Enhancing Generative AI Chatbot Accuracy Using Knowledge Graph

Ajay Bandi[1(✉)], Jameer Babu[1], Ruida Zeng[2], and Sai Ram Muthyala[1]

[1] School of Computer Science and Information Systems, Northwest Missouri State University, 800 University Dr, Maryville, MO 64468, USA
ajay@nwmissouri.edu

[2] Department of Computer Science, Brown University, Providence, RI 02912, USA

Abstract. In recent years, generative AI chatbots have significantly improved in their ability to simulate human-like conversations. However, ensuring the accuracy and contextual relevance of their responses remains a challenge. This paper presents an innovative approach to enhancing the accuracy of generative AI chatbots by integrating knowledge graphs using Neo4j. We demonstrate how combining structured data from Knowledge Graphs with advanced large language models can result in more accurate and context-aware chatbot interactions. By implementing this approach, we aim to provide a robust framework for developing intelligent chatbots that can deliver precise and contextually appropriate responses. We created three categories of test cases: Data-Relevant Inquiries, Non-Contextual Queries, and Contextually Relevant but Data-Irrelevant Questions. The accuracy obtained for the data-relevant test cases was 91.44%.

Keywords: Retrieval and Augmented Generation (RAG) · Large Language Model (LLM) · Knowledge graphs · Chatbot · Generative AI · Word embeddings · Vector index · Cypher · Similarity search · Neo4j

1 Introduction

Generative AI refers to artificial intelligence systems that can create new content or data. Unlike traditional AI models, which primarily classify or predict outcomes based on existing data, generative AI models can generate text, images, music, and even complex designs from scratch [1]. Most generative AI models are built using deep learning techniques, particularly neural networks like Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Transformers.

A chatbot is a software application designed to simulate human conversation through text or voice interactions [6, 22]. Chatbots can be rule-based, following predefined scripts, or more advanced AI-powered, utilizing AI to understand and respond to user inputs dynamically. Rule-based chatbots operate based on a set of predefined rules. They can handle simple queries but struggle with more complex conversations. Whereas, AI-powered chatbots use natural language processing (NLP) and machine learning to understand and respond to user inputs more agility. They can learn from interactions

W. Feng et al. (Eds.): SEDE 2024, CCIS 2244, pp. 157–167, 2024.
https://doi.org/10.1007/978-3-031-75201-8_11

and improve over time [7]. Customer support, personal assistants, and e-commerce are a few applications.

Rule-based chatbots provide responses based on a fixed set of predefined options, typically using if-then logic or decision trees. They struggle with limited flexibility. Whereas, Generative AI chatbots use machine learning models, such as GPT (Generative Pre-trained Transformer), to generate responses on the fly. They understand context [26] and generate natural language responses providing high flexibility. Rule-based chatbots rely on keyword matching and pattern recognition to understand [26] user input. They lack awareness over complex phrasings and their interactions look artificial [2]. Generative AI chatbots use advanced NLP techniques to understand and interpret user input. They interact in a natural and coherent way. The knowledge and capabilities of rule-based chatbots are fixed and must be manually updated. They do not learn from interactions. They cannot improve their performance over time without human intervention. Any updates or improvements require reprogramming. Generative AI chatbots can learn from new data and interactions. They can be fine-tuned and updated with new information, improving their performance and accuracy. They can adapt to new types of queries and improve their responses based on user feedback and additional training data. One such adaptation is Retrieval Augmented Generation (RAG) [6]. RAG is an advanced NLP technique that combines the strengths of retrieval-based and generative models. It improves the accuracy and contextual relevance of generated responses by incorporating external knowledge through a retrieval component. RAG offers numerous advantages, including enhanced accuracy, better contextual understanding, scalability, flexibility, reduced hallucination, and efficient information retrieval [7]. These benefits make RAG a powerful tool for various applications, such as customer support, research assistance, content creation, and question-answering systems.

In this paper, we aimed to improve the accuracy by creating three different test categories namely three different test categories namely: *Data-Relevant*, which focuses on known data; *Non-Contextual*, which deals with out-of-context data; and *Contextually Relevant but Data-Irrelevant*, based on contextual data but previously not encountered by the chatbot. The remainder of the paper is organized as follows. Section 2 describes related work on chatbots, RAG, and knowledge graphs. Section 3 discusses the implementation of the chatbot, detailing the tools, technologies, and steps involved. Section 4 presents the results and evaluation. Section 5 provides research conclusions.

2 Related Work

Bandi and Kagitha in their article [2] presented the different phases of generative AI project life cycle with the implementation of chatbot using generative AI LLMs. While we used those different phases of generative AI project, we observed that there are limitations in the evaluating the responses of the chatbot. They have tested only on the contextually relevant data and their test cases are completely based on the trained data able to get one hundred percent accurate results. The same article did not address if the prompts are not relevant to the trained data and the results for such prompts from the chatbot can't be predicted.

LLMs are empowered with Retrieval Augmented Generation (RAG) [4] in enabling them to deliver correct and contextual answers. Particularly in the space of open-domain

question answering [5], educational question-answering system [9], adaptable AI assistant for network management [15], medical consultation chatbot [3, 17]. RAG model has its own applications, methodologies and their effectiveness, techniques and challenges [6, 7]. LLMs utilising RAG have a proven ability in extracting relations from the text and corresponding models were proven effective [8].

These RAG enabled Chatbots when enhanced with Knowledge-Graphs [10, 18, 20] can improve customer service question answering [11]. LLMs are capable of performing generative graph analytics, focusing on query processing, learning [12]. Mainly, Also there are other approaches such as hybrid context retrieval-augmented generation, that combines knowledge graphs and vector databases [13], graph-based retrieval-augmented generation approach for query-focused summarization [16].

There is an inadvertent need to integrate large language models with vector databases [14]. Neo4j [24, 25] comes with the combination of Knowledge graphs and Vector indices [19]. These vector indices are built based on word embeddings [21].

Our research focuses on increasing the accuracy of Chatbot by implementing a Knowledge-Graph enabled RAG combined with vector indices by considering all the types of data such as relevant, irrelevant, contextually relevant but data irrelevant. We developed a chatbot for the MSACS program at Northwest Missouri State University, which constructs knowledge graphs and vector indices using data available on NWMSU websites related to courses, professors, schedules, and more. This chatbot utilizes RAG to provide responses to student prompts or queries.

3 Chatbot Implementation

This section presents the details of the chatbot implementation. We developed a chatbot for prospective students of the Master of Science in Applied Computer Science (MSACS) program. This AI-powered chatbot assists prospective and current students in obtaining information using the following tools and technologies.

3.1 Tools and Technologies

- **Python**: The primary programming language used to develop the chatbot
- **Neo4j**: A graph database that aids in constructing and managing the knowledge graph, with efficient storage and retrieval of entities and relationships within the MSACS program data.
- **LangChain**: LangChain is an open-source framework designed to aid developers in creating applications that utilize large language models (LLMs) for natural language processing (NLP). It offers a variety of tools, components, and interfaces that streamline the development process and facilitate the integration of language models with external data sources such as Neo4j.
- **OpenAI GPT-3.5-Turbo-0125 LLM**: OpenAI GPT-3.5-Turbo-0125 is an advanced version of the GPT-3 model, designed to be faster and more efficient. The "0125" indicates specific enhancements and adjustments in this version. It can handle a variety of language-related tasks, such as generating text, translating languages, summarizing content, and question-answering chats.
- **Tiktoken**: Library used for tokenizing the text.

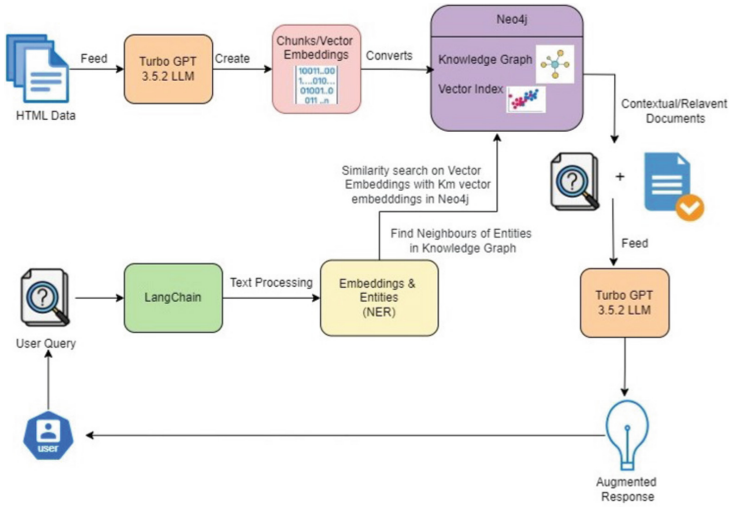

Fig. 1. Architecture of chatbot

3.2 Methodology

This section details the different steps in implementing the MSACS chatbot. Figure 1 represents the high-level architecture of the chatbot, illustrating how different components interact and integrate with each other, providing an easier understanding of the overall system layout and data flow. We implemented our chatbot by adopting practical guide to constructing and retrieving information from knowledge graphs in RAG applications with Neo4j and LangChain by Tomaz Bratanic [26]. Figure 2 shows the sequence diagram, which demonstrates the dynamic behavior of the system with different actors, visualizing interactions between components and representing the system behavior over time. All the files related to the implementation of the paper are provided in the GitHub repo.[1]

1. Extract data from the Html pages: Using data from MSACS websites, WebBase Loaders of *LangChain* fetch and load the content of multiple web pages. This content is stored in a buffer for further processing.
2. Split the data into chunks: Tokenizing data is essential for effectively leveraging large language models. It transforms raw text into a structured and standardized format that the model can process efficiently. By capturing semantic meaning, reducing complexity, and enabling efficient computation, tokenization plays a fundamental role in the performance and capabilities of LLMs. Using *TikToken*, the buffered data is split into chunks of 512 tokens with an overlap of 24 tokens. The overlap ensures that contextual relevance is maintained across the tokenized data. This step facilitates breaking down the content into more digestible parts while preserving context.

[1] https://github.com/bandiajay/Enhancing-Generative-AI-Chatbot-Accuracy-Using-Knowledge-Graph.

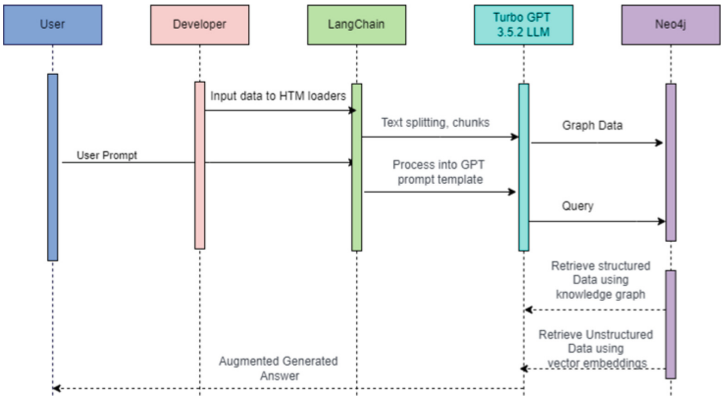

Fig. 2. Interactions of chatbot's components in sequence diagram

3. Transform the Data into Knowledge Graphs Using LLMs and Store in Neo4j Database: We use the *gpt-3.5-turbo-0125* LLM to implement the chatbot. By controlling the randomness of the model's responses, ensuring deterministic and consistent outputs. The tokenized data from the previous step is transformed into a graph-based format, known as a *Knowledge Graph*, as shown in Fig. 3. This aids in advanced document analysis, enhanced querying of textual data, and knowledge extraction using a graph transformer called *LLM- GraphTransformer*. The nodes in the graph contain primary entity information, and the original source of each document is retained within the graph, enhancing traceability and context preservation. This transformation is crucial for complex document analysis tasks, enabling more sophisticated knowledge extraction and representation. The entire pipeline—from loading and splitting documents to transforming them into graph documents—is designed to enhance the processing and analysis capabilities of textual data.

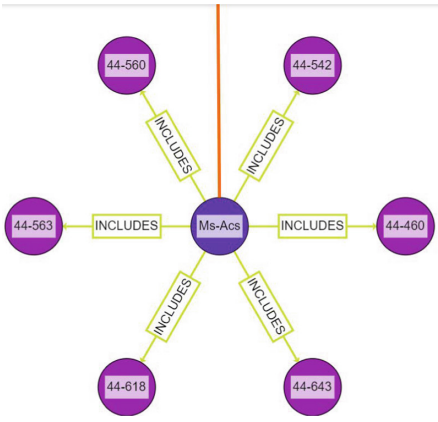

Fig. 3. Knowledge Graph of MSACS program courses

4 Create Vector Indices on Knowledge Graphs and Store Word Embed- dings: Vector indices enable rapid and scalable search and retrieval operations, and they help manage high-dimensional data, optimizing performance and enabling sophisticated data-driven solutions. A vector index is created from the existing graph developed in the previous step. OpenAI's embedding model is used to convert the data into vector representations. The search types are hybrid, effectively using both vector-based search methods like cosine similarity and traditional search methods. Full-text indices in Neo4j are also utilized for efficient text searches, enhancing query performance and enabling advanced search capabilities.
5. Give a Prompt: A prompt is provided, which may or may not be related to the extracted data. The LLM should respond with contextual knowledge derived from the previously developed knowledge graphs.
6. Retrieve Structured Context from Neo4j Using Knowledge Graphs: The structured retriever operates on the knowledge graph, extracting specific entities such as courses, professors, requirements, and projects using prompt-based entity extraction. For each extracted entity, it constructs a full-text search query and retrieves relevant nodes and their relationships from the Neo4j graph database, including both outgoing and incoming relationships. The results are formatted into a structured output.
7. Retrieve Unstructured Context from Neo4j Using Similarity Search on Vector Embeddings: The unstructured retriever performs a similarity search on the vector indices using the prompt. This search aims to provide flexible, content-based retrieval without focusing on predefined entity types or relationships.
8. Augment Both Contexts and Provide to LLM: The structured and unstructured retrievers extract and query entities related to the user's prompt, collecting their neighborhood in the graph database to create a single context. This augmented context is then passed to the LLM.
9. Get the Response from LLM: A template is created with the user's question and the RAG's context, forming a coherent prompt. The LLM then responds to this prompt.

4 Results and Evaluation

The chatbot's accuracy is evaluated based on the following three categories of test cases:

1. **Data-Relevant Inquiries**: These are questions directly related to the specific data provided. The chatbot needs to accurately retrieve and respond to queries based on the information it has been trained on or has access to within the dataset.
   Example: "Is a score of 105 in Duolingo sufficient for admission?" This question requires the chatbot to reference the admissions criteria data for Duolingo scores.
2. **Non-Contextual Queries**: These are questions that are not related to the context of the data provided. Such queries are outside the scope of the chatbot's intended knowledge base, and it should ideally recognize these and handle them appropriately, potentially by stating that the information is not available.
   Example: "Does the sun rise in the east?" This question is a general knowledge inquiry and not relevant to the specific dataset or context the chatbot is designed to address.

3. **Contextually Relevant but Data-Irrelevant Questions**: These questions are within the broader context, but are not specifically covered by the data the chatbot has. The chatbot should be able to recognize the context and acknowledge the relevance, yet correctly indicate that the specific data point is not available.
   Example: "How long does it take to commute from campus to downtown during rush hour?" This question is relevant to someone considering the practical aspects of attending the institution, but it falls outside the scope of the chatbot's knowledge on admissions criteria. The chatbot should acknowledge the relevance of commuting concerns, but state that it doesn't have specific data regarding commuting times.

By organizing the test cases in this way, we can thoroughly assess how well the chatbot handles different types of questions. This helps us gauge its strength in understanding and responding appropriately across various contexts. We've prepared a set of 1000 test cases using generative AI based on the following prompt. The test cases are designed to mimic the types of prompts the chatbot would encounter in real-life scenarios. To create a balanced dataset, we included an equal mix of 50% true prompts and 50% false prompts. The goal of using true/false format questions is to construct a confusion matrix. After inputting these prompts into the chatbot, we manually review the results to classify them as true positives (TP), false positives (FP), true negatives (TN), or false negatives (FN). This manual validation allows us to accurately calculate the confusion matrix values.

Precision:

$$P = \frac{TP}{TP + FP}$$

Accuracy:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall:

$$R = \frac{TP}{TP + FN}$$

Using the website URLs from the MSACS website, generate 1000 test cases, with nearly 330 each in all three categories such that the test case is similar to the prompt and the answer is always either True or False. 1. Questions relevant to the above data 2. Questions that are not in context to the data 3. Questions that are in context, but not related to the above data.

Each of these categories represents different types of questions the chatbot may encounter in real-world scenarios. To evaluate its performance, we used metrics such as precision, accuracy, and recall, which are crucial for assessing how effectively the chatbot handles various contexts. Table 1 presents the confusion matrix, which shows the number of true positives, false positives, true negatives, and false negatives for each category of test cases. Table 2 provides the final precision, accuracy, and recall values for all three categories of test cases.

In the Data-Relevant Inquiries group, the chatbot excelled with a precision of 95.33%, an accuracy of 91.45%, and a recall of 91.48%. These impressive metrics highlight the chatbot's effectiveness in dealing with straightforward questions directly related to the provided data. The high precision means that the chatbot's responses are mostly relevant and accurate, ensuring users get the correct information. The high recall indicates that the chatbot successfully retrieves a significant portion of relevant information, demonstrating its thoroughness. The accuracy score reflects the overall correctness of the chatbot's responses in this category, showing its reliability when answering data-specific questions.

By categorizing the test cases this way, we can comprehensively evaluate the chatbot's ability to understand and respond accurately across different types of inquiries. This approach helps us measure its robustness and contextual understanding in a variety of scenarios. We prepared a total of 1000 test cases using these criteria to ensure a thorough assessment. The Non-Contextual Queries group posed a significant challenge for the chatbot. While the precision was notably high at 98.92%, indicating that the chatbot correctly identified irrelevant queries with great accuracy, the recall dropped significantly to 54.12%. This means that the chatbot failed to identify nearly half of the irrelevant instances, missing a substantial number of them. This insight suggests that while the chatbot broadly classified positive and negative irrelevant queries correctly, it struggled to catch all irrelevant queries. The overall accuracy was 76.76%, showing that the chatbot could correctly identify a substantial proportion of instances, but still missed a significant number of relevant responses. This discrepancy indicates that although the chatbot performed well in recognizing true positives (correctly identifying irrelevant queries), it had difficulty in identifying all irrelevant responses, especially when the context of the queries did not match the available data. This performance gap highlights the need for improvements in the chatbot's ability to understand and classify non-contextual queries, ensuring it can better distinguish between relevant and irrelevant questions in various contexts.

Table 1. Confusion Matrix

	Predicted					
	Category I		Category II		Category III	
Actual positive	204 (TP)	19 (FN)	92 (TP)	78 (FN)	82 (TP)	141 (FN)
Actual negative	10 (FP)	106 (TN)	1 (FP)	169 (TN)	4 (FP)	104 (TN)

In the Contextually Relevant but Data-Irrelevant Questions group, the chatbot faced moderate difficulty. The main reason for it is *Hallucination.* **Hallucination** in the context of language models refers to the generation of text that is plausible-sounding but factually incorrect or nonsensical. GPT-3.5-Turbo-0125, like other large language models, can sometimes produce information that is fabricated or not grounded in its training data. It achieved a precision of 95.35%, indicating that when the chatbot made a prediction, it was highly likely to be correct. However, the recall dropped significantly to 36.77%, revealing that a large number of relevant responses were not retrieved by the chatbot. The overall accuracy was relatively low at 56.19%, suggesting that many of the chatbot's predictions

Table 2. Performance metrics for different categories of test cases

	Data-relevant	Non-contextual	Contextually relevant but data-irrelevant	Overall
Precision	95.327	98.93	95.348	97.512
Accuracy	91.445	76.76	56.193	75.698
Recall	91.479	54.12	36.771	61.583

were incorrect when considered in total. These performance metrics reflect the inherent challenge of dealing with questions that are contextually relevant but lack specific data within the chatbot's training set. This situation hampers the chatbot's ability to provide accurate and comprehensive responses. The lower recall and accuracy in this category underscore the difficulty the chatbot encounters when it needs to navigate incomplete or missing contextual data. This highlights the need for the chatbot to improve its ability to recognize and handle questions that are relevant to the context but fall outside the scope of its available data, ensuring it can better serve users with accurate information even when specific details are not directly available.

Overall, the chatbot's performance across all test cases was quite strong. For the data-relevant inquiries, an average precision of 95.32%, accuracy of 91.44%, and recall of 91.49%. These numbers show that the chatbot is performance is good at identifying relevant information when it's there, meaning its answers are generally reliable. However, there's room for improvement, especially in how it handles tricky questions. For instance, when it comes to Non-Contextual Queries and Contextually Relevant but Data-Irrelevant questions, the chatbot's recall is much lower. This means it often misses relevant information in these harder scenarios, even though it does well when it does make a prediction. The high precision shows that the chatbot's responses are accurate when it decides to answer, but the lower recall indicates it doesn't always catch everything it should. To make the chatbot even better, we need to focus on helping it deal with questions that are outside its usual scope or where the context is not clear. Moving forward, we should work on enhancing the chatbot's ability to understand and respond to a wider range of questions, particularly those that are more complex or less directly related to its data. This will help improve its recall and overall accuracy, making it more effective and reliable for users in all kinds of situations.

5 Conclusion

In conclusion, our methodology demonstrated significant progress in addressing the common limitations of LLMs, such as hallucinations, while retaining the benefits of a Graph-RAG. The chatbot performed well with data-relevant queries, showing a strong precision of precision of 95.32%, accuracy of 91.44%, and recall of 91.49%. However, it faced challenges with non-contextual queries and those that were contextually relevant but data-irrelevant, as evidenced by a lower recall of 61.58%. Improving the chatbot's ability to understand and manage context will be crucial for enhancing its performance

in these more complex scenarios. By refining the algorithms to better handle these types of queries, we can boost both recall and overall accuracy, making the chatbot more effective and reliable. Our research establishes a promising framework for developing intelligent chatbots that leverage the strengths of Generative AI and Knowledge Graphs. This combination paves the way for creating more accurate, efficient, and reliable AI-driven communication tools. Such advancements have broad applications across various fields, including education and customer service, promising to revolutionize how we interact with AI in many aspects of our daily lives.

References

1. Bandi, A., Adapa, P.V.S.R., Kuchi, Y.E.V.P.K.: The power of generative AI: a review of requirements, models, input–output formats, evaluation metrics, and challenges. Future Internet **15**(8), 260 (2023)
2. Bandi, A., Kagitha, H.: A case study on the generative AI project life cycle using large language models. In: Proceedings of 39th International Confererence, vol. 98, pp. 189–199 (2024)
3. Naseem, U., Bandi, A., Raza, S., Rashid, J., Chakravarthi, B.R.: Incorporating medical knowledge to transformer-based language models for medical dialogue generation. In: Proceedings of the 21st Workshop on Biomedical Language Processing, pp. 110–115 (2022)
4. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Ku¨ttler, H., Lewis, M., Yih, W.T., Rockt¨aschel, T.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv. Neural Inf. Process. Syst. **33**, 9459–9474 (2020)
5. Siriwardhana, S., Weerasekera, R., Wen, E., Kaluarachchi, T., Rana, R., Nanayakkara, S.: Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. Trans. Assoc. Comput. Linguistics **11**, 1–17. MIT Press (2023)
6. Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Cui, B.: Retrieval-augmented generation for AI-generated content: a survey. arXiv preprint arXiv:2402.19473 (2024)
7. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H.: Retrieval-augmented generation for large language models: a survey. arXiv preprint arXiv:2312.10997 (2023)
8. Efeoglu, S., Paschke, A.: Retrieval-augmented generation-based relation extraction. arXiv preprint arXiv:2404.13397 (2024)
9. Bui, T., Tran, O., Nguyen, P., Ho, B., Nguyen, L., Bui, T., Quan, T.: Cross-data knowledge graph construction for llm-enabled educational question-answering system: a case study at HCMUT. arXiv preprint arXiv:2404.09296 (2024)
10. Pan, J.Z., Razniewski, S., Kalo, J.C., Singhania, S., Chen, J., Dietze, S., Jabeen, H., Omeliyanenko, J., Zhang, W., Lissandrini, M., et al.: Large language models and knowledge graphs: opportunities and challenges. arXiv preprint arXiv:2308.06374 (2023)
11. Xu, Z., Cruz, M.J., Guevara, M., Wang, T., Deshpande, M., Wang, X., Li, Z.: Retrieval-augmented generation with knowledge graphs for customer service question answering. arXiv preprint arXiv:2404.17723 (2024)
12. Shang, W., Huang, X.: A survey of large language models on generative graph analytics: query, learning, and applications. arXiv preprint arXiv:2404.14809 (2024)
13. Edwards, C.: Hybrid context retrieval augmented generation pipeline: LLM-augmented knowledge graphs and vector database for accreditation reporting assistance. arXiv preprint arXiv:2405.15436 (2024)

14. Jing, Z., Su, Y., Han, Y., Yuan, B., Liu, C., Xu, H., Chen, K.: When large language models meet vector databases: a survey. arXiv preprint arXiv:2402.01763 (2024)
15. Abane, A., Battou, A., Merzouki, M.: An adaptable AI assistant for network management. In: IEEE/IFIP Network Operations and Management Symposium. Seoul, KR (2024)
16. Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Larson, J.: From local to global: a graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130 (2024)
17. Ni, P., Okhrati, R., Guan, S., Chang, V.: Knowledge graph and deep learning-based text-to-GraphQL model for intelligent medical consultation chatbot. Inf. Syst. Front. **26**(1), 137–156. Springer (2024)
18. Barrasa, J., Webber, J.: Building Knowledge Graphs. Inc, O'Reilly Media (2023)
19. Hodler, A.E., Needham, M.: Graph data science using Neo4j. In: Massive Graph Analytics, pp. 433–457. Chapman and Hall/CRC (2022)
20. Tamaˇsauskaite˙, G., Groth, P.: Defining a knowledge graph development process through a systematic review. ACM Trans. Softw. Eng. Methodol. **32**(1), 1–40. ACM (2023)
21. Galke, L., Saleh, A., Scherp, A.: Word embeddings for practical in- formation retrieval. In: Informatik 2017, pp. 2155–2167. Gesellschaft fu¨r Informatik (2017)
22. Kong, X., Wang, G., Nichol, A.: Conversational AI with Rasa: build, test, and deploy AI-powered, enterprise-grade virtual assistants and chatbots. Packt Publishing Ltd. (2021)
23. Palmonari, M., De Paoli, F.: Enabling data enrichment pipelines for AI-driven business products and services
24. Wita, R., Bubphachuen, K., Chawachat, J.: Content-based filtering recommendation in abstract search using neo4j. In: 2017 21st International Computer Science and Engineering Conference (ICSEC), pp. 1–5. IEEE (2017)
25. Mathew, A.B., Kumar, S.M.: An efficient index based query handling model for neo4j. IJCST **3**(2), 12–18. Citeseer (2014)
26. Bratanic, T.: Enhancing the accuracy of RAG applications with knowledge graphs. Medium (2024). https://medium.com/neo4j/enhancing-the-accuracy-of-rag-applications-with-knowledge-graphs-ad5e2ffab663