MSDS 499 Project: Kowledge-Graph-Based Linear Programming Chatbot

Yue Yang

Abstract

This project explores the integration of Retrieval-Augmented Generation (RAG) with a structured knowledge graph to address the challenge of hallucinations in Large Language Models (LLMs), particularly within the domain of linear programming. Hallucinations, or the generation of inaccurate information by LLMs, pose significant obstacles in professional and educational settings. By leveraging a knowledge graph constructed from linear programming textbooks and employing the RAG model, this study aims to enhance the accuracy and relevance of responses generated by a domain-specific chatbot. The knowledge graph organizes textbook content into a structured format that the RAG model can efficiently query. This methodology not only improves the chatbot's ability to provide precise answers but also enriches its responses with contextually relevant information, making it a valuable tool for students and professionals alike. The project's approach demonstrates the potential of combining knowledge graphs with RAG to mitigate the limitations of LLMs, offering a possible direction for future research in enhancing conversational AI.

Introduction

Linear programming is a pivotal area within mathematical optimization, presenting significant challenges in comprehension and application, particularly for students and professionals grappling with complex problems. Traditional Large Language Models often generate responses that, despite being fluent, may lack in accuracy or relevance, a phenomenon known as hallucination. To mitigate this, integrating a knowledge graph-based approach utilizing the Retrieval-Augmented Generation (RAG) model can significantly enhance the precision and clarity of answers provided by a domain-specific chatbot. This integration leverages a knowledge graph constructed from linear programming textbooks, encompassing both integer and linear programming concepts. Such a graph not only encapsulates the structure of these textbooks but also elucidates the relationships between various concepts, thereby fostering a comprehensive understanding of the domain. The RAG model employs this structured knowledge to inform its responses, ensuring that the generated answers are grounded in the authoritative content of the textbooks. This methodology aims to deliver a chatbot that not only accurately responds to

queries but also provides explanations that reflect a deep understanding of linear programming, rendering it an invaluable tool for learners and professionals alike.

Literature Review

The integration of contextual knowledge has been pivotal in reducing errors and enhancing the reliability of LLMs. In this regard, Lewis et al.'s (2020) study on "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" exemplifies the significant strides made in language generation through the innovative use of RAG models. By amalgamating pre-trained parametric models with expansive non-parametric memory, this work sets new precedents in the efficacy of open-domain question answering tasks, demonstrating the critical role of retrieval mechanisms in augmenting language models with external knowledge sources.

Further exploration by Izacard and Grave (2021) in "Benchmarking Large Language Models in Retrieval-Augmented Generation" evaluates the implications of integrating RAG with LLMs. Their findings reveal that while LLMs possess inherent noise robustness, they encounter hurdles in areas such as negative rejection, information synthesis, and counterfactual reasoning. These insights highlight the instrumental role of RAG models in overcoming the intrinsic limitations of LLMs, particularly concerning the issue of hallucinations, thereby underscoring the transformative potential of RAG models in enhancing the fidelity and robustness of LLM outputs.

Additionally, the synergistic integration of conversational agents with knowledge graphs presents a promising avenue for elevating chatbot capabilities, as illustrated by Moon et al.'s (2020) discussion in "Integrating Conversational Agents and Knowledge Graphs Within the Scholarly Domain." This approach emphasizes the utility of knowledge graphs in furnishing chatbots with a structured, scalable representation of information, thereby significantly augmenting their response accuracy and logical coherence. Such advancements are further corroborated by the application of knowledge graphs in the domain of linear programming chatbots, where the structured representation of textbook data dramatically improves the chatbot's ability to deliver accurate and logically sound responses.

In summary, the literature supports the notion that RAG models, when augmented with structured knowledge from knowledge graphs, can effectively mitigate the issue of hallucination in LLMs and enhance the accuracy and relevance of chatbot responses. This foundation sets the stage for the development of a linear programming chatbot that leverages a knowledge graph and RAG to provide precise, contextually relevant solutions to complex queries, indicates a possible application of chatbots for educational and professional purposes in the domain of mathematical programming.

Methodology

Knowledge Graph Generation

The generation of the knowledge graph begins with the transformation of textbook data into structured nodes. This is accomplished using the HierarchicalNodeParser from the LlamaIndex library, which parses PDF textbooks into nodes that represent the hierarchical structure of the content. With a total of 8683 nodes, these nodes can be leaf nodes, which do not have children, or root nodes, which do. The relationships between nodes are identified and ingested as links within the EdgeDB schema, allowing for a rich representation of the domain's knowledge.

Schema and Data Ingestion

The EdgeDB schema, detailed in the appendix, is designed to accommodate the structured data from the textbooks. The schema includes TextNode entities, which are populated with the structured data, and metadata extracted using custom extractors such as TitleExtractor, QuestionsAnsweredExtractor, EntityExtractor, SummaryExtractor, and KeywordExtractor. This metadata enrichment is crucial for augmenting the chatbot's search and retrieval capabilities, aligning user queries with the most relevant information from the knowledge graph.

Retrieval Mechanism

The retrieval method employed is the Auto Merging Retriever, which examines a set of leaf nodes and recursively "merges" subsets that reference a parent node beyond a certain threshold.

This method consolidates smaller contexts into a larger, more coherent context that aids in the synthesis of information.

Embeddings and Indexing

To further enhance the chatbot's ability to retrieve information efficiently, the `fetch_and_embed_nodes` function generates embeddings for the nodes. These embeddings are indexed using the `NearestNeighbors` algorithm from `scikit-learn`, enabling the chatbot to quickly identify and retrieve the most pertinent information in response to user queries.

User Interface Development

The user interface of the chatbot, developed using Streamlit, incorporates the Perplexity API for response generation. It utilizes the structured knowledge graph and the embeddings index to provide responses that are not only accurate but also highly relevant to the user's context. This interface is a critical component of the chatbot, facilitating user interaction and enhancing the overall user experience.

In summary, the methodology involves a systematic approach to constructing a knowledge graph using the LlamaIndex framework and EdgeDB, enriching the data with metadata, and employing advanced retrieval and indexing techniques to power the RAG functionality of the chatbot. This comprehensive process ensures that the chatbot is equipped to handle the intricacies of linear programming queries with high accuracy and relevance.

Results

Benchmarking and Performance Evaluation

A benchmark comparison between our RAG-enhanced LLM and the original LLM, codellama-70b-instruct, revealed that the RAG-enhanced model produced longer and more comprehensive answers. This was particularly evident in response to complex queries such as, "I am managing a software development project. How can I identify activities on the critical path?" The detailed responses generated by the RAG-enhanced LLM, as documented in the appendix, highlight the

model's superior ability to synthesize and present information in a coherent and informative manner.

Expositions

The project also explored the potential of enriching the knowledge graph with metadata to improve the chatbot's search and retrieval functions. Utilizing functions provided by LlamaIndex, powered by OpenAI's model, for metadata retrieval—such as generating summaries, possible questions, keywords, and entity predictions—showed promising results. However, due to computational constraints and the cost associated with GPT model usage, a comprehensive application of these functions across all 8000+ nodes was not feasible within the project's scope.

The choice of LLM, codellama-70b-instruct, was primarily influenced by cost considerations. While this model provided a cost-effective solution, the exploration of alternative models such as langchain or more advanced GPT models could potentially yield even better results, given sufficient resources. The project's findings suggest that the selection of the underlying LLM model plays a crucial role in determining the overall effectiveness of the RAG-enhanced chatbot.

Ongoing management and refinement of the knowledge graph are identified as critical factors for further enhancing the chatbot's capabilities. The addition of more data to the knowledge graph is anticipated to improve the chatbot's performance, making it an even more powerful tool for users seeking assistance with linear programming problems.

Conclusion

In conclusion, the project has demonstrated the viability and effectiveness of integrating a structured knowledge graph with a RAG architecture to enhance the performance of LLMs. While the current results are positive, future work will focus on expanding the knowledge graph, exploring alternative LLM models, and optimizing metadata enrichment processes to further improve the chatbot's accuracy and relevance in responding to domain-specific queries. Future directions include expanding the knowledge graph with more sources to enrich the chatbot's

information base, potentially improving its ability to address a broader range of mathematical programming questions. Additionally, evaluating different LLM models could identify options with superior performance in areas like noise tolerance and information integration, which would be beneficial for the chatbot's functionality.

References

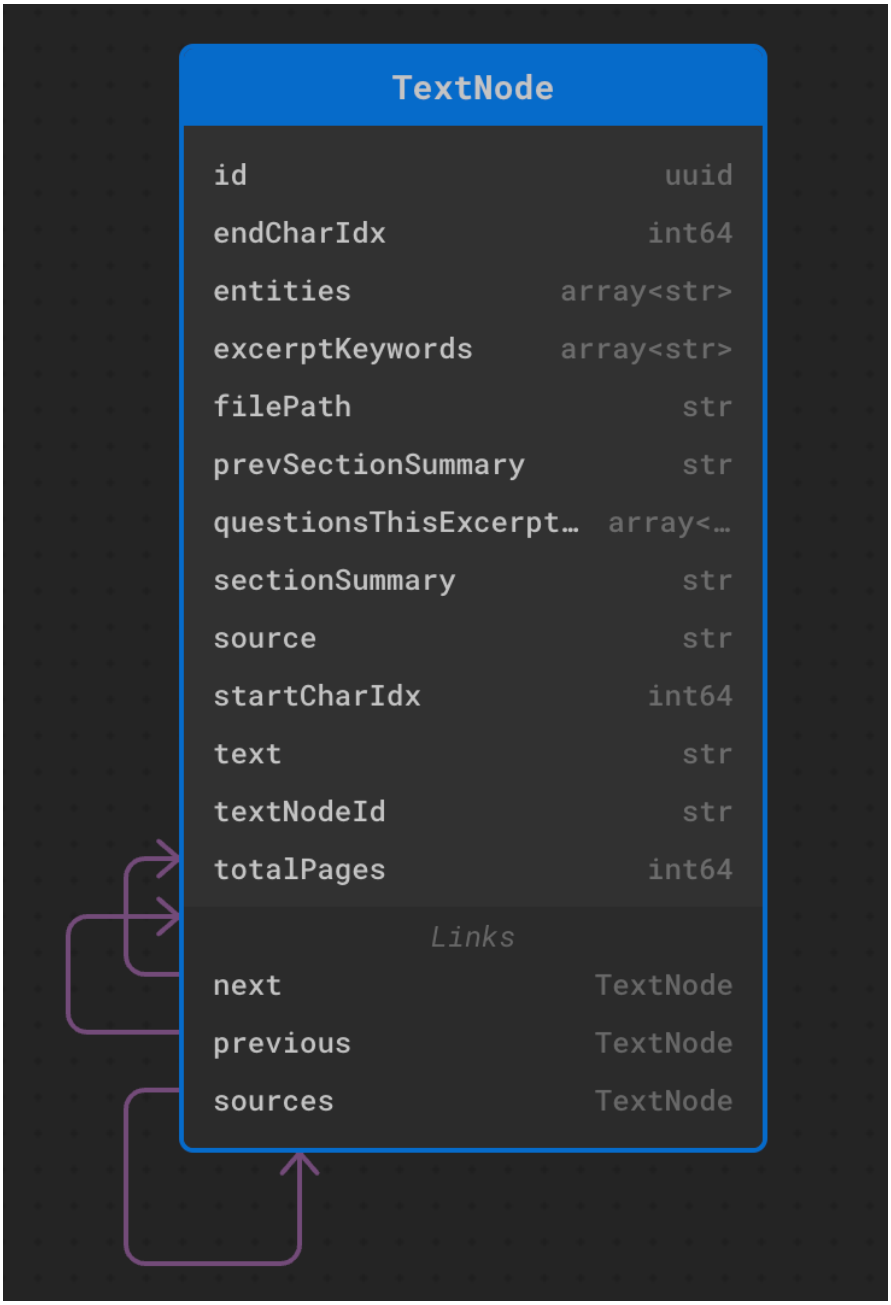
Lewis, Patrick, Ethan Perez, Aleksandar P. Stojanovic, Arun Babu, Yuxiang Wu, and Douwe Kiela. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." arXiv preprint arXiv:2005.11401 (2020). https://arxiv.org/abs/2005.11401.

Anonymous. "Benchmarking Large Language Models in Retrieval-Augmented Generation." arXiv preprint arXiv:2309.01431 (2023). https://arxiv.org/abs/2309.01431.

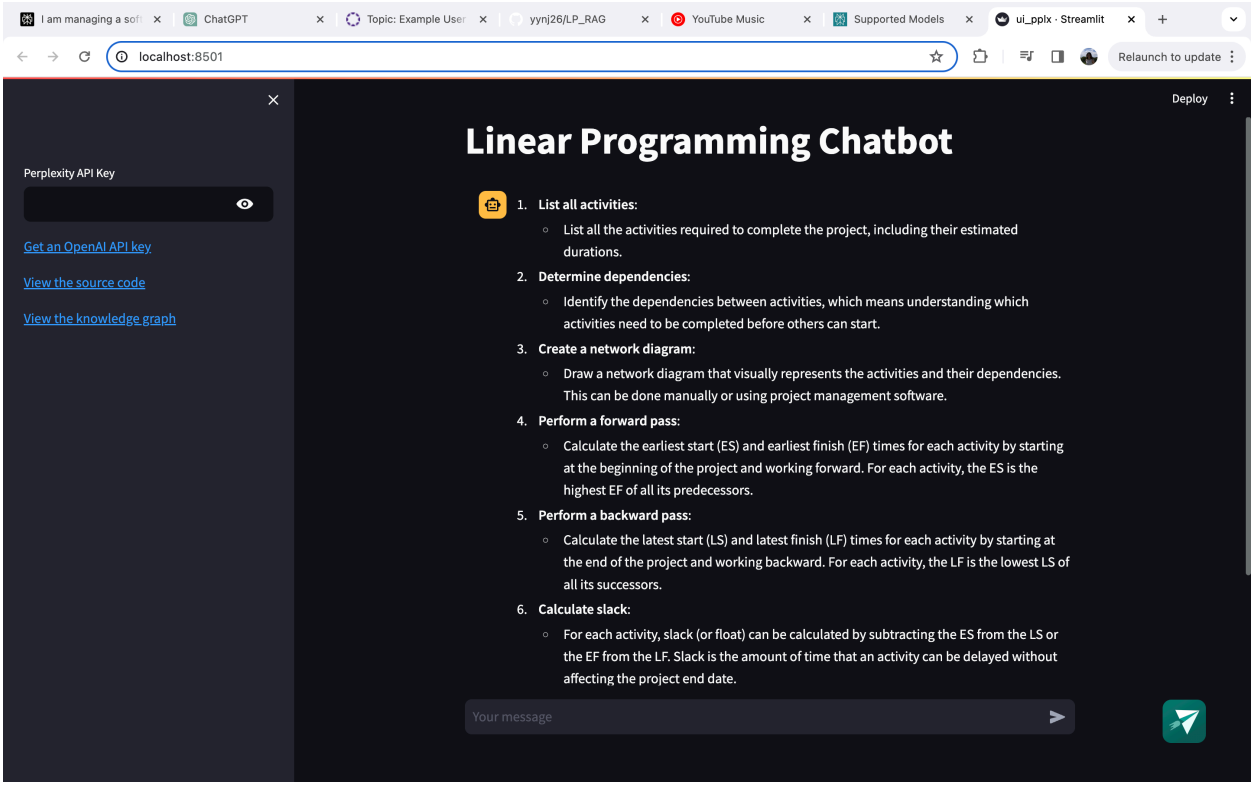
Dessì, Danilo, Harald Sack, and Maria-Esther Vidal. "Integrating Conversational Agents and Knowledge Graphs Within the Scholarly Domain." Semantic Scholar. Accessed March 17, 2024. https://www.semanticscholar.org/paper/47effa64fdf759a9adfb98c10aa56747fdff7a9e.

Llama Index Documentation. "Parse Chunk Hierarchy from Text & Load into Storage." Accessed March 17, 2024.https://docs.llamaindex.ai/en/stable/examples/retrievers/auto_merging_retriever.html#parse-chunk-hierarchy-from-text-load-into-storage.

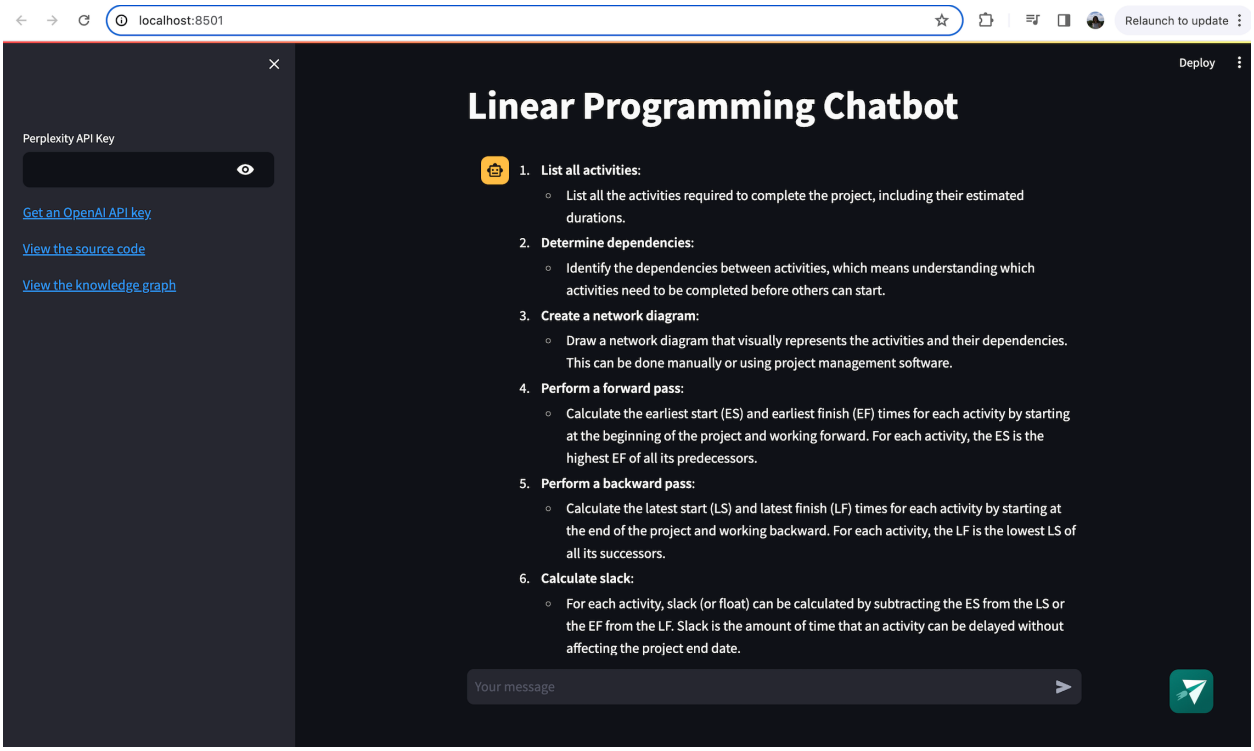
Appendix



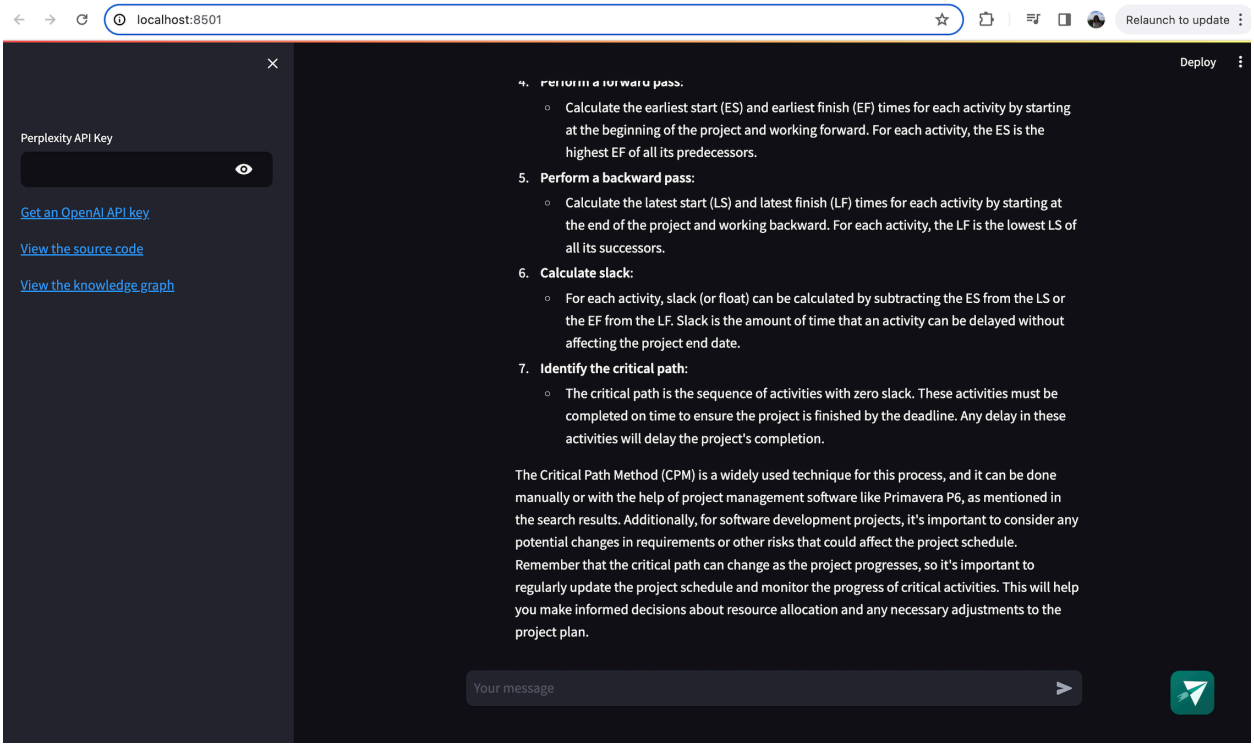
Appendix 1. EdgeDB schema built on Llamaindex node



Appendix 2. LLM answer of "I am managing a software development project. How can I identify activities on the critical path?"



Appendix 3. Knowledge-based LLM answer of "I am managing a software development project. How can I identify activities on the critical path?" part 1.



Appendix 4. Knowledge-based LLM answer of "I am managing a software development project. How can I identify activities on the critical path?" part 2.

Appendix 5. The code is posted at https://github.com/yynj26/LP_RAG.