11-667 Course Project: Intelli-Research Assistant : Retrieval Augmented Generation system leveraging Knowledge Graphs for Downstream Natural Language Tasks

Sahithya Senthilkumar **Ipsita Praharaj** **Jiuyuan Xie**

Abstract

Confronted with the exponential growth of scholarly publications, our groundbreaking system introduces a significant enhancement in the field of academic research navigation. By integrating a Neo4j knowledge graph with advanced large language models (LLMs) such as PaLM2, GPT-3.5, Code LlaMa, and LlaMa2, our approach ushers in a new era of search precision. Our system adeptly converts text-based inquiries into Cypher queries, enabling the exact curation and synthesis of scholarly content. It excels at distilling intricate research into accessible, well-structured synopses while taking into account critical metadata attributes like authorship, abstracts, citation metrics, and subject matter. With a focus on minimizing the search space for embeddings, our approach enhances the precision and efficiency of LLM-based retrieval systems with replacement of a vector DB with a Graph DB. "It offers backward data compatibility with many existing e-commerce and social media platforms. Additionally, it confronts the cold-start challenges associated with LLMs and can adjust easily to new user preferences and contextual information in the graph edges."Our proposed system is set to become an essential tool for academics, simplifying the digestion of vast amounts of scholarly data and catalyzing the discovery of new interdisciplinary research opportunities within large-scale, data-rich environments.

1 Introduction

In the digital domain of academia, where the volume of scholarly material is surging, there is an imperative need for retrieval and summarization systems capable of efficiently guiding researchers through the ever-expanding repository of knowledge(Lewis et al., 2020). Traditional recommendation algorithms falter, notably with the cold start problem, where insufficient data on new entities hinders the ability to provide reliable recommendations. These systems also struggle with data scarcity and a generic approach to user preferences, inadequacies magnified by the vast and varied corpus of academic research. Even sophisticated Large Language Models (LLMs) like GPT-3.5, Code Llama, and Llama, despite their broad generative abilities, often miss the mark in capturing the detailed personalization required for intricate user-item interaction dynamics(Tamkin et al., 2021).

Our research introduces a cutting-edge retrieval-augmented generation (RAG)(Es et al., 2023)system that capitalizes on the robustness of Neo4j's graph database combined with the flexible computational power of LLMs. RAG is an advanced system that aids in content creation. It retrieves relevant information to support the creation of coherent and contextually accurate texts. This new addition of Knowledge graphs is not just an incremental improvement; it is a transformative leap forward, providing a customized and dynamic method for discovering and managing academic literature. Our Knowledge Graph (KG) + RAG system overcomes the limitations of conventional models and is set to redefine the way researchers access and interact with scholarly content.

Central to our approach is the use of a Neo4j graph database, which sidesteps the need for embedding conversion when new data is added—streamlining the integration of the latest research into the system.(Wang et al., 2020) This is particularly beneficial for our dataset of research papers, as it ensures that users are provided with summaries that reflect the most current findings and insights without the computational delay of text - to - embedding [1].Updating weights in a graph database, in response to user interactions and preferences, is more efficient than traditional re-embedding new additions on the graph Db in future. This fast updation leads to a more responsive and refined sum-

[1] https://neo4j.com/blog/knowledge-graph-vs-vectordb-for-retrieval-augmented-generation/

marization using cumulative user-weighted trends system, (Lu et al., 2023).

2 Related Work

The interplay of Knowledge Graphs (KGs) and Large Language Models (LLMs) in the field of academic literature management is at the forefront of current research. The study "Unifying Large Language Models and Knowledge Graphs: A Roadmap"(Pan et al., 2023) provides valuable insights into the potential of this integration, emphasizing the complementary strengths of KGs and LLMs in information retrieval and processing.

Furthermore, the application of LLMs in enhancing KGs is explored in "LLM-assisted Knowledge Graph Engineering: Experiments with ChatGPT" (Meyer et al., 2023), which demonstrates the efficacy of LLMs in refining and expanding KGs. This highlights the evolving landscape of KGs, which our system aims to capitalize on for the summarization of academic literature.

The paper "SKILL: Structured Knowledge Infusion for Large Language Models"(Moiseev et al., 2022a) further delves into the integration of structured knowledge into LLMs. This approach aligns with our methodology, where we leverage the structured nature of KGs to enhance the capabilities of RAG systems in summarizing complex academic content.

Additionally, Google's research on KGs and LLMs (Moiseev et al., 2022b) provides a comprehensive overview of the advancements in this domain, offering context and validation to our approach of combining Neo4j's graph database with LLMs for efficient literature management.

In line with our focus on summarization rather than plain retrieval, (He et al., 2017) and (Guo et al., 2020) offer foundational insights into the challenges, which we adapt to the to getting exact context of academic summarization via querying first approach. Our approach, while distinct, draws upon these principles to enhance the accuracy and relevance of our summarization because we have the restricted.

(Blumenfeld, 2021) specifically demonstrates the utility of Neo4j as a robust KG system, which we employ in our framework. The agility and efficiency of Neo4j in managing and updating data make it an ideal choice for our RAG system.

The potential of LLMs as ranking models in retrieval systems, as explored in (Hou et al., 2023), (Ji et al., 2023), and (Wang et al., 2023), further underscores the versatility of LLMs in handling complex tasks. While our system focuses on summarization, the underlying principles of these works inform our approach, especially in terms of data handling and user interaction.

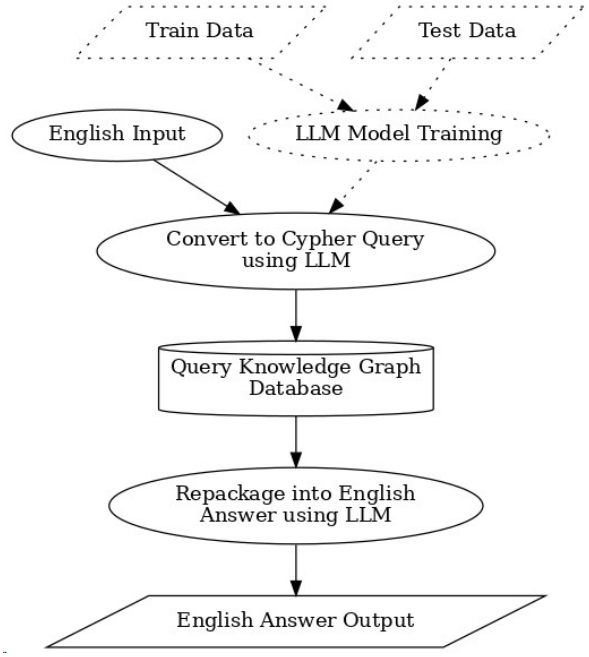

Figure 1: Our Proposed Pipeline

The above exploration sets the stage for our detailed exploration of the integration of KGs and RAG systems in the subsequent sections of the paper, focusing on their novel application in the summarization of academic literature.

3 Methodology and Pipeline

Our system's architecture is designed to harness the synergy between custom-created knowledge graphs (KGs) and large language models (LLMs) to improve the retrieval and summarization of academic papers. This section outlines the methodology and pipeline, each step buttressed with academic research to underscore its validity Our proposed pipeline can be seen in Fig.1.

3.1 Custom-Created Knowledge Graphs:

We have developed a knowledge graph using Semantic Scholar APIs and a subset of the Microsoft Academic Graph. The purpose of employing a KG is to leverage its structured data for efficient query processing, which is crucial for handling the diverse and complex data associated with academic papers. KGs are optimized for efficient data re-

trieval, which is fundamental in academic research where rapid access to information is crucial

3.2 Cypher + Retrieval Augmented Generation (RAG):

The integration of Cypher query language with a RAG system enables the transformation of human text queries into their Cypher equivalents. This is done using the advanced capabilities of LLMs, facilitating precise knowledge extraction and speedy retrieval. This step is pivotal as it combines the interpretive power of LLMs with the structured querying capabilities of Cypher, leading to enhanced retrieval accuracy

3.3 Use of Various Language Models:

Our framework utilizes a variety of LLMs—each selected for its specific strengths. Llama is known for its general applicability, although it is not open source. CodeLlama, on the other hand, is tailored for interpreting and generating code-related queries. By employing multiple models, we ensure a wide coverage of data handling capabilities, from general inquiries to specialized requests. The choice of these models is grounded in their proven efficacy in handling varied datasets

3.4 Fine-Tuning with Qlora on Llama, CodeLlama, PALM, GPT 3.5:

Fine-tuning is an essential step to adapt LLMs to our specific domain. We apply LoRA, QLoRA, an advanced fine-tuning method, to both Llama and CodeLlama, enhancing their ability to understand and process complex queries and contexts. PaLM2 is fine-tuned using Google Vertex AI and GPT-3.5 is fine-tuned using OpenAI's API, which provides a "black box" approach to fine-tuning that has been shown to improve model performance significantly

In summary, this pipeline meticulously combines the structured data handling of KGs with the interpretive and generative prowess of LLMs, culminating in a system adept at summarizing complex academic literature. The selection of Llama, CodeLlama, and GPT-3.5 is strategic, aimed at providing comprehensive coverage of the diverse needs within academic research and ensuring that the system remains at the forefront of technological advancement.

4 Knowledge Graph

4.1 Knowledge Graph Creation

The graph dataset is a cornerstone of the Intelli-research Assistant, a state-of-the-art Retrieval Augmented Generation system. This system is designed to enhance search and retrieval efficiency through a robust knowledge graph database. Our dataset creation pipeline is depicted in Fig.2. Our approach to constructing this database begins by sourcing a subset of the Microsoft Academic Graph which provides foundational details such as paper titles, authors, citations, affiliations, and conference names.
However, a significant challenge we encountered was the absence of abstract information within the dataset, which is vital for advanced natural language processing tasks.

4.2 Node Representation and Relationships

The knowledge graph of the Intelli-research Assistant consists of nodes representing academic papers, conferences, authors, affiliations, and domains, each embodying a distinct aspect of scholarly work. These nodes Fig. 3 are interlinked by relationships Fig. 4 that illustrate author affiliations, paper authorships, citation networks, domain classifications, and publication venues. This interconnected structure provides a comprehensive map of the academic landscape, delineating the intricate web of scholarly activity and collaboration.

4.3 Neo4j and Implementation Details

Neo4j is particularly advantageous for our Intelli-research Assistant's knowledge graph due to its efficient handling of complex queries and relationships. The relationships within an academic graph are not merely linear but multidimensional, reflecting the multifaceted connections of real-world academia. Neo4j excels in storing these relationships and facilitating fast and efficient queries, which is essential for the research assistant's functionality.

4.4 Abstract Retrieval and Embeddings Enhancement

To augment our graph with essential scholarly content, we integrated a sophisticated abstract retrieval process leveraging Semantic Scholar's API. This process is twofold: initially, we query the API to match paper titles with their unique identifiers within the Semantic Scholar database; subsequently, we use these identifiers to retrieve compre-

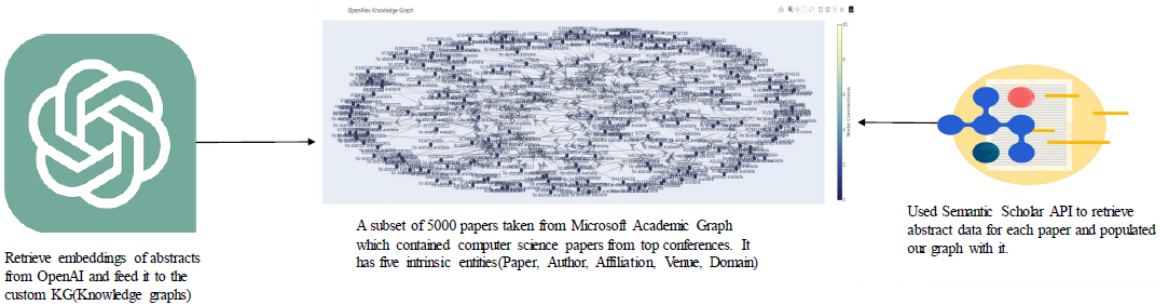

Figure 2: Dataset Creation Pipeline

Relationships	Size
Author_in_affiliation	14488
author_write_paper	28192
paper_cite_paper	17166
paper_in_domain	41274
paper_in_venue	10094

Figure 3: Nodes of Custom Graph

Nodes	Count
Papers	5047
Conferences	20
Authors	8680
Affiliation	692
Domain	1923

Figure 4: Relationship between Nodes of Custom Graph

```
def create_node(self, entity_id, name, entity_type):
    with self.driver.session() as session:
        session.write_transaction(self._create_node, entity_id, name, entity_type)

@staticmethod
def _create_node(tx, entity_id, name, entity_type):
    # Using entity_type as the label for the node
    query = (
        f"CREATE (:{entity_type} {{id: $entity_id, name: $name}})"
    )
    tx.run(query, entity_id=entity_id, name=name)

def create_relationship(self, entity1_id, relation, entity2_id):
    with self.driver.session() as session:
        session.write_transaction(self._create_relationship, entity1_id, relation, entity2_id)

@staticmethod
def _create_relationship(tx, entity1_id, relation, entity2_id):
    query = (
        f"MATCH (a {{id: '{entity1_id}'}}), (b {{id: '{entity2_id}'}}) "
        f"CREATE (a)-[:{relation}]->(b)"
    )
    tx.run(query)
```

Figure 5: Research papers Graph on Neo4j

hensive abstracts for each paper. This integration of abstracts is pivotal, as it provides the depth of content necessary for advanced analysis and insight generation.

Once the abstracts are obtained, the next critical step is to transform this textual information into a format that is amenable to machine learning algorithms. For this, we employ OpenAI's 'text-similarity-davinci-001' embeddings. These embeddings translate the abstracts into a vector space, which is added as a property to the paper node enabling the application of natural language processing tasks such as semantic search, summarization, and comparison. By adding these embeddings as properties to the paper nodes in our Neo4j database, we significantly enhance the graph's utility, allowing for more nuanced and intelligent retrieval capabilities within the Intelli-research Assistant.

5 Model Experiments

5.1 Evaluation Dataset Creation

This section details the methodology employed in constructing an evaluation dataset of Cypher queries using OpenAI's GPT-4. The dataset comprises 150 data points, categorized into three difficulty levels: easy, medium, and hard. Each data point consists of an English question aligned with a corresponding Cypher query, designed to test the query-generating capabilities of GPT-4 in a database context.

5.1.1 Data Collection Method

The GPT-4 API, developed by OpenAI, was employed for this task. The model was provided with English question strings, each associated with the database schema. These question strings represent various types of queries that one might want to execute on a database.

One-Shot and Few-Shot Prompting Techniques:

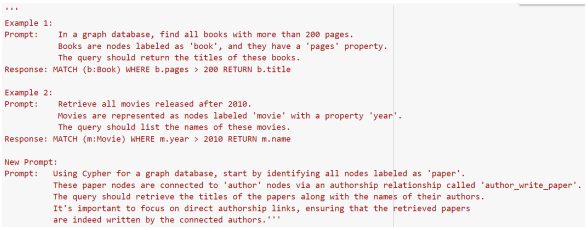

Figure 6: Few Shot Example

Two primary prompting techniques were used: one-shot and few-shot.

One-Shot Prompting: This involves providing the model with a single example or no example at all, relying solely on its pre-trained knowledge to generate a query.

Few-Shot Prompting: In contrast, this technique provides the model with a few examples of English questions and their corresponding Cypher queries. This approach helps in guiding the model about the format and structure expected in the output.These techniques are instrumental in teaching and guiding GPT-4 about the task at hand without extensive training on a specialized dataset. For example, see Fig.6

To ensure the validity of the generated queries, each was executed on a database. This step served to filter out syntactically incorrect or non-functional queries.

5.1.2 Correctness of Data

Manual Correction for Near Regex Matches: In this phase, the focus was on queries that were almost correct but had minor errors. These errors could be syntactical mistakes, slight misalignments with the database schema, or other small inaccuracies. The term "Near Regex Matches" implies that these queries closely resembled the correct format of Cypher queries, as defined by the regular expression patterns that typify the Cypher query language syntax. The manual correction process involved carefully reviewing each query and making precise adjustments to fix these small errors. This required a deep understanding of both the Cypher language and the specific database schema involved. By choosing to manually correct these queries, the goal was to maintain the original intent and structure of the query as generated by GPT-4, while ensuring its functional correctness. This approach respects the AI's output while fine-tuning it for practical application.

Chain of Thought Prompting for Significant Deviations: This approach was reserved for queries that had significant deviations from the correct syntax or logic. These deviations were more substantial than those addressed in the manual correction process and required a more comprehensive rethinking of the query. The "Chain of Thought" prompting method involved breaking down the query generation process into smaller, more manageable steps. This technique is akin to problem-solving, where each step builds on the previous one, leading to a logical and correct solution. In practice, this might involve first clarifying the intent of the original English question, then mapping out the logical steps needed to construct the query, and finally translating these steps into the correct Cypher syntax. This iterative and reasoned approach helps GPT-4 to better understand and correct its errors, leading to the generation of more accurate and logically sound queries. For example in Fig. 7 Both these techniques, Manual Correction and Chain of Thought Prompting are critical in ensuring the accuracy and reliability of the Cypher queries generated by GPT-4.

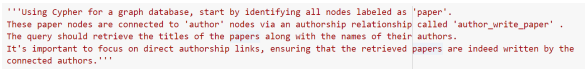

Figure 7: Chain of Thought Prompting

The process resulted in the generation of 150 preliminary Cypher queries, representing a diverse range of questions and complexities.

5.2 Hypothesis and Fine Tuning Experiment

5.2.1 Experiment 1

This experiment aimed to assess the efficacy of Large Language Models (LLMs) in generating runnable Cypher queries for a customized graph database. The LLMs, pre-trained on general Cypher query generation, were provided with user questions alongside the database schema. The dataset consisted of two columns: user questions and corresponding Cypher queries tailored for a customized database. The database schema was made part of the input to provide contextual understanding to the models. The experiment involved several LLMs, including Llama2, PaLM2, and GPT-3.5. These models are known for their capability in language understanding and generation but are not specifically trained for database querying.Each model was tasked to generate Cypher queries based on the provided user questions and

the database schema. The focus was on the models' ability to accurately use the database schema in the query generation process.

Results: The primary metric was the percentage of runnable queries generated by each model.As detailed in Table1: Llama2: Showed poor performance with only 16% of the queries being runnable. PaLM2: Achieved a moderate performance level with 46.4% runnable queries. GPT-3.5: Outperformed the others with 82.1% runnable queries. Error Analysis A significant number of errors were attributed to the models' inadequate understanding of the customized database schema, leading to incorrect use of relationship names and other schema-specific elements. Another noted issue was the models' inability to perform vector similarity searches, a feature recently developed by Neo4j in 2023. Based on the observed limitations, the hypothesis is that using custom models that are fine-tuned for querying would perform better in generating queries. Therefore the current popular language wrapper used in RAG for example Langchain which prompts LLM with the database's Schema can lead to very unstable and inaccurate results. Therefore our hypothesis is to teach LLMs to write cypher queries for our customized database through fine tuning on our customized dataset. Our current dataset has two columns, the users' questions and the corresponding cypher queries. Fine-tuning LLMs on a dataset specifically designed for our customized database will significantly improve their ability to generate accurate and runnable Cypher queries. This approach aims to tailor the model's understanding to the intricacies of the custom schema and its querying requirements.

5.2.2 Experiment 2

Experiment 2 was designed to test the hypothesis that custom models, fine-tuned specifically for database querying, would outperform general LLMs in generating accurate Cypher queries for a customized database. The dataset comprised user questions and corresponding Cypher queries tailored to a customized database. This dataset was created similar to our evaluation dataset creation. This dataset was used for fine-tuning the models.Different fine-tuning methods were employed for various LLMs:

PaLM2: Supervised fine-tuning using Google's Vertex AI.

Llama2 and CodeLlama: LoRA fine-tuning through HuggingFace.

GPT-3.5 Turbo: Supervised fine-tuning via the OpenAI API.

These models were selected based on their initial performances and potential for improvement through fine-tuning. The fine-tuning process focused on enhancing the models' ability to understand and generate Cypher queries directly from user questions, without requiring the database schema as input. The aim was to reduce the token length processing burden at inference time, previously necessitated by including the database schema in the prompts.

Results The primary metric was the percentage of runnable queries generated by each fine-tuned model. Results As detailed in Table1: PaLM2 and GPT-3.5: Exhibited the best performance with 96.4This marked improvement supports the hypothesis that fine-tuned models are more effective in this context. Comparative Analysis Initial Performance: Llama2, CodeLlama, PaLM2 initially showed low accuracy, suggesting a lack of understanding of the mapping from database schema plus user question to Cypher query. GPT-3.5 initially understood the schema and user questions better, starting with higher accuracy. All models showed significant improvement post fine-tuning, with PaLM2 and GPT-3.5 achieving the highest accuracy but Llama and CodeLlama does not perform as well as the rest. Eliminating the need to pass the database schema as part of the prompt led to a more efficient process, saving over 900 tokens per query during inference. However, the fine-tuned models offer a viable solution for diverse information retrieval and question-answering tasks on customized databases, showcasing the potential of fine-tuning in specialized application domains.

6 Evaluation

We break our evaluation into two parts. First, we evaluate the quality of natural language question to cypher query with Running Rate, BLEU score and ROUGE score, second part is to use GPTScore to evaluate the summarization quality from retrieved results.

Running rate is to make sure LLM's generated query actually run and retrieve results from our customized graph database. BLEU and ROUGE score is to see how the generated cypher matches the content and writing style of our cypher query. And we compare the results generated from the four

Table 1: Natural Language Question to Cypher Query results

Metrics	Run-Rate	BLEU	ROUGE-1	ROUGE-2
PaLM2	0.464	0.06	0.501	0.219
PaLM2_SFT	**0.964**	0.501	0.794	0.617
LlaMa2	0.16	0.017	0.395	0.133
LlaMa2_LoRA	0.892	0.479	0.64	0.549
CodeLlama	0.1	0.0	0.004	0.0
CodeLlama_LoRA	0.78	0.249	0.488	0.359
GPT3.5	0.821	0.154	0.595	0.315
GPT3.5_SFT	**0.964**	**0.786**	**0.907**	**0.832**

LLMs(PaLM2-text-bison002, LlaMa2-7b, CodeLlama, GPT3.5-turbo) without fine tuning and after fine tuning. For LLMs without fine tuning, we pass in the prompt our customized database's schema to help it know the keywords and structure when generating cypher query. But for fine tuned models, we prompt it in a zero shot manner, by just using natural language questions like: "What is the most cited paper in Machine Learning?". Although seems difficult without knowledge about the database's structure, our experiments show through fine tuning the goal is well-achieved.

We can see from Table1, after fine tuning all three models' performance drastically improved. This is a very much promising results that shows us the possibility of fine tuning LLMs for database retrieval and question answering on customized database. And very notably, Llama2 starts with a 0.16 rate to generate the correct query, but after LoRA fine tuning, the results increased to 0.892. Through fine tuning, the two open source models got close to GPT's performance in generating cypher queries for our customized database.

7 Summary prompt selection Experiment - GPT_Score

In the pursuit of optimizing the summarization capabilities of language models for clinical study reports, we devised an experiment titled "Summary Prompt Selection." This experiment aims to discern the most effective prompts that yield coherent summaries from a language model (LLM), given the absence of ground truth for direct correctness measurement. Coherence, rated on a scale from 1 to 5, serves as the primary metric, resonating with the DUC's quality criteria for structure and organization—summaries should transition smoothly from one sentence to the next, forming a cohesive narrative.

The experiment involved an LLM generating summaries based on a variety of prompts. Each prompt's effectiveness was evaluated by its coherence score—a higher score denoted a narrative that was not merely a collection of facts but a well-connected synopsis. However, this metric was subject to the underlying data distribution, occasionally skewing results towards certain scores. To mitigate this, a manual review of GPT notes was integral, providing a qualitative assessment to complement the quantitative scores.

For instance, a prompt requesting a summary inclusive of all details about a study scored lower in coherence, potentially due to information overload. Conversely, prompts demanding concise information tended to score higher, indicating that brevity might align with perceived coherence. This experiment underlines the importance of manual intervention in the prompt selection process, ensuring that the resulting summaries are not just compact but also contextually comprehensive. Through iterative evaluations, we refine the summarization process, steering towards prompts that encapsulate essential details without compromising the narrative's integrity.

7.1 Evaluation of English String size vs Query Precision

Our analysis revealed a slight negative correlation between the length of the English string in the queries and the BLEU score precision of the fine-tuned GPT-3.5 model's responses. The relationship is quantified by the regression equation *FT_BLEU = -0.0007*CHAR_LEN + 0.716*, suggesting that as the character length of the query string increases, the precision of the generated query slightly decreases. This trend is graphically depicted in a

Finding Summarizing Best Prompts	Coherence score	GPT Notes
Generate paragraph of study	5	Better coherence and detail
Generate paragraph of study (include study ID)	4.5	More comprehensive, includes study ID
How many studies in the database	4	Concise and direct answer
Write summary of study in database	3.5	More detailed and informative
Write summary of study, include all information	3	Comprehensive coverage of study details

Table 2: Evaluation of Summarizing Prompts

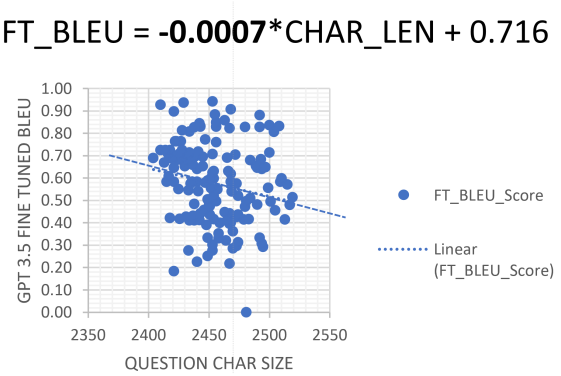

Figure 8: String size vs Query Precision

scatter plot with a fitted linear regression line, indicating that longer queries may introduce more complexity or ambiguity that could affect the model's performance.

However, it is important to note that the coherence of the responses does not necessarily degrade with length, as longer queries may provide more context and result in more informative answers. This nuanced understanding of query length and precision interplay is crucial for optimizing the use of LLMs in generating accurate and relevant responses to complex queries.

7.2 Evaluation of Query Syntax using Simple REGEX match to find Error Trends

To rigorously evaluate the accuracy of Cypher queries generated by GPT-3.5, we employed a regex-based evaluation method anchored in the known schema of our database. This approach allowed us to systematically validate the queries against a set of predefined patterns reflective of correct Cypher syntax and structure.

Our findings, as visualized in the accompanying pie chart, indicate that a significant proportion of queries—59%—lacked basic Cypher structure elements, highlighting a key area for model improvement. Misuse of property identifiers for common database elements such as 'author' and 'conference' was also prevalent, comprising 24% of the errors. This suggests a need for the model to better understand and apply the database's schema when generating queries.

Interestingly, our analysis also revealed that the model's performance degraded with entities having multiple synonyms within the database, causing frequent inaccuracies in the generated queries. These results underscore the challenge LLMs face in maintaining schema consistency, particularly when dealing with variable entity representations.

The regex evaluation thus proved instrumental in quantifying the model's proficiency in structuring queries and in identifying specific aspects where the model's training could be further optimized to enhance its understanding of complex query construction within the context of a structured database.

8 Conclusion and Future Work

This study has conducted an extensive evaluation of Large Language Models (LLMs) like GPT-3.5 and its fine-tuned version, alongside other models such as PaLM2 and LlaMa2, in the context of generating Cypher queries for graph databases. The standout finding is the superior performance of the fine-tuned GPT-3.5, which demonstrated approximately a twofold increase in run rate scores, as evidenced in the provided metrics.

The regression analysis indicated a minor negative correlation between the length of queries and the precision of results, urging us to consider query conciseness as a factor in model optimization. A further pattern was noted in the Regex Evaluation: nodes or relationships with similar names showed a higher rate of syntax errors. We also concluded that the **'Query first – LLM Later'** approach provided more consistent summarization outcomes.

As we look to the future, honing the fine-tuning

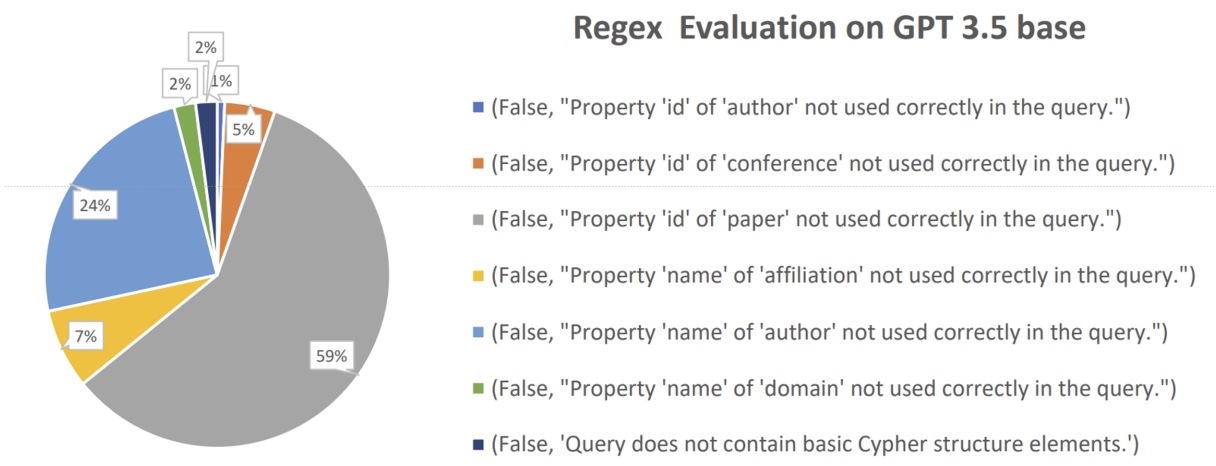

Figure 9: Query Syntax evaluation using regex

methods to adapt to synonymous entities and intricate schema terminologies remains a priority. The integration of domain-specific ontologies could address syntactical errors and enhance overall model efficacy. Expanding the evaluation to include diverse models and datasets is imperative in developing solutions that are market-ready for graph database users globally.

The potential for these solutions is underscored by market analyses; a report forecasts a compound annual growth rate (CAGR) of 21.9% for the global graph database market, with revenue expected to surge from USD 1.59 billion in 2020 to USD 11.25 billion by 2030[2]. This anticipated growth provides a robust backdrop for our continued research, which will focus on scalable and deployable LLM solutions to cater to this expanding demand for sophisticated data management systems in the burgeoning graph database market.

9 Expected Ethical Implications

When using LLM for database information retrieval, it's essential to ensure privacy and confidentiality, as sensitive data must be protected from unauthorized access or disclosure. Addressing bias is crucial, as LLMs may reflect biases in their training data, necessitating measures to promote fairness and avoid discrimination. Finally, transparency and accountability are key, with users needing clear information about the LLM's operation, data usage, and decision-making processes.

[2]Statista Report

10 Limitation

LLMs used for database information retrieval can sometimes provide inaccurate or misleading information, as they base responses on data patterns rather than verified facts. These models may struggle with context retention in extended interactions, leading to irrelevant or incomplete data outputs. Additionally, LLMs can reflect and amplify biases present in their training data, raising concerns about the fairness and impartiality of the information they retrieve.

11 Softwares and Technologies

Neo4j, LLMs, GCP, OpenAI, HuggingFace, AWS.

References

Zach Blumenfeld. 2021. Exploring practical recommendation systems in neo4j. *Towards Data Science*.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems.

Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2023. Genrec: Large language model for generative recommendation.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Zekun Lu, Qiancheng Yu, Xia Li, Xiaoning Li, and Qinwen Yang. 2023. Learning weight signed network embedding with graph neural networks. *Data Science and Engineering*, 8(1):36–46.

Lars-Peter Meyer, Claus Stadler, Johannes Frey, Norman Radtke, Kurt Junghanns, Roy Meissner, Gordian Dziwis, Kirill Bulert, and Michael Martin. 2023. Llm-assisted knowledge graph engineering: Experiments with chatgpt. *arXiv preprint arXiv:2307.06917*.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022a. Skill: structured knowledge infusion for large language models. *arXiv preprint arXiv:2205.08184*.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022b. Skill: Structured knowledge infusion for large language models.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.

Alex Tamkin, Miles Brundage, Jack Clark, and Deep Ganguli. 2021. Understanding the capabilities, limitations, and societal impact of large language models. *arXiv preprint arXiv:2102.02503*.

Ran Wang, Zhengyi Yang, Wenjie Zhang, and Xuemin Lin. 2020. An empirical study on recent graph database systems. In *Knowledge Science, Engineering and Management: 13th International Conference, KSEM 2020, Hangzhou, China, August 28–30, 2020, Proceedings, Part I 13*, pages 328–340. Springer.

Yancheng Wang, Ziyan Jiang, Zheng Chen, Fan Yang, Yingxue Zhou, Eunah Cho, Xing Fan, Xiaojiang Huang, Yanbin Lu, and Yingzhen Yang. 2023. Recmind: Large language model powered agent for recommendation.