative model, typically a transformer-based LLM, uses the retrieved documents to generate a coherent and contextually relevant response [31].

Fusion Mechanism: Combines the retrieved information with the generative model's outputs. This can be done by concatenating retrieved texts with the input or through more sophisticated integration techniques.

Workflow:

Query Input: The user provides an input query.

Document Retrieval: The retriever fetches the top-N relevant documents from the knowledge base.

Response Generation: The generator uses both the input query and the retrieved documents to generate the final response.

Applications: RAG models are used in various applications, including customer support systems, academic advisors, search engines, and any scenario where contextual and accurate information retrieval is crucial.

B. Advanced Retrieval-Augmented Generation Techniques:

Improvements and Innovations: Recent research has introduced several advancements to the basic RAG framework, enhancing its effectiveness and applicability:

Active Retrieval-Augmented Generation (FLARE) [10]: Forward-Looking Active Retrieval: This approach involves actively retrieving information during the generation process, rather than relying solely on initial retrieval. This dynamic retrieval can significantly improve response accuracy and relevance by continuously integrating new information as needed.

Domain Adaptation in RAG [27]:

Improving Domain-Specific Knowledge: Techniques have been developed to better adapt RAG models to specific domains by fine-tuning both the retriever and the generator on domain-specific data. This helps in providing more accurate and contextually appropriate responses for specialized fields (Direct MIT Press).

Auxiliary Rationale Memory (ARM-RAG) [18]: Rationale Storage and Retrieval: ARM-RAG introduces an auxiliary memory to store and retrieve chains of reasoning or rationales. This technique enhances the problem-solving capabilities of RAG models, particularly in tasks requiring step-by-step logical reasoning, such as mathematical problem-solving (Papers with Code).

Some implementation techniques of the advanced RAG model as bellow:

Multi-Stage Retrieval [4]: Advanced RAG models may implement multi-stage retrieval processes, where initial retrievals are refined through subsequent rounds to improve the quality and relevance of the retrieved documents.

Fusion Techniques [22]: Sophisticated fusion mechanisms, such as attention-based fusion or learning-to-rank models, are employed to better integrate retrieved information with the generated response.

The advancements in Retrieval-Augmented Generation (RAG) techniques have significantly improved the capabilities of large language models, making them more accurate and contextually aware. With continuous research and development, RAG and advanced RAG techniques are poised to revolutionize various applications, from customer support to specialized academic advising, by effectively integrating external knowledge with generative models. The next direction of development of Advanced RAG models is to reduce the latency when processing large-scale knowledge bases, ensuring a balance in the responses generated and adapting to real-time data.

C. ChromaDB

ChromaDB is a database specifically designed for managing and querying vector embeddings, which are numerical representations of data often used in machine learning, natural language processing, and recommendation systems [25]. Here's an overview of its structure and key features:

1) Structure of ChromaDB: The ChromaDB architecture has the following components:

Collections: ChromaDB is organized into collections, which are akin to tables in a traditional database. Each collection can store a set of related embeddings [28].

Vectors: Within each collection, ChromaDB stores vectors (embeddings), which are numerical representations of the data you want to store and query. These vectors can represent text, images, or any other type of data that can be encoded into a vector[25].

Metadata: Alongside the vectors, ChromaDB allows you to store metadata for each vector. This metadata can be any additional information related to the vector, such as the original text or other identifiers [25].

Indexes: ChromaDB uses various indexing techniques to optimize the storage and retrieval of vectors. These indexes are designed to handle high-dimensional data efficiently [25].

2) Features of ChromaDB: Some outstanding features of ChromaDB:

Vector Search: ChromaDB provides efficient vector similarity search capabilities. It allows you to perform nearest-neighbor searches, which are critical for tasks like finding similar documents, images, or other types of data.

Scalability: ChromaDB is built to handle large-scale datasets. It can manage millions of vectors while still providing fast query responses.

Multi-Modal Data Support: It can handle different types of data (e.g., text, images) by storing their respective embeddings in the same or different collections.

Integration with Machine Learning Models: ChromaDB is designed to work seamlessly with machine learning models, particularly those that generate embeddings, like transformers or convolutional neural networks.

APIs and Libraries: It provides APIs and libraries that make it easy to integrate ChromaDB into your applications, whether you're working with Python, Java, or other languages.

Metadata Filtering: You can perform searches not just based on vector similarity but also by filtering on the metadata associated with the vectors. This allows for more complex queries and better precision in retrieval.

Versioning: ChromaDB supports versioning of collections, enabling you to manage changes in your datasets over time effectively.

Data Persistence and Durability: ChromaDB ensures that your data is safely stored and can be recovered in case of failures. It supports various persistence mechanisms depending on your needs.

Distributed and Cloud-Ready: ChromaDB can be deployed in distributed environments and is optimized for cloud deployment, making it suitable for large-scale and high-availability applications.

Real-Time Updates: It supports real-time updates to the stored embeddings and metadata, allowing dynamic applications to remain up-to-date with minimal latency.

Using ChromaDB in Natural Language Processing to help handle tasks like sentence similarity, text clustering, or topic modeling. ChromaDB is particularly powerful in scenarios where you need to manage and query large volumes of vectorized data efficiently, making it a valuable tool for modern AI and machine learning applications.

III. METHOD

A. Overall the architecture

Method of AI Agent in learning advisors with Advanced RAG has steps as bellow:

1) Using large language models to create answers and interact with users.
2) Convert text into the vector by *intfloat/multilingual-e5-large model*
3) Stored vector in ChromaDB and using cosine similarity to find the information
4) Using GPT-3.5-turbo-instruct-0914 for metadata enrichment
5) Using llamaindex framework to build Agent AI background
6) Develop application by Streamlit on web, or mobile to react with users
7) Ensure security with Zero Tier access method

Figure 2 presents the overview model of the AI-generated learning advising architecture in Van Hien University based on other overview models. [1]

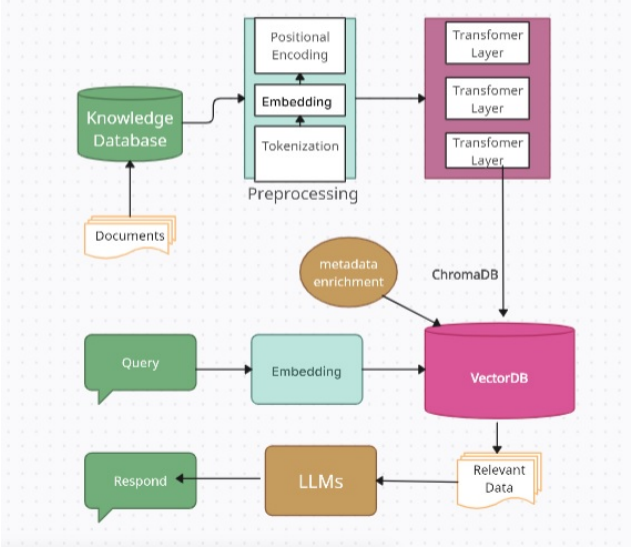

Fig. 2. Overall the architecture AI-generated learning advisors with Advanced RAG

B. Detail of model architecture

a) Preprocessing and Tokenization: The input text T is converted into tokens via a tokenizer. Suppose T is the input text as bellow:

T = "I need to know the required subjects to complete the Information Technology program"

Tokenizer will convert the text T into a sequence of tokens $\{t_1, t_2, \ldots, t_n\}$ After that, the machine will embed the Tokens. in which t_i, it mapped into a vector e_i via the embedding layer

$$e_i = E(t_i)$$

E is the embedding matrix with size V x d (V is the number of words and d is the size of the embedding vector)

b) Transformer layers: The embedding vector $\{e_1, e_2, \ldots, e_n\}$ are pass through many Transformer classes. Each Transformer class includes the following main components:

- Self-Attention mechanism

At each class, calculate attention scores between words. With query **Q**, key **K**, and value **V** are calculated as follows:

$$Q = W_Q E, K = W_K E, V = W_V E$$

Where W_Q, W_K, W_V are Weight Matrices.
Attention score A is calculated:

$$A = softmax\left(\frac{QK^T}{\sqrt{d}}\right) \quad [26]$$

Vector output **O** is:

$$O = AV$$

- Feed-Forward Network

After the self-attention step, the **O** vector is transmitted over the neural Feed-Forward Network:

$$F = ReLU(OW_1 + b_1)W_2 + b_2$$

Where W_1, W_2, b_1, b_2 are the parameters of the neural network.

- Aggregation

After going through many Transformer layers, the text string gets the context vector $\{h_1, h_2, \ldots, h_n\}$. Perform averaging or pooling to create a vector representing the entire text:

$$v = \frac{1}{n}\sum_{i=1}^{n} h_i$$

We can use other pooling mechanisms such as max-pooling or the last vector h_n depending on the model.

- Representative Vector

The final vector v, which is the representing vector of the text T, can be used for tasks such as information retrieval, text classification, or as input to the next model.

The whole process can be described mathematically as follows:

1) **Tokenization:** $T \rightarrow \{t_1, t_2, \ldots, t_n\}$
2) **Embedding:** $\{t_1, t_2, \ldots, t_n\} \rightarrow \{e_1, e_2, \ldots, e_n\}$
3) **Transformer Layers:** $\{e_1, e_2, \ldots, e_n\} \rightarrow \{h_1, h_2, \ldots, h_n\}$
4) **Aggregation:** $\{h_1, h_2, \ldots, h_n\} \rightarrow v$

Vector **v** is the final result, representing the text T as a vector in the semantic space. If the input is a set of documents $\{T_1, T_2, \ldots, T_n\}$ then each document Ti is converted to a vector vi through the *intfloat/multilingual-e5-large model*

$$v_i = f(T_i)$$

Where f is the function that maps text to vector. Vector $\{v_1, v_2, \ldots, v_n\}$ is stored in *ChromaDB*. When a text query Q is received, the system converts the query into a vector q

$$q = f(Q)$$

c) Calculating cosine similarity : To find the text that matches the query, the system calculates the similarity score between the query vector q and the storage vector vi, one of the popular ways is to use cosine similarity [12]:

$$cosine\_similarity(q, v_i) = \frac{q \cdot v_i}{\|q\|\|v_i\|} \quad [13]$$

Where (.) is the dot product and $\|q\|$ is the Euclidean standard of q vector.

Cosine similarity has a value in the range $[-1, 1]$. The higher the value, the better the similarity between the vectors.

To get the most suitable return results, the AI agent selects the v_i vectors that have the highest cosine similarity to the query vector **q**.

Example:
Suppose we have two vectors
A = [1, 2, 3]
B = [4, 5, 6]
Step 1: Calculate the dot product $A \cdot B$:

$$A \cdot B = (14) + (25) + (36) = 4 + 10 + 18 = 32$$

Step 2: Calculate the magnitude of each vector:

$$\|A\| = \sqrt{1^2 + 2^2 + 3^2} = 3.472$$

$$\|B\| = \sqrt{4^2 + 5^2 + 6^2} = 8.775$$

Step 3: Calculate the cosine similarity:

$$Cosine\ Similarity = \frac{32}{3.472\ x\ 8.77} = 0.974$$

So, the cosine similarity between vectors A and vector B is approximately 0.974, indicating a high degree of similarity (close to 1 means the vectors are very similar, while 0 would indicate they are orthogonal, or not similar at all).

d) Query in ChromaDB: The query process in *ChromaDB* can be described as follows:

1) Transfer the query text Q into vector q.
2) Cosine similarity between q and add vector v_i in database.
3) Sort all vectors v_i in descending order of cosine similarity.
4) Returns the text for which the best matching vector map has the highest cosine similarity.

Formula:

$$Result = Top - k(\{T_i | cosine\_similarity(q, v_i)\})$$

Where Top-k is a function that returns the k documents with the highest cosine similarity.

The storage of data in ChromaDB can be performed in several different ways as follows:

- *Store vectors as array*: This way is the most simple and the most popular. Each vector is stored as an array containing values. Example: 4-dimensional array [1, 2, 3, 4]

- *Store vectors as string*: For vectors with unknown dimensions, or variable dimensions, the data is stored as string. Example: '1234'
- *Store vectors as matrix*: Sometimes, the vector is stored as a row or a collum of the matrix.
  Example: Vector [1, 2, 3, 4] is a row of matrix.
- *Store vectors as JSON*: The JSON structure allows storing complex data, including vectors.
  Example { "vector": [1,2,3,4]}
- *Store vectors as blob*: in some cases, the vector is stored as a blob object and it is not encoded. It is only accessed and used by the application.

Depending on factors such as retrieval capability, performance, and flexibility in data processing, the appropriate method for storing vectors should be selected.

e) Using GPT-3.5-turbo-instruct-0914 for metadata enrichment: Each input document contains Metadata information to further clarify the document. It stores metadata information of the document, such as name, source, last updated date, owner, or any other relevant details.

Metadata is stored as a key-value map, where the key is of type String, and the value can be one of the basic data types.

Benefits of using GPT-3.5-turbo-instruct-0914 [19] for metadata enrichment:

- Improved Instruction Following:

This version is fine-tuned specifically to better understand and follow user instructions. It is optimized to generate responses that are aligned with the prompt's requirements.

- Enhanced Conversational Abilities:

The model has improved conversational coherence and is more adept at maintaining context over longer interactions.

- Better Handling of Edge Cases:

It is more robust when dealing with ambiguous or complex instructions, reducing the likelihood of generating incorrect or irrelevant responses.

- Fine-Tuned Response Generation:

The model produces more concise, accurate, and contextually relevant responses, making it suitable for use cases where clarity and precision are critical.

- Customizability:

Users can provide detailed instructions or context, and the model is capable of generating responses that align closely with specific requirements.

- Support for Diverse Use Cases:

GPT-3.5-turbo-instruct-0914 is versatile and can be applied to a wide range of applications, from generating content to answering questions, providing summaries, and more.

f) Using Zero Tier access method: When integrating ZeroTier with a chatbot, its features create secure and private communication channels. Here are some key features of using ZeroTier access in the context of a chatbot: [32]

- Secure Communication:

Encrypted Connections: ZeroTier encrypts all communication between devices on the network, ensuring that interactions between the chatbot and users are secure and protected from unauthorized access.

Private Networking: The chatbot can operate within a private virtual network, isolating its traffic from the public internet, which adds an extra layer of security.

- Remote Access:

Global Reach: ZeroTier allows you to access the chatbot from anywhere in the world as long as you are connected to the same ZeroTier network. This is particularly useful for remote teams or distributed applications.

Consistent Access: Users can interact with the chatbot as if they were on the same local network, ensuring consistent and reliable access without the need for complex VPN setups.

- Simplified Network Management:

Easy Setup: Integrating ZeroTier with a chatbot doesn't require extensive network configuration. Once both the server hosting the chatbot and the client devices are connected to the same ZeroTier network, they can communicate directly.

Scalability: ZeroTier makes it easy to add or remove devices (e.g., new chatbot instances or client devices) from the network, providing scalability without additional infrastructure.

- Cross-Platform Compatibility:

Multiple Device Support: ZeroTier supports various operating systems, allowing the chatbot to interact with clients on different platforms (Windows, macOS, Linux, Android, iOS) seamlessly.

Device Flexibility: You can run the chatbot on different devices and still maintain secure communication channels with all clients connected to the ZeroTier network.

- Access Control:

Network Permissions: ZeroTier allows fine-grained control over who can join the network and access the chatbot. You can set up rules to restrict access based on device identity or other criteria.

User Management: Easily manage users who can interact with the chatbot by controlling their access to the ZeroTier network.

- Low Latency:

Efficient Routing: ZeroTier optimizes the routing of data packets between devices on the network, often leading to lower latency than traditional VPN solutions. This ensures faster response times for chatbot interactions.

- Reduced Network Overhead:

Minimal Infrastructure: By using ZeroTier, you avoid the need for extensive on-premises networking infrastructure. This reduces the cost and complexity of managing a secure environment for your chatbot.

Use Case Example:

Internal Company Chatbot: A company could deploy a chatbot on a secure ZeroTier network, allowing employees to interact with the bot to access sensitive company information or perform internal tasks securely, no matter where they are located.

By integrating ZeroTier with a chatbot, you can ensure that the communication is secure, private, and efficient, while also benefiting from the flexibility and ease of use that ZeroTier provides.

IV. RESULT

A. DATA INPUT PROCESSING

We utilize data from Van Hien University's regulations concerning student advisory issues, including:

- Credit-based Training Regulations in Van Hien University
- Grades and Assessment Regulations
- University Graduation Outcomes Requirements
- Guidance for Course Registration
- Regulations on the Implementation of Theses and Graduation Projects
- Internship Regulations
- Policies and Scholarships for Students
- Regulations on Procedures and Processes for Students within the University

These advisory topics have distinct contexts, making it challenging for a conventional AI Agent to accurately address the necessary queries.

The data is stored in PDF format. During the preprocessing step, the text segments will be chunked depending on the length and the content of the text, each chunk might consist of a few paragraphs or around 500 to 1,000 words.

The text is then cleaned by removing irrelevant words, converting the text to lowercase, and removing stopwords unless they are contextually important.

After cleaning, the text is broken down into tokens (words or subwords). Metadata is attached to each chunk, such as the document title, section headings, page number, or other relevant identifiers. This metadata can help the AI agent understand the context of each chunk.

Next, the intfloat/multilingual-e5-large model is used to generate embeddings (vectors) for the text in multiple languages, where each text input is converted into a fixed-size vector of 1,024 dimensions. With text like this, it could generate thousands of vectors, depending on the content of the text. These embed vectors are stored and retrieved in ChromaDB.

B. EXPERIMENT

This evaluation will compare ChatGPT-3.5 with our AI Agent using Advanced RAG under the same conditions data, user and query.

The same metrics will be used for evaluation:

Precision@k (P@k): The fraction of the top-k retrieved results that are relevant [24].

Recall@k (R@k): The fraction of all relevant results that are retrieved in the top-k results [24].

Mean Reciprocal Rank (MRR): The average of the reciprocal ranks of the relevant results [23].

Response Quality: Student-annotated score based on relevance, accuracy, and completeness.

Results:

ChatGPT-3.5:

Top-k = 5:

Precision@5: 0.6 (3 out of 5 responses are relevant)

Recall@5: 0.55 (55% of all relevant responses are retrieved)

MRR: 0.65 (Relevant responses are more frequently found within the top 2 positions)

Response Quality: 6.5/10 (The responses are more accurate and contextually relevant than earlier versions, with better specificity)

Top-k = 10:

Precision@10: 0.65 (6.5 out of 10 responses are relevant, rounded up to 7)

Recall@10: 0.70 (70% of all relevant responses are retrieved)

MRR: 0.70 (Relevant responses are spread across the top 5 positions, with a few lower-ranked results)

Response Quality: 7.0/10 (Increased Top-k improves recall, but precision slightly decreases, and the relevance is slightly diluted)

AI Agent with Advanced RAG:

Top-k = 5:

Precision@5: 0.92 (4.6 out of 5 responses are relevant, rounded up to 5)

Recall@5: 0.80 (80% of all relevant responses are captured in the top 5 results)

MRR: 0.88 (Relevant responses are almost always within the top 2 positions)

Response Quality: 9.0/10 (Extremely relevant and accurate responses, directly pulling from specific document sections, providing highly detailed advice)

Top-k = 10:

Precision@10: 0.85 (8.5 out of 10 responses are relevant, rounded up to 9)

Recall@10: 0.95 (95% of all relevant responses are retrieved)

MRR: 0.85 (Most relevant responses are within the top 3 positions, with slightly less focused results as k increases)

Response Quality: 8.7/10 (The system retrieves almost all relevant information, maintaining high relevance and specificity)

Chat-GPT 3.5: While Chat-GPT can generate useful and coherent responses, it may struggle with providing specific advice from the listed documents, particularly when fine details or exact document references are needed. It relies heavily on its pre-existing knowledge and may not always retrieve the most relevant content in the top positions.

AI Agent with Advanced RAG: This system excels in retrieving precise and contextually relevant information directly from the documents. The retrieval component ensures that the most pertinent chunks are brought into context, and the generative model can then tailor the advice based on these. As a result, the advice is more accurate and aligned with the specific contents of the documents.

V. CONCLUSION

The application of advanced Retrieval-Augmented Generation (RAG) in AI Agents for student advising at Van Hien

University represents a significant leap forward in addressing the complexities of student queries across various contexts. For advising students on the listed documents, the AI Agent with Advanced RAG offers superior performance, particularly in terms of precision and response quality. The ability to retrieve specific document content and generate context-aware responses leads to better overall advice.

Chat-GPT, while still useful, may not perform as well in scenarios requiring detailed knowledge retrieval and might provide more generic advice that could lack the specificity needed for accurate student guidance.

However, while the results are promising, there are several limitations that warrant further exploration. First, the reliance on predefined datasets means that the AI Agent may still encounter difficulties in handling novel or evolving queries that fall outside the scope of its training data. Additionally, the current system's performance may degrade when processing highly nuanced or contextually ambiguous queries, indicating a need for more sophisticated context disambiguation techniques.

In the future, the research focus on expanding the contextual understanding of the AI Agent by incorporating dynamic learning mechanisms that allow the system to adapt to new information in real time. Moreover, enhancing the integration of multi-modal data—such as visual or auditory inputs—could further improve the AI's ability to understand and respond to complex advisory situations. Finally, user feedback loops and continuous model retraining should be implemented to refine the system's accuracy and relevance over time, ensuring that it remains responsive to the diverse and changing needs of students.

The potential for expanding context-based AI agent applications to other domains is enormous. By adapting core models to domain-specific data, incorporating advanced contextual discrimination techniques, and handling specific queries, AI systems can be applied to industries such as healthcare, finance, law, customer service, and human resources. In these cases, they can provide more efficient, personalized, and scalable solutions to improve decision-making, automate routine tasks, and enhance user experiences.

REFERENCES

[1] L. Bansal. Advance rag- improve rag performance. URL `https://luv-bansal.medium.com/`.

[2] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

[3] S. Dakshit. Faculty perspectives on the potential of rag in computer science higher education. *arXiv preprint arXiv:2408.01462*, 2024.

[4] N. R. Davidson, S. K. Sattiraju, and U. Gangadharaiah. Multi-stage fine-tuning process for optimizing small llms in rag applications, 2024.

[5] Y. Ding, W. Fan, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li. A survey on rag meets llms: Towards retrieval-augmented large language models. *arXiv preprint arXiv:2405.06211*, 2024.

[6] D. Fleischer, M. Berchansky, M. Wasserblat, and P. Izsak. Rag foundry: A framework for enhancing llms for retrieval augmented generation. *arXiv preprint arXiv:2408.02545*, 2024.

[7] L. Galstyan, H. Martirosyan, E. Vardanyan, and K. Vahanyan. Smartadvisor university chatbot spring 2024, 2024.

[8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[9] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

[10] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023.

[11] Z. Jiang, X. Ma, and W. Chen. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*, 2024.

[12] K. Juvekar and A. Purwar. Cos-mix: cosine similarity and distance fusion for improved information retrieval. *arXiv preprint arXiv:2406.00638*, 2024.

[13] F. Karabiber. Cosine similarity. URL `https://www.learndatasci.com/`.

[14] V. Katragadda. Leveraging intent detection and generative ai for enhanced customer support. *Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023*, 5(1):109–114, 2024.

[15] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[16] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*, 2022.

[17] C. Ma, J. Zhang, Z. Zhu, C. Yang, Y. Yang, Y. Jin, Z. Lan, L. Kong, and J. He. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*, 2024.

[18] E. Melz. Enhancing llm intelligence with arm-rag: Auxiliary rationale memory for retrieval augmented generation. *arXiv preprint arXiv:2311.04177*, 2023.

[19] Nextideatech. Openai's new gpt 3.5 instruct. URL `https://blog.nextideatech.com/`.

[20] Nvidia. What is retrieval-augmented generation, aka rag? URL `https://blogs.nvidia.com`.

[21] A. Purwar et al. Evaluating the efficacy of open-source llms in enterprise-specific rag systems: A comparative study of performance and scalability. *arXiv preprint*

arXiv:2406.11424, 2024.

[22] Z. Rackauckas. Rag-fusion: a new take on retrieval-augmented generation. *arXiv preprint arXiv:2402.03367*, 2024.

[23] E. A. Team. Mean reciprocal rank (mrr) explained, . URL https://www.evidentlyai.com/ranking-metrics/mean-reciprocal-rank-mrr.

[24] E. A. Team. Precision and recall at k in ranking and recommendations, . URL https://www.evidentlyai.com/ranking-metrics/precision-recall-at-k.

[25] trychroma.com. Chroma. URL https://docs.trychroma.com/.

[26] A. Vaswani. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[27] A. Wang, L. Song, G. Xu, and J. Su. Domain adaptation for conversational query production with the rag model feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9129–9141, 2023.

[28] www.datacamp.com. Chroma db tutorial: A step-by-step guide. URL https://www.datacamp.com/tutorial/.

[29] Z. Xu, M. J. Cruz, M. Guevara, T. Wang, M. Deshpande, X. Wang, and Z. Li. Retrieval-augmented generation with knowledge graphs for customer service question answering. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2905–2909, 2024.

[30] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, and X. Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6): 1–32, 2024.

[31] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu. Evaluation of retrieval-augmented generation: A survey. *arXiv preprint arXiv:2405.07437*, 2024.

[32] Zerotier. Zerotier review: Everything you need to know about zerotier in 2023. URL https://www.zerotier.com/.

[33] H. Zhang, A. B. Sediq, A. Afana, and M. Erol-Kantarci. Generative ai-in-the-loop: Integrating llms and gpts into the next generation networks. *arXiv preprint arXiv:2406.04276*, 2024.