New Technologies for Spoken Dialogue Systems: LLMs, RAG and the GenAI Stack

Graham Wilcock

Abstract The paper describes some recent developments in Generative AI that can be applied in spoken dialogue systems, with a central theme of retrieval augmented generation from documents and knowledge graphs using open source large language models. The developments include vector indexing in Neo4j graph databases, the GenAI Stack provided by a collaboration of Docker, Neo4j, LangChain and Ollama, and the use of LLMs to generate Cypher queries that access knowledge graphs in graph databases. The example dialogue applications include Chat with PDFs, Chat with Wikipedia, and Chat with a knowledge graph.

1 Introduction: Conversational AI and Generative AI

In the past two years, spoken dialogue systems technology has seen a paradigm shift from Conversational AI to Generative AI. Previously, most dialogue systems were domain-specific, and Conversational AI provides effective tools for domain-specific natural language understanding (NLU) and for domain-specific natural language generation (NLG). For example Rasa conversational AI [4] provides transformer-based components for rapid training of domain-specific neural models for NLU [12] and for dialogue policy [20] from a small set of domain-specific examples.

By contrast, Generative AI provides pre-trained large language models (LLMs) that can perform NLU and NLG at a high level for any domain. Chat-tuned LLMs can also manage multi-turn dialogue interaction. This has led to rapid deployments of chatbots by developers with no previous experience of dialogue systems.

Wilcock and Jokinen [26] contrasted two approaches to dialogue in human-robot interaction. In one approach a robot used ChatGPT to access information from the LLM's knowledge base. In the other approach the robot used Rasa conversational

Graham Wilcock
CDM Interact, Helsinki, Finland and University of Helsinki, Finland.
e-mail: graham.wilcock@cdminteract.com

AI to access information from knowledge graphs in a Neo4j database. We described errors that occur in robot dialogues with the approaches, including false implication errors, ontological errors, theory of mind errors, and handling of speech recognition errors. We discussed the importance for human-robot interaction of grounding facts and actions in the real world, with particular concern for earning user trust.

In conclusion we noted that the approaches have complementary strengths and weaknesses. LLMs give more friendly responses but suffer from hallucinations that cannot earn trust. Knowledge graphs give more trustworthy replies with reliable data provenance but the responses are less natural. Research must focus on how to combine the strengths of the two approaches.

This paper describes recent developments that enable LLMs and knowledge graphs to be combined in dialogue systems. The intention is to generate friendly and more natural responses that will earn trust by avoiding hallucinations and by specifying the provenance of the information.

2 Recent Developments in Generative AI

The success of Generative AI means that LLMs now have high-level linguistic skills and can perform many dialogue tasks, including NLU and NLG. However, LLMs are trained on public data and know nothing about private domain data. An LLM can be fine-tuned with data for a particular private domain, but not for all domains.

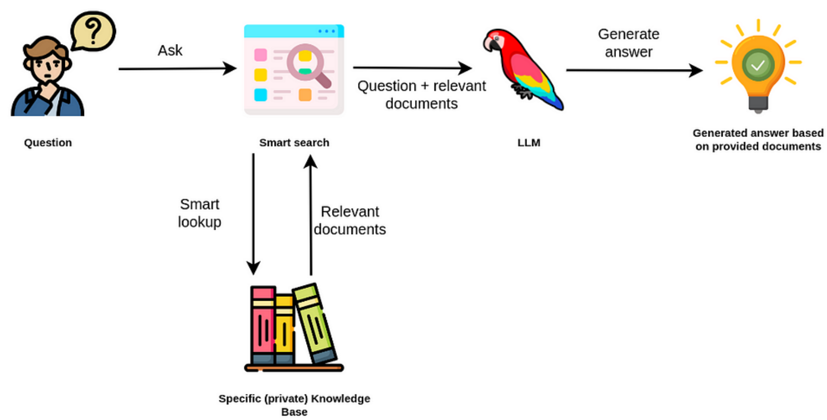

Fig. 1 Retrieval Augmented Generation from documents. Image by Tomaz Bratanic, from [8].

2.1 Vector-based Retrieval Augmented Generation

Figure 1 illustrates Retrieval Augmented Generation (RAG) from documents [17] as described by Bratanic [8]. RAG combines the advanced linguistic skills of LLMs with private domain-specific information. The task of the LLM is to perform NLG, producing natural-sounding and user-friendly responses, but only using information supplied from private documents, not from the LLM's own knowledge base. The aim is to increase accuracy, reduce the risk of hallucinations, and earn user trust.

As shown in Figure 1, when the user asks a question a subset of documents most relevant to the question are *retrieved* from a private domain-specific document set. The question and the subset of documents are passed to the LLM in an *augmented* prompt to *generate* an accurate answer that is relevant to the query. The retrieval process is done by vector-based semantic search [10] which compares an embedding of the question for semantic similarity with embeddings of domain documents.

For RAG applications, documents and their embeddings can be stored in vector databases [19] such as Chroma [13]. In a recent development, Neo4j databases now include vector indexing [15], enabling vector-based search on documents stored in Neo4j knowledge graphs. This allows both vector-based and graph-based semantic searches to be combined [2], as discussed in Section 4.

2.2 Neo4j Vector Indexing and GenAI Stack

Neo4j vector indexing allows Neo4j databases to play a central role in another recent development, GenAI Stack. This is a software stack for Generative AI applications,

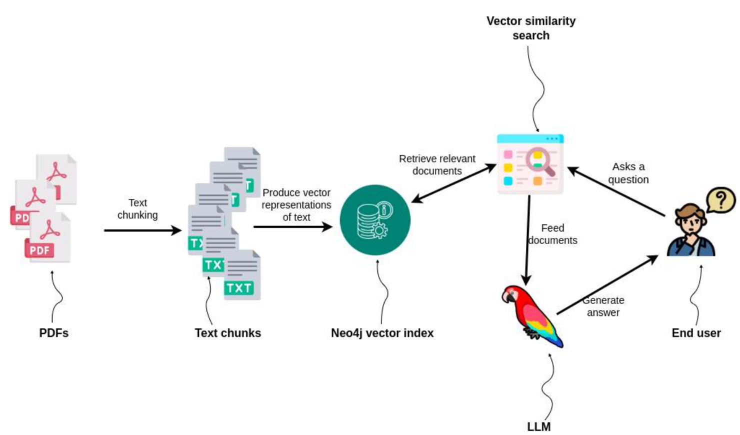

Fig. 2 Retrieval Augmented Generation in GenAI Stack. Image by Tomaz Bratanic, from [16].

created by a collaboration of Docker [14], Neo4j [16], LangChain and Ollama. Tools from Ollama manage local LLMs such as Llama2. Tools from LangChain combine LLMs with components such as document loaders and conversation memories. A Neo4j graph database provides storage for knowledge graphs and vector indexes. All the components run in Docker containers defined by Docker Compose.

Figure 2 shows a GenAI Stack application for retrieval augmented generation from PDF documents. The Neo4j database in the centre provides vector indexing for the text chunks. GenAI Stack defaults can all be changed. It uses Llama2 from Ollama as a default local LLM, but paid cloud LLM services can be used. By default LangChain tools are used with SentenceTransformers embeddings [18].

2.3 Graph-based Retrieval Augmented Generation

Vector-based RAG is a good approach for question answering (QA) from legacy documents like PDFs, but many organizations are moving enterprise information into knowledge graphs. An advantage of knowledge graphs over vector databases is that document text chunks, stored in the graph as nodes with their own properties such as provenance URLs, are linked to other nodes by meaningful relationships that can have their own properties. Nodes can be organized into meaningful taxonomies using semantic metadata [3], for example from WikiData. After finding nodes by vector search, applications retrieve additional information from the node properties and can traverse the graph to find further related nodes [16].

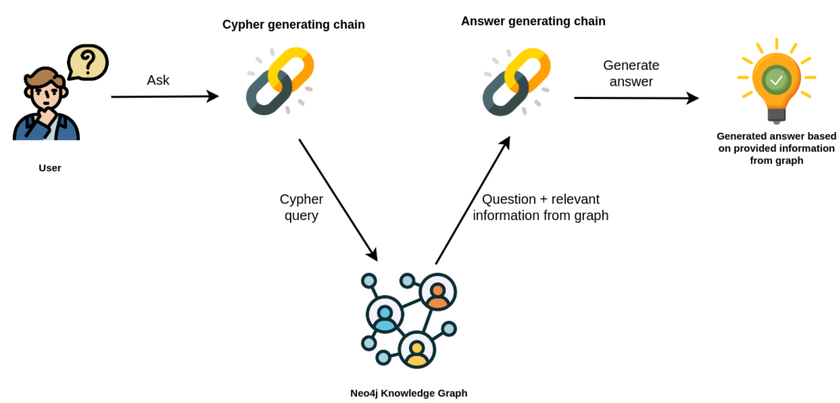

Fig. 3 Retrieval Augmented Generation from a knowledge graph using *neo4j_cypher* LangChain Template with two LLM chains. Image by Tomaz Bratanic, from [9].

Previously, dialogue systems that used knowledge graphs with conversational AI (such as [24]) had to implement custom actions [11, 22] to generate database query code to access the knowledge graphs. In another recent GenAI development, Bratanic [5, 7] showed that LLMs can generate Cypher code to query Neo4j graph databases, and implemented new components integrating Neo4j with LangChain. However, Cypher skills are still valuable for developers as code generation works better if the LLM is given accurate examples for few-shot learning.

Figure 3 shows the current state of the art for RAG from knowledge graphs. In the *neo4j_cypher* LangChain Template [9], two LLMs combine to perform QA from a Neo4j knowledge graph. The first LLM is used in the Cypher generating chain to generate a Cypher query from the user's natural language question. The generated query retrieves relevant information from the knowledge graph. The user's question and the relevant information are passed to the second LLM in the Answer generating chain, which generates the natural language answer to the user. The same LLM can be used in both chains, but it makes sense to use a code-tuned LLM for Cypher generation and a chat-tuned LLM for Answer generation. The application described in Section 3.2 uses CodeLlama for the first task and Llama2 for the second task.

3 Applying Generative AI in Dialogue Systems

This section discusses some examples of applying the developments in Section 2 to dialogues, starting with document-based and graph-based dialogue applications.

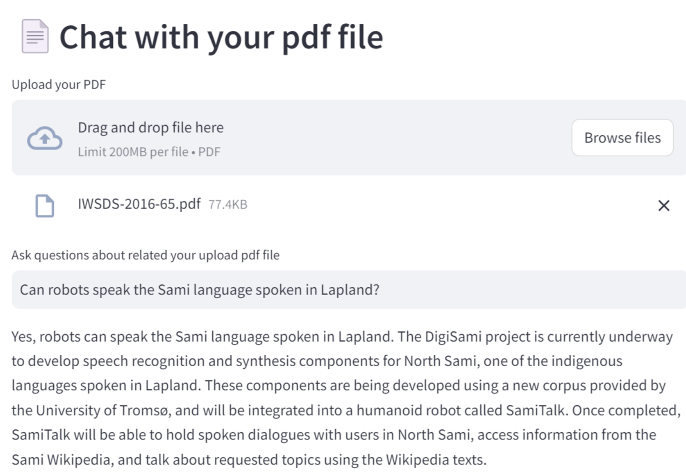

Fig. 4 PDFs are soon out-of-date. The DigiSami project was underway in 2016, but not today.

3.1 Document-based Dialogue Applications

The *Chat with your PDFs* application in Figure 2 [16] loads PDF files, stores text chunks and their embeddings in a Neo4j database, and uses vector-based RAG with Llama2 to generate natural language answers to user questions about the document contents. Note that the LLM generates confident but partly incorrect answers from old PDFs such as [27] where the information is out-of-date, as in Figure 4.

We added a *Chat with Wikipedia* application described by [7] to GenAI Stack. It downloads Wikipedia articles about a specified topic, stores text chunks in Neo4j, and uses vector-based RAG with Llama2 to answer user questions about the topic. This has an advantage over PDFs, as Wikipedia articles are frequently updated.

One of the aims of RAG was to reduce hallucinations by LLMs, and prompts tell LLMs to generate outputs using only the information given in the context and not to make up false facts. This can cause the LLM to answer as if the only true facts in the world are the facts given in the context.

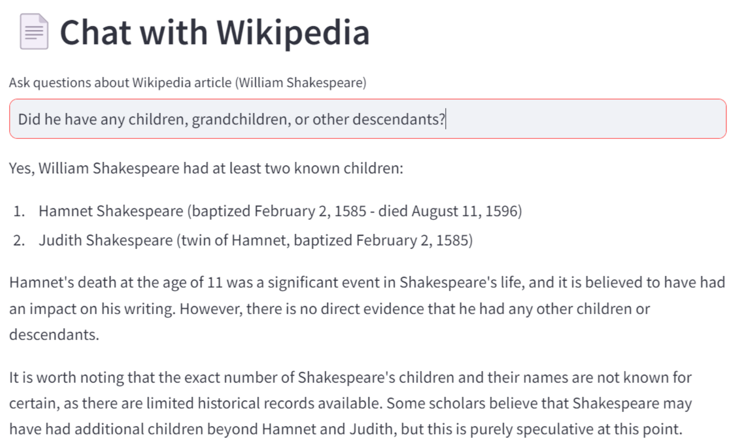

Fig. 5 When RAG goes wrong: William Shakespeare and Anne Hathaway had 3 children.

For example, when *Chat with Wikipedia* was given the topic *William Shakespeare* and the question *Did he have any children, grandchildren or other descendants?*, one RAG retrieval omitted his child Susanna from the context. The LLM (despite its knowledge of Shakespeare in its own knowledge base) insisted *There is no direct evidence that he had any other children*, as shown in Figure 5.

When asked *Who was Susanna Shakespeare?* a new set of chunks was found and the LLM answered *Susanna Shakespeare was the daughter of William Shakespeare and his wife Anne Hathaway*, adding: *Susanna is one of three children known to have been born to Shakespeare and his wife, with her two younger siblings being Hamnet and Judith.* This answer is fine, but the previous answer was very misleading.

Research on improving RAG techniques is currently very active. A promising approach [6] is to organize the indexing of text chunks into some kind of hierarchy, and in Neo4j such hierarchies can be represented directly by the graph.

3.2 Graph-based Dialogue Applications

The CityTalk robot dialogue system was first demonstrated at IWSDS 2018 [21]. More recent versions [24, 25] use Furhat robots [1] with Rasa conversational AI [4] and knowledge graphs in Neo4j databases [3]. An advantage of graph databases is that ontological and other semantic metadata, such as taxonomies and geographical data, can be added to the graphs. The metadata enables more intelligent responses to be generated [22], which are also more cooperative [25]. For example, when a user asked for restaurants serving European cuisine and there were Italian restaurants in the desired location, but CityTalk did not know that Italian cuisine is a subclass of European cuisine, the robot gave a negative response. This ontological error [26] was solved by importing a taxonomy of cuisines from WikiData [22].

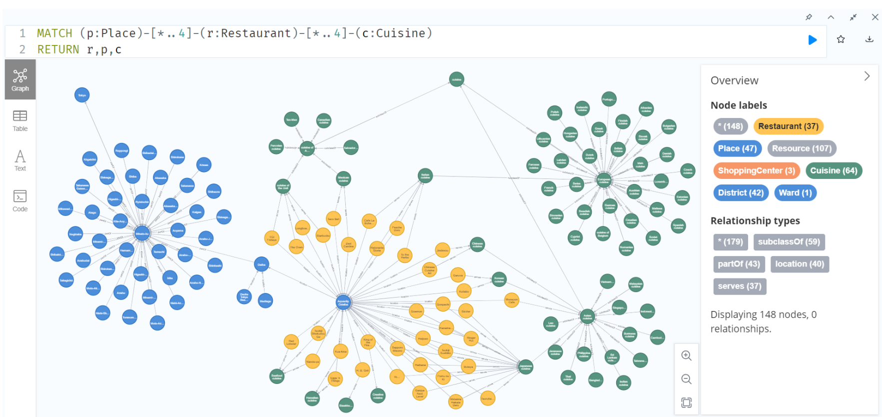

Fig. 6 Some restaurants and a taxonomy of cuisines in a CityTalk knowledge graph in Neo4j.

Figure 6 shows part of a CityTalk graph with 37 restaurants (yellow) linked to their cuisines in the taxonomy (green) and other districts (blue) in the same ward of Tokyo. A video [23] shows the robot giving a positive response to the query about European cuisine by checking the taxonomy and finding the Italian restaurants.

A new GenAI version of CityTalk uses LLMs with LangChain and employs the *neo4j_cypher* approach (Figure 3). Cypher queries are generated by CodeLlama in the first LLM chain, and natural language answers are generated by Llama2 in the second LLM chain. The responses are more friendly, as shown in Figure 7.

Fig. 7 A new version of CityTalk uses CodeLlama and Llama2 for graph-based semantic search.

4 Conclusion: Vector-based and Graph-based Semantic Search

Nodes and relationships in knowledge graphs are explicit symbolic representations of entities in the real world and the relationships between them, in contrast to the implicit subsymbolic vectorized represention of words, sentences and texts used in LLMs [2]. Using knowledge graphs with LLMs provides a means of *grounding* the sentences in the LLMs to the real world, which is especially important for embodied, situated robots that perform actions in the real world [26].

However, knowledge graphs are good for objective, static, well-structured facts, such as the address, postcode and telephone number of a restaurant. They are less good at subjective, transient and unstructured information, such as whether there is a wonderful view of a famous bridge from the window seats at sunset, or whether the staff are particularly helpful. This kind of information is found in texts of reviews and social media comments by customers after visiting the restaurant.

The training corpora for LLMs include subjective reviews, but not what people have said recently about a new restaurant in a local shopping mall. Vector-based searching of unstructured texts is needed. If a user is looking for a restaurant with a beautiful view and a friendly ambiance that serves European cuisine, the system needs to compare an embedding of this query to embeddings of reviews and user comments about restaurants in the desired area to find the most relevant candidates. These candidates can then be checked with the knowledge graph. If the vector search finds a good candidate with the desired view and ambience but with Italian cuisine, the taxonomy in the graph will help by showing that Italian cuisine is a subclass of European cuisine, so the candidate is a good match.

Ongoing work will develop ways to combine vector-based and graph-based semantic search to improve dialogue systems.

References

1. Al Moubayed, S., Beskow, J., Skantze, G., Granström, B.: Furhat: A back-projected human-like robot head for multiparty human-machine interaction. In: A. Esposito, A.M. Esposito, A. Vinciarelli, R. Hoffmann, V.C. Müller (eds.) Cognitive Behavioural Systems, pp. 114–130. Springer (2012)
2. Barrasa, J.: Going Meta Episode 21: Vector-based semantic search and graph-based semantic search. https://www.youtube.com/watch?v=bRD09ndyJNs (2023)
3. Barrasa, J., Webber, J.: Building Knowledge Graphs: A Practitioner's Guide. O'Reilly Media (2023)
4. Bocklisch, T., Faulkner, J., Pawlowski, N., Nichol, A.: Rasa: Open source language understanding and dialogue management (2017). ArXiv:1712.05181
5. Bratanic, T.: Generating Cypher queries with ChatGPT 4 on any graph schema. https://neo4j.com/developer-blog/generating-cypher-queries-with-chatgpt-4-on-any-graph-schema/ (2023)
6. Bratanic, T.: Implementing advanced RAG strategies with Neo4j. https://blog.langchain.dev/implementing-advanced-retrieval-rag-strategies-with-neo4j/ (2023)
7. Bratanic, T.: Integrating Neo4j into the LangChain ecosystem. https://towardsdatascience.com/integrating-neo4j-into-the-langchain-ecosystem-df0e988344d2 (2023)
8. Bratanic, T.: Knowledge Graphs & LLMs: Fine-Tuning vs. Retrieval-Augmented Generation. https://neo4j.com/developer-blog/fine-tuning-retrieval-augmented-generation/ (2023)
9. Bratanic, T.: neo4j_cypher. https://github.com/langchain-ai/langchain/tree/master/templates/neo4j-cypher (2023)
10. Briggs, J.: NLP for Semantic Search. https://www.pinecone.io/learn/series/nlp/ (2023)
11. Bunk, T.: Integrating Rasa with knowledge bases. https://rasa.com/blog/integrating-rasa-with-knowledge-bases/ (2019)
12. Bunk, T., Varshneya, D., Vlasov, V., Nichol, A.: DIET: Lightweight language understanding for dialogue systems (2020). ArXiv:2004.09936
13. Chroma: Chroma - the open-source embedding database. https://github.com/chroma-core/chroma/ (2023)
14. Docker Team: Introducing a New GenAI Stack: Streamlined AI/ML Integration Made Easy. https://www.docker.com/blog/introducing-a-new-genai-stack/ (2023)
15. Hasbe, S.: Neo4j's Vector Search: Unlocking Deeper Insights for AI-Powered Applications. https://neo4j.com/blog/vector-search-deeper-insights/ (2023)
16. Hunger, M., Bratanic, T., Hane, O.: GenAI Stack Walkthrough: Behind the Scenes with Neo4j, LangChain and Ollama in Docker. https://neo4j.com/developer-blog/genai-app-how-to-build/ (2023)
17. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), pp. 9459–9474. Vancouver, Canada (2020)
18. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: K. Inui, J. Jiang, V. Ng, X. Wan (eds.) EMNLP/IJCNLP (1), pp. 3980–3990. Association for Computational Linguistics (2019)
19. Schwaber-Cohen, R.: What is a Vector Database & How Does it Work? Use Cases + Examples. https://www.pinecone.io/learn/vector-database/ (2023)
20. Vlasov, V., Mosig, J.E.M., Nichol, A.: Dialogue transformers (2019). ArXiv:1910.00486
21. Wilcock, G.: CityTalk: Robots that talk to tourists and can switch domains during the dialogue. In: L.F. D'Haro, R.E. Banchs, H. Li (eds.) 9th International Workshop on Spoken Dialogue Systems Technology, pp. 411–417. Springer (2019)

22. Wilcock, G.: Generating more intelligent responses and explanations with conversational AI and knowledge graphs. In: Annual Conference of the Japanese Society for Artificial Intelligence (JSAI 2022). Kyoto, Japan (2022)
23. Wilcock, G.: Robot responses improved by knowledge graph metadata. https://www.youtube.com/watch?v=Ql5nbap5cRs (2022)
24. Wilcock, G., Jokinen, K.: Conversational AI and knowledge graphs for social robot interaction. In: ACM/IEEE International Conference on Human-Robot Interaction (HRI 2022), pp. 1090–1094. Sapporo, Japan (2022)
25. Wilcock, G., Jokinen, K.: Cooperative and uncooperative behaviour in task-oriented dialogues with social robots. In: 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2022), pp. 763–768. Napoli, Italy (2022)
26. Wilcock, G., Jokinen, K.: To Err Is Robotic; to Earn Trust, Divine: Comparing ChatGPT and Knowledge Graphs for HRI. In: 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2023), pp. 1396–1401. Busan, Korea (2023)
27. Wilcock, G., Laxström, N., Leinonen, J., Smit, P., Kurimo, M., Jokinen, K.: Towards SamiTalk: a Sami-speaking Robot linked to Sami Wikipedia. In: K. Jokinen, G. Wilcock (eds.) Dialogues with Social Robots: Enablements, Analyses, and Evaluation, pp. 343–351. Springer (2017)