https://doi.org/10.31891/csit-2025-4-11
UDC 004.8:004.652:004.912

SUTIAHIN Oleksandr, CHEREDNICHENKO Olga
National Technical University "Kharkiv Polytechnic Institute"

LLM-DRIVEN QUERY GENERATION FOR GRAPH-BASED BUSINESS INTELLIGENCE: TOWARDS A COLLABORATIVE KNOWLEDGE RETRIEVAL TOOL

This paper explores the use of large language models (LLMs) to support collaborative business intelligence in the tourism domain through two key tasks: extracting travel-related tags from user queries and generating Cypher queries for accessing knowledge graphs. We focus on evaluating the performance of compact and efficient LLMs, aiming to balance accuracy with computational feasibility. To assess tag extraction, we evaluated Phi-3 Mini, LLaMA 3.2, and Gemma 3 using the DeepEval framework with G-Eval scoring. Phi-3 Mini showed the best balance between accuracy and efficiency, while Gemma 3 achieved the highest scores at the cost of increased resource usage. For Cypher query generation, we tested more powerful models: Mistral Small 3.1, Phi-4, Gemma 3, and ChatGPT-4o. ChatGPT-4o achieved the highest correctness, while Mistral Small demonstrated the best trade-off among smaller models. Our results suggest that lightweight LLMs are suitable for basic natural language processing tasks, but structured query generation remains challenging and requires stronger models. Further research is needed to improve the reliability of generated queries and to develop robust validation mechanisms. This study introduces a comparative evaluation of lightweight and standard LLMs specifically applied to collaborative business intelligence in the tourism domain. It highlights the feasibility of using compact LLMs for natural language processing tasks while demonstrating the challenges of structured query generation, which requires more powerful models.

Keywords: Large Language Models, Cypher Query Generation, Knowledge Graphs, Collaborative Business Intelligence, Tourism Data Analysis, Prompt Engineering.

СУТЯГІН Олександр, ЧЕРЕДНІЧЕНКО Ольга
Національний технічний університет «Харківський політехнічний інститут»

LLM-КЕРОВАНА ГЕНЕРАЦІЯ ЗАПИТІВ ДЛЯ ГРАФОВОЇ БІЗНЕС-АНАЛІТИКИ: СТВОРЕННЯ ІНСТРУМЕНТУ СПІЛЬНОГО ОТРИМАННЯ ЗНАНЬ

У цій роботі досліджується використання великих мовних моделей (LLM) для підтримки колаборативної бізнес-аналітики на прикладі предметної області сфері туризму. Розглядається два ключові завдання: вилучення тегів, пов'язаних із подорожами, із запитів користувачів, та генерація Cypher-запитів для доступу до бази графів знань. Метою дослідження є пошук відповіді на питання чи можуть LLM служити ефективними посередниками між кінцевими користувачами та графовими системами знань, перетворюючи запити простою мовою у виконуваний код Cypher. Два дослідницькі питання поставлено: наскільки компактні LLM можуть точно видобувати релевантні теги або концепції із запитів користувача природною мовою? Наскільки ефективно LLM можуть генерувати синтаксично та семантично коректні запити Cypher на основі намірів користувача? Дослідження присвячено оцінювання ефективності застосування великих мовних моделей та зосереджується на оцінці продуктивності компактних та більш потужних LLM, прагнучи збалансувати точність та обчислювальні витрати. Для оцінки вилучення тегів було протестовано Phi-3 Mini, LLaMA 3.2 та Gemma 3 за допомогою фреймворку DeepEval із оцінюванням G-Eval. Модель Phi-3 Mini показала найкращий баланс між точністю та ефективністю, тоді як Gemma 3 досягла найвищих результатів, але з більшими витратами ресурсів. Для генерації Cypher-запитів було протестовано більш потужні моделі: Mistral Small 3.1, Phi-4, Gemma 3 та ChatGPT-4o. Модель ChatGPT-4o досягла найвищої точності, тоді як модель Mistral Small продемонструвала найкращий компроміс серед компактних моделей. Наші результати свідчать, що легкі LLM підходять для базових задач обробки природної мови, проте генерація структурованих запитів залишається складною і потребує потужніших моделей. Сучасні моделі демонструють проблеми з узгодженістю, особливо у створенні добре сформованих та виконуваних запитів протягом кількох ітерацій. Більшість помилок, що виникли під час експериментів, були пов'язані з випадками, коли модель видавала правдоподібні, але неправильні або нефункціональні запити. Подальші дослідження необхідні для підвищення надійності згенерованих запитів та розробки надійних механізмів валідації.

Ключові слова: великі мовні моделі, генерація cypher-запитів, графи знань, колаборативна бізнес-аналітика, аналіз туристичних даних, промт-інженерія

Received / Стаття надійшла до редакції 14.10.2025
Accepted / Прийнята до друку 11.12.2025

Introduction

The growing complexity of modern business environments, particularly in the tourism sector, demands new approaches to collective decision-making and knowledge management. In our previous work [1], we introduced a prototype of a collaborative business intelligence framework tailored for tourism domain analysis, emphasizing the integration of domain-specific knowledge into decision-support workflows.

In this study, we focus on enhancing the knowledge interaction component of that framework. Specifically, we propose storing collaboratively accumulated knowledge in a structured knowledge base, namely a graph database. Graph-based representations, such as those implemented in Neo4j, offer flexible and semantically rich data structures well-suited for capturing dynamic, interconnected domain knowledge. However, querying such graphs often requires proficiency in formal query languages like Cypher, which poses a significant barrier for non-technical users. To address this challenge, this study explores the use of large language models (LLMs) for natural language interface development. Nowadays, LLM has become more popular in almost every sphere of people's activity. For tourism sphere language models could solve different tasks: sentiment analysis [2], named entity recognition [3], question answering [4]. But in our case, we need that LLM answer on questions of user based on our data, which is saved in local database.

The knowledge graph (KG) was invented for better manipulation of knowledge [5]. The graph consists of entities that are connected by edges with semantic description to each other. Entities are real world objects that are presented like triples (subject, predicate, and object) and edges are relations between those objects. There are two types of construction graphs: top-down and bottom-up. Top-down is based on creating ontology first and after that we extract knowledge of this graph, second type works vice versa. One of the most popular storing databases of knowledge graphs is Neo4j [6].

Our goal is to investigate whether LLMs can serve as effective intermediaries between end-users and graph-based knowledge systems by translating plain language queries into executable Cypher code. We define two research questions:

RQ1: To what extent can compact LLMs accurately extract relevant tags or concepts from natural language user queries?

RQ2: How effectively can LLMs generate syntactically, and semantically correct Cypher queries based on user intent?

By answering these questions, we aim to contribute to the development of novel ICT tools that support intuitive knowledge retrieval, promote knowledge transfer, and reduce entry barriers to graph-based business intelligence systems.

Related works

Despite widespread success across various applications, large language models have faced criticism for limitations in factual accuracy. They tend to rely on memorized information derived from their training data, rather than demonstrating a true understanding of facts or external knowledge [7]. That leads to hallucinations of LLM and generation wrong answers. A possible solution to these issues is to integrate knowledge graphs into large language models. This study [8] provides a strategy for LLMs to communicate with KGs for improving reasoning. Authors developed three categories of interaction between LLM and KG: KG-enhanced LLM, LLM-augmented KG, synergized LLM + KG [8].

The first type of communication involves utilizing knowledge graphs during the inference stage of LLMs, enabling LLMs to retrieve knowledge from KGs during pre-training and tuning. Problem of this method is that it takes a lot of time to retrain the LLM. To solve it, the authors proposed to generate a prompt that based on a knowledge graph [8]. The prompt should be implemented by human and requires a lot of efforts. For example, MindMap [9] combines knowledge from a knowledge graph and an LLM by constructing reasoning pathways based on a knowledge ontology to solve question-answering tasks. So main idea is to extract key entities from query, then create and explore sub-graphs for those entities of nearest neighbors inside of knowledge graph and then prompt LLM to describe those reasoning graph structure. This helps LLM to build mind map and achieve knowledge reasoning on it. But this method is quite complicated and overengineered for tasks that don’t need overwhelming accuracy of answers.

Chain-of-Knowledge was developed by prompting LLM to solve complex tasks like commonsense, factual, symbolic, and arithmetic reasoning [10]. The algorithm starts with construction exemplars from KG, from which evidence triplets and explanation hints are gathered to make right prompt for LLM. Second category is about using large language models for inserting data to knowledge graphs, constructing knowledge graph and text encoding. StAR [11] utilizes textual encoders to independently encode input texts into contextualized representations. To address the combinatorial complexity inherent in pairwise textual encoding methods such as KG-BERT, StAR incorporates a scoring module that combines a deterministic classifier for representation learning with a spatial measurement component for structure learning. This dual mechanism not only reduces computational overhead but also enhances the integration of structured knowledge by capturing and leveraging the spatial characteristics of the encoded representations. Also LLMs are used to analyze KG models structure to enhance effectiveness in candidate retrieval tasks. But most suitable model for our tasks is knowledge graph question answering. Knowledge Graph Question Answering (KGQA) focuses on generating answers to natural language questions by leveraging structured information encoded within knowledge graphs [8]. So LLM is used as question reasoner and entity extractor for knowledge graph.

To more effectively leverage knowledge from both textual corpora and knowledge graph-based reasoning, this method proposes a unified framework designed to perform joint reasoning by integrating the complementary strengths of large language models and knowledge graphs [12]. This integration fosters a comprehensive framework that consolidates existing knowledge and identifies new avenues for real-world applications, ultimately amplifying the translational impact of academic research in related fields. This collaboration is essential for advancing both technologies effectively [13]. We identified our tasks mostly as KG-enhanced LLM category with prompting LLM by information from knowledge graph. So we developed our method based on this technology.

Methodology

It’s quite a complicated task to choose the right data for performing business analysis, usually non-professional users don’t have such knowledge. That’s why they need some help during their work with BI tools. Virtual assistant can ask for more details from users or provide some similar cases that were already created by other users. Overall, our proposed BI4Tourism application architecture, as illustrated in Fig. 1, aims to give non-professional users the ability to make business analysis of tourism data in a collaborative way with other users or cooperative chatbot [1]. In our framework, the data storage is distributed across multiple databases. MongoDB is utilized for storing collected information related to places and weather from the internet. PostgreSQL manages user-generated use cases, while Neo4j transforms and represents data from PostgreSQL in a graph-based structure as a knowledge graph. Also, our LLM is connected to the Neo4j database to query it. So, our final graph is presented in Fig. 2.

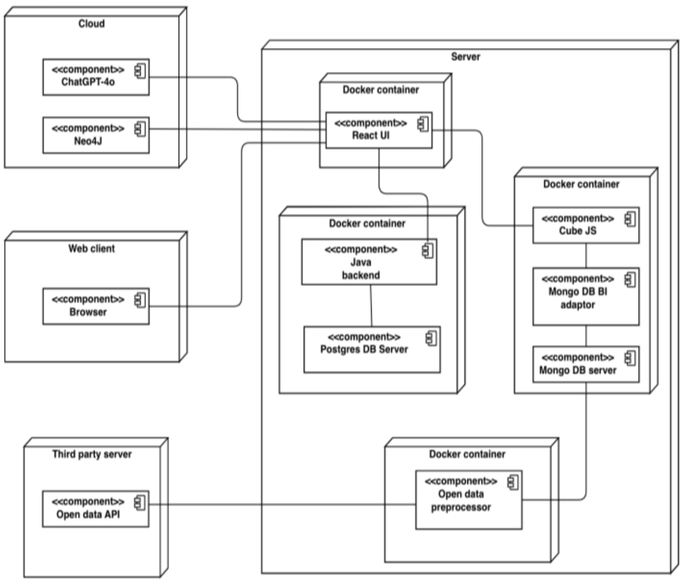

Fig. 1. Bi4Tourism architecture [1]

In this study we will investigate only part of the implementation of a chatbot. We prompted the chatbot beforehand for those actions with a prompt in Appendix A. So, we identified the main use cases in the next scenarios. For the first scenario we have a new user that wants to gather some information about a trip he planned. His main aim is to explore the Monet museum and then visit the nearest park if the weather is fine. But the tourist has a problem with crowds so the less people he meets the better it would be. That’s why he tries to come to those places in less crowded time. All requirements are transmitted to the pre-trained LLM. The chatbot extracts relevant tags from the text and utilizes them to identify previously created use cases with the highest tag similarity. To achieve this, a Cypher query is generated with the extracted tags as parameters to retrieve the most similar use case.

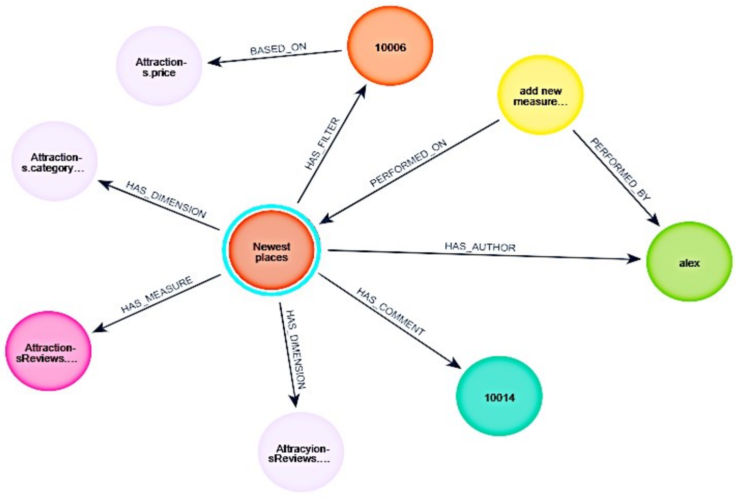

Fig. 2. Knowledge graph

This query is constructed by the LLM based on prior prompting. If the user finds any of the presented use cases satisfactory, they can access the corresponding use case page, analyze the result chart, and provide feedback by submitting a comment indicating that the use case was beneficial. However, if the user is not satisfied, the chatbot collects additional contextual information. Based on this data, the system generates a query to create a new use case, facilitating the generation of a corresponding chart for further analysis. Finally, the user can see that on the chart of weather next Thursday afternoon is sunny and Monet Museum is quite empty during 16:00 hours on this day. His final decision will be to visit the museum on Thursday and then have a walk in Jardin du Ranelagh Park. All this final information must be mentioned in comment for next users' usage and saved in PostgreSQL and then transferred to Neo4J. The result is in Fig.3.

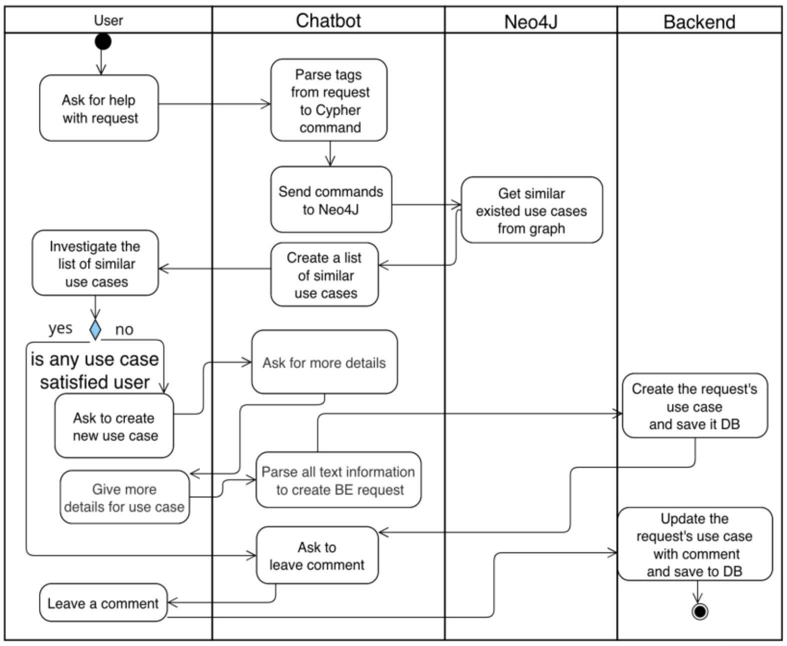
User
Chatbot
Neo4J
Backend
Ask for help with request
Parse tags from request to Cypher command
Send commands to Neo4J
Get similar existed use cases from graph
Investigate the list of similar use cases
Create a list of similar use cases
yes
no
is any use case satisfied user
Ask for more details
Create the request's use case and save it DB
Ask to create new use case
Parse all text information to create BE request
Give more details for use case
Update the request's use case with comment and save to DB
Ask to leave comment
Leave a comment

Fig. 3. Creation of new use case workflow

The second scenario is like the first; however, in this case, the user is uncertain about activities in Paris. The chatbot assists by suggesting leisure options for the trip. It requests additional details from the user and extracts relevant tags, which are then used to construct a Cypher query. This query retrieves the most popular use cases based on criteria such as the highest number of comments and the highest user ratings.

Experiments

We conducted two independent experiments to identify optimal configurations for our natural language processing pipeline. The first experiment aimed to determine the most suitable LLM for extracting travel-related tags from user-input sentences. This involved benchmarking multiple LLMs using standardized Retrieval-Augmented Generation (RAG) metrics, including answer correctness, precision, and recall, within the DeepEval evaluation framework. The second experiment focused on prompt engineering for ChatGPT-4, with the objective of identifying the most effective LLM for translating user queries into Cypher, the query language for graph databases. All LLM models, except ChatGPT-4o, were deployed and evaluated locally on MacBook Pro 2021. We collected approximately 100 tourism-related queries from real people about Paris from Reddit [14] and manually extracted keywords and relevant thematic tags associated with tourism to build a dataset for Tag retrieval task. In this study, we evaluated the suitability of various large language models (LLMs) for the task of extracting travel-related tags from natural language sentences.

To determine the most effective model, we assessed key Retrieval-Augmented Generation (RAG) performance metrics, specifically answer correctness, recall, and precision. For this evaluation, we employed the DeepEval framework [15], which was selected based on evidence from GroUSE [16] – a meta-evaluation tool for benchmarking evaluators. According to GroUSE, DeepEval outperforms comparable frameworks such as RAGAS [17] in terms of faithfulness, correctness, and answer relevance when applied to grounded question answering tasks. DeepEval utilizes the LLM-as-a-judge paradigm, which has been demonstrated to offer greater reliability compared to traditional statistical or human-based evaluation approaches [18].

During implementation of evaluation in DeepEval framework we chose a prompt-based GPTEval method [19] which uses a chain of thoughts model to assess natural language generation output. The framework asks LLM to score some value for each evaluation aspect, based on the defined criteria, then LLM should add weights to those scores and summarize it. We defined our evaluation aspects for tag retrieval in Python code variable in Appendix C. For GPTEval scoring function is presented as predefined set of discrete scores (e.g., 1 to 5) specified within the prompt, serving as the evaluation scale for subsequent assessments. $S = \{s_1, s_2, \ldots, s_n\}$, the probability of each score $p(s_i)$ is calculated by the LLM, and the final score (1) is:

$$score = \sum_{i=1}^{n} 2p(s_i) \times s_i. \tag{1}$$

Average time execution was calculated simply by average tests execution for all five Python test series and gathered by DeepEval framework. After execution of LLM function to retrieve tags, we must validate whether those tags are presented in our system and LLM is not hallucinated, so we prepared one more validation step and in case of invalid tags, we asked LLM to try again retrieve tags with previous. We configured the whole process of evaluation in the form of five use cases, each containing twenty tests with data from our dataset, and wrote it in Python with the DeepEval framework. For testing we choose three most popular tiny LLMs: Llama 3.2 (3B), Gemma 3 (4B), Phi-3 Mini (3.8B). We chose those lightweight models due to low complexity of task and availability of deploying locally on the average laptop.

The First Experiment Results

The evaluation results are summarized in Table 1. Among the tested models, Gemma 3 achieved the highest overall scores across the considered metrics. However, it exhibited significant inefficiencies in terms of time and resource consumption, particularly during precision evaluation. LLaMA 3.2 demonstrated the lowest performance in answer correctness and maintained only average efficiency with respect to time consumption. Based on the trade-off between performance and resource utilization, Phi-3 Mini was selected as the most suitable model. Although its precision and recall were moderate, it delivered consistently high scores in answer correctness, which was prioritized for this application.

The Second Experiment Results

For datasets we use the same questions about Paris from Reddit [14] and create our own cloud based Neo4J database and fill it with data manually. Our main task was to evaluate which LLM will suit best for our task of creating a Cypher query to the Neo4j database. Firstly, we found out that our small LLM from the previous experiment couldn't handle this work, so we had to increase the number of parameters. The method was the same as for the previous investigation – DeepEval with the G-Eval framework. We developed a prompt that helps LLM to create proper Cypher query, the prompt can be checked in Appendix A. Also, in case Cypher query will be invalid or return wrong result we implemented validators that will force LLM to rewrite Cypher query. The prompt for this action could be checked in Appendix B.

Table 1

Result for different LLM

LLM	**Score**			**Average time execution**		
	Answer correctness	**Recall**	**Precision**	**Answer correctness**	**Recall**	**Precision**
Llama 3.2	0,61	0,89	0,90	125s	65s	68s
Gemma 3	0,65	0,93	0,95	130s	69s	76s
Phi-3 Mini	0,69	0,91	0,92	127s	70s	74s

We configured the whole process of evaluation in the form of two use cases, each containing ten tests with data from our dataset, and wrote it in Python with the DeepEval framework. For testing we choose three most popular tiny LLMs: Mistral Small 3.1 (24B), Gemma 3 (27B), Phi-4 (14B), ChatGPT-4o. Among the models, ChatGPT-4o achieved the highest score in answer correctness, indicating superior capability in generating accurate responses but average precision and recall (Table 2). But among smaller LLMs, Mistral Small 3.1 shows the best result in answer correctness but average resource consumption and recall with precision. However, Phi-4 consumed all GPU resources of the laptop and the response for each query was the longest. Gemma 3 was average on all metrics.

Table 2

Result for different LLM

LLM	**Score**			**Average time execution**		
	Answer correctness	**Recall**	**Precision**	**Answer correctness**	**Recall**	**Precision**
Mistral Small 3.1	0,72	0,85	0,89	50s	34s	32s
Gemma 3	0,67	0,90	0,91	39s	31s	28s
Phi-4	0,65	0,86	0,83	64s	28s	24s
ChatGPT-4o	0,74	0,85	0,84	40s	30s	32s

In our study, we established that a minimum of a 3-billion-parameter large language model with English language understanding and free availability, is required for effective performance in the tag retrieval task. The evaluation was structured into five test series, each consisting of twenty test cases, with the complexity of each subsequent series incrementally increased. Interestingly, the LLMs demonstrated progressively higher scores across successive series, suggesting a potential in-context learning effect - where the model improved its tag retrieval capabilities based on prior exposure to earlier test cases. Among the evaluated models – LLaMA 3.2, Gemma 3, and Phi-3 – our findings indicate that Phi-3 consistently outperformed the others and emerged as the most suitable choice for this task.

For the Cypher query creation task, we found that locally deployed LLMs with at least 15 billion parameters are not able to overperform ChatGPT-4o. But in case of low budget requirements, we can use Mistrall Small 3.1, which answer correctness metric was highest among Gemma 3 and Phi-4. So the main idea is that the bigger number of parameters LLM has the better it will solve the Cypher query generation task.

Discussion and Conclusion

This study explores how collaborative business intelligence can be enhanced through the integration of a prepared chatbot interface and a knowledge base. The central focus lies in leveraging large language models to process user queries formulated in natural language, with the goal of extracting relevant tags and automatically generating Cypher queries for interaction with a Neo4j graph database. We designed an experimental framework to evaluate the performance of several LLMs on two key tasks: (1) retrieval of tags from user input, and (2) generation of Cypher queries based on retrieved tags. Among the evaluated models, Phi-3 Mini demonstrated the highest overall score in tag retrieval, while ChatGPT-4o outperformed others in generating syntactically and semantically correct Cypher queries. Notably, Mistral Small 3.1 emerged as the best-performing model among compact LLMs, offering a strong balance between performance and computational efficiency.

Our findings indicate that even small-scale LLMs can serve as effective tools for interpreting and processing natural language expressions in business intelligence scenarios. However, the generation of structured queries, such as Cypher, remains a challenging task. Current models struggle with consistency, particularly in producing well-formed and executable queries across multiple iterations. Most errors encountered during the experiments were linked to hallucinations—instances where the model produced plausible but incorrect or non-functional queries. Encouragingly, such issues may be mitigated through more precise prompt engineering. Despite these promising outcomes, several limitations and concerns merit discussion. First, the reliance on prompt engineering for controlling LLM behavior introduces fragility and subjectivity into the system. Minor changes in phrasing or context can lead to significantly different outputs, undermining reliability in enterprise scenarios where consistency is crucial. Second, while LLMs exhibit impressive language understanding, they lack domain-awareness and often fail to account for database schema constraints or data types, leading to invalid queries. This limitation suggests the need for hybrid approaches that combine LLMs with schema-aware systems or symbolic reasoning modules to ensure query validity. Third, performance evaluations were conducted in a controlled setting with predefined tasks and data. This raises concerns about generalizability. It remains unclear how well these models would perform in real-world CBI platforms, where user inputs are highly variable and ambiguous. Furthermore, smaller models such as Mistral Small showed competitive results. This trade-off between efficiency and capability highlights a key design tension in deploying LLMs in resource-constrained environments.

While our work does not yet fully resolve the problem of reliable Cypher query generation, it contributes a step forward by formalizing a query generation pipeline and demonstrating the potential of LLMs in facilitating user interaction with graph databases. In future work, we aim to explore methods for reliably validating generated queries.

Author Contributions

Conceptualization, O.S. and O.C.; methodology, O.S.; software, O.S.; validation, O.C.; formal analysis, O.S.; investigation, O.S.; resources, O.S.; data curation, O.S.; writing – original draft preparation, O.S.; writing – review and editing, O.S. and O.C.; visualization, O.S.; supervision, O.C. All authors have read and agreed to the published version of the manuscript.

Declaration on the use of generative artificial intelligence tools

In preparing this manuscript, the authors used Grammarly and ChatGPT solely for language-related assistance, including grammar checking, spelling correction, and improvement of sentence clarity. These tools were not used to generate new scientific content, paragraphs, or sections of the manuscript. All scientific content, interpretation of results, and conclusions were produced by the authors. After using these tools, the authors critically reviewed and edited the text and take full responsibility for the content of this publication.

References

1. Cherednichenko, O., Sutiahin, O.: Development of Collaborative Business Intelligence Framework for Tourism Domain Analysis. In: Tekli, J., et al. (eds.) New Trends in Database and Information Systems. ADBIS 2024. CCIS, vol. 2186, pp. 270–281. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-70421-5_21
2. Arefieva, V., Egger, R.: TourBERT: A Pretrained Language Model for the Tourism Industry. arXiv preprint arXiv:2201.07449 (2022)
3. Gao, K., Zhou, J., Chi, Y., Wen, Y.: TourismNER: A Tourism Named Entity Recognition Method Based on Entity Boundary Joint Prediction. Intell. Syst. Appl. 25, 200475 (2025). https://doi.org/10.1016/j.iswa.2025.200475

4. Wei, Q., Yang, M., Wang, J., Mao, W., Xu, J., Ning, H.: TourLLM: Enhancing LLMs with Tourism Knowledge. arXiv preprint arXiv:2407.12791 (2024)
5. Tan, X., Wang, X., Liu, Q., Xu, X., Yuan, X., Zhang, W.: Paths-over-Graph: Knowledge Graph Empowered Large Language Model Reasoning. arXiv preprint arXiv:2410.14211 (2024)
6. Tamašauskaitė, G., Groth, P.: Defining a Knowledge Graph Development Process Through a Systematic Review. ACM Trans. Softw. Eng. Methodol. 32, 1–40 (2022). https://doi.org/10.1145/3522586
7. Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.: Language Models as Knowledge Bases? In: Proc. EMNLP-IJCNLP, pp. 2463–2473 (2019)
8. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying Large Language Models and Knowledge Graphs: A Roadmap. IEEE Trans. Knowl. Data Eng. (2024)
9. Wen, Y., Wang, Z., Sun, J.: Mindmap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models. arXiv preprint arXiv:2308.09729 (2023)
10. Wang, J., Sun, Q., Chen, N., Li, X., Gao, M.: Boosting Language Models Reasoning with Chain-of-Knowledge Prompting. arXiv preprint arXiv:2306.06427 (2023)
11. Wang, B., Shen, T., Long, G., Zhou, T., Wang, Y., Chang, Y.: Structure-augmented text representation learning for efficient knowledge graph completion. In: WWW 2021, pp. 1737–1748 (2021)
12. 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. arXiv preprint arXiv:2005.11401 (2020)
13. Li, D., Xu, F.: Synergizing Knowledge Graphs with Large Language Models: A Comprehensive Review and Future Prospects. arXiv preprint arXiv:2407.18470 (2024). https://doi.org/10.48550/arXiv.2407.18470
14. Reddit: Paris Travel Guide. https://www.reddit.com/r/ParisTravelGuide/ , last accessed 2025/05/13
15. Confident AI: deepeval documentation. https://docs.confident-ai.com/ (2025), last accessed 2025/05/13
16. Muller, S., Loison, A., Omrani, B., Viaud, G.: GroUSE: A Benchmark to Evaluate Evaluators in Grounded Question Answering. arXiv preprint arXiv:2409.06595 (2024)
17. Shahul Es, J.J., Espinosa-Anke, L., Schockaert, S.: Ragas: Automated Evaluation of Retrieval Augmented Generation. arXiv preprint arXiv:2309.15217 (2023)
18. Gu, J., Jiang, X., Shi, Z., Tan, H., Zhai, X., Xu, C., Li, W., Shen, Y., Ma, S., Liu, H., Wang, Y., Guo, J.: A Survey on LLM-as-a-Judge. arXiv preprint arXiv:2411.15594 (2024)
19. Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., Zhu, C.: G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. arXiv preprint arXiv:2303.16634 (2023)

Oleksandr Sutiahin **Олександр Сутягін**	PhD student, National Technical University "Kharkiv Polytechnical Institute", Kharkiv, Ukraine, e-mail: sutiahin.oleksandr@cs.khpi.edu.ua https://orcid.org/0009-0005-6527-455X, Scopus Author ID: 59424446500	аспірант, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна.
Olga Cherednichenko **Ольга Чередніченко**	D.Sc., Professor of Software Engineering and Management Intelligence Technologies Department, National Technical University "Kharkiv Polytechnical Institute", Kharkiv, Ukraine, e-mail: olga.cherednichenko@khpi.edu.ua https://orcid.org/0000-0002-9391-5220, Scopus Author ID: 55332933600, ResearcherID: M-8261-2015	професор кафедри програмної інженерії та інтелектуальних технологій управління, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна.

Appendix A

```
This is the schema representation of the Neo4j database.
  Node properties are the following:
  {node_props}
  Relationship properties are the following:
  {rel_props}
  Relationship point from source to target nodes
  {rels}
  Make sure to respect relationship types and directions
Additional explanation:
         Mainly our database represents ontology of use cases that were
created by users to investigate tourism in BI way.
     Data is presented with measures, use case filters (each filter consists
of dimension, operation and value of filter),
     dimensions, tags, comments and use cases that merge those entities in a
single object.
     In comments users can leave their opinion on how some use cases help
them with their trip.
    You have to check tags, comments, use case filters for recommendation.
     We consider that similarity between use cases depends on similar tags.
    You have to be BI tourism adviser that retrieves from users' questions
part of use cases, dimensions,
    measure, comments, tags, use cases' description, use case filters that
are connected to some dimensions.
    If user mentioned something similar to limitations, you have to check
useCaseFilters from schema also.
    Example:
    Q: What are the best hotels by stars in Paris?
    Chat: So chat should consider that we have limitations - hotels (that
looks like categories),
     Paris sounds like a concrete place and could be tag, stars - looks like
some measure.
     So query should be generated with use case filters by categories = hotel
     then we should check by tag Paris and them add check for measure
connected to stars.
  Don't include ```cypher\n, ```cypher to Cypher request.
We retrieve node properties (node_props), relation properties (rel_props),
relations (rels) from Neo4J database dynamically.
```

Appendix B

```
def retryErrorOneMoreTime(self, cypher, e, question):
        return self.run(
            question,
            [
                {"role": "assistant", "content": cypher},
                {
                    "role": "user",
                    "content": f"""This query returns an error: {str(e)}
                        Give me a improved query that works without any
explanations or apologies""",
                },
            ],
            retry=False
        )
```

Appendix C

```
correctness_metric = GEval(
    name="CORRECTNESS",
    evaluation_params=[
        LLMTestCaseParams.EXPECTED_OUTPUT,
        LLMTestCaseParams.ACTUAL_OUTPUT],
    evaluation_steps=[
        "Main aim is to parse tags from the user sentence and present them
like an array of strings",
        "We don't need answer the question, we need only parsed tags",
        "Score every evaluation step from 1 to 5",
        "We need as much as possible tags from expected_output in
actual_output",
        "Tags in actual_output can be in random order"
        "All tags must be retrieved from sentence."
        "Tag in actual_output could be synonym to expected_output."
    ],
)
```