Pythia: Facilitating Access to Internet Data Using LLMs and IYP

Dimitrios Giakatos
IIJ Research Laboratory
Tokyo, Japan
dimitrios@iij.ad.jp

Malte Tashiro
IIJ Research Laboratory
Tokyo, Japan
malte@iij.ad.jp

Romain Fontugne
IIJ Research Laboratory
Tokyo, Japan
romain@iij.ad.jp

***Abstract*—Internet data analysis is essential for policymakers and regulators to make informed decisions. Despite the various datasets made available by the research community, the analysis of these datasets is challenging due to their various formats and required analysis tools. The Internet Yellow Pages (IYP) database is designed to simplify Internet data analysis by combining many datasets into a single format. Hence, IYP enables users to retrieve data from various Internet datasets using a single querying language called Cypher. However, learning Cypher and understanding IYP's schema is still challenging. Large Language Models (LLMs) offer the potential to generate Cypher queries from English text. In this paper, we assess the ability of different LLMs to generate Cypher queries. We introduce CypherEval, a dataset for evaluating LLM-generated Cypher queries, and we propose a methodology to benchmark LLMs for IYP. Finally, we present Pythia, a system for the generation of IYP Cypher queries.**

***Index Terms*—LLMs, Internet Yellow Pages, Internet data, Cypher, Benchmark.**

I. Introduction

The availability of Internet data is crucial for network operators, policymakers, regulators, and researchers. They rely on this data to get actionable insights and make well-informed decisions that shape the evolution of the Internet [1]. Although there are many datasets publicly available [2]–[7], each has its unique format and requires expertise in specific analytical tools for a thorough examination, understanding, and analysis of the data.

The Internet Yellow Pages (IYP) [8] is a knowledge graph database built with Neo4J that combines Internet data, such as Autonomous System Numbers (ASNs), IP addresses, prefixes, and domain names, from various sources into a single format. It includes datasets from over 24 different organizations, including CAIDA, RIPE, and PeeringDB, among others. Each dataset includes multiple features describing the Internet data, which IYP maps to node attributes. As a result, the graph currently contains more than 35 million nodes.

In IYP, nodes can form relationships with one another. For instance, the nodes ”AS” and ”Organization” are linked by a ”Managed by” relationship, indicating that a particular AS is managed by a specific organization. Overall, the database includes approximately 180 million relationship attributes. Both nodes and their relationships have properties used to store data from the original datasets. For example, an ”AS” node includes an ”asn” property that has the autonomous system number, while a ”Managed by” relationship includes a ”reference_org” property that specifies the dataset's organization of that relationship.

This design enables IYP to support multiple datasets using the same nodes and relationships, distinguished only by their associated properties. This minimizes the need for users to learn different tools to analyze the data. With IYP, users only need to learn Cypher, a querying language for Neo4J graph databases [9].

Despite facilitating access to Internet data, querying IYP is still challenging. It requires a good understanding of both Cypher and the IYP database schema[1]. The schema provides detailed descriptions of node and relationship attributes, their properties, and how they interconnect, enabling users to understand the structure and semantics of the data. However, due to the inclusion of numerous open-source Internet data datasets, the schema is highly complex. Furthermore, the frequent addition of new datasets into IYP continually modifies the database schema, and users have to stay updated with these changes.

To address these challenges, this paper explores the benefits that Large Language Models (LLMs) can provide to simplify interactions with the IYP database. In particular, we aim to evaluate the ability of LLMs to translate an English question related to the Internet into a Cypher query for IYP and provide a practical LLM-based solution for easy access to IYP data.

Our study is based on different LLMs trained for code generation [10]–[15]. Although these LLMs can already generate Cypher from English text to the best of our knowledge, no past study has evaluated the use of LLMs with graph databases, so it is unclear how effective these models are in generating Cypher queries from English text.

In this work, we create CypherEval, a real-world ground-truth dataset for evaluating LLMs' Cypher generation, and propose a methodology leveraging CypherEval to benchmark LLMs. We compare the performance of four popular LLMs to generate Cypher queries, showing their potential and limitations. In addition, we develop Pythia, a system that significantly improves the generation of IYP Cypher queries from English text in comparison with LLMs.

The main contributions of this paper are:

[1]https://codeberg.org/dimitrios/Pythia/src/branch/main/IYP-Schema.md

- CypherEval, a real-world dataset designed to evaluate LLMs on generating Cypher queries from English text.
- Benchmark results for popular LLMs to generate Cypher queries.
- Pythia, a system that improves the IYP querying problem.

II. CypherEval: Ground-Truth Dataset

CypherEval (Table I) is a ground-truth dataset that we create especially to evaluate the ability of LLMs to generate Cypher queries for IYP [8]. The dataset includes 83 Cypher queries, each producing a specific result. Each query is paired with two types of English prompts, one technical and one general, to simulate varying levels of user expertise. The technical prompts reflect questions asked by users familiar with the nodes, relationships, and properties of the IYP database. The general prompts mimic users who do not have specific knowledge of the database structure. This results in a total of 166 prompts.

To further imitate the diversity of human communication for each prompt, we include two variations with linguistic differences but the same meaning. This increases the dataset to 332 ground-truth English prompts.

The English prompts cover a wide range of real-world networking use cases with varying difficulty levels, defined by the complexity of constructing the Cypher query needed to return the appropriate results. These prompts have been independently reviewed by three IYP users and Internet measurement researchers. The reviewers validated each prompt by executing the canonical solution on the IYP database and confirming that the results accurately addressed the corresponding prompt. This validation process ensures the reliability of the dataset, i.e., each prompt aligns with its canonical solution and is relevant to the IYP database. Furthermore, all prompts represent meaningful real-world networking scenarios, such as determining the ranking of ASes in Japan, used regularly by researchers and network operators.

We release CypherEval[2] as an open-source dataset. Our goal is to provide a reliable resource for evaluating LLMs' capability to effectively generate Cypher queries from English text.

III. CypherEval-Based Benchmarking

We propose a CypherEval-based benchmarking methodology and evaluate the performance of open-source 7B LLMs in generating Cypher queries based on functional correctness. We select CodeLlama 7B (CodeLlama), DeepSeek Coder 6.7B (DeepSeek), Qwen2.5 Coder 7B (Qwen), and Granite Code 8B (Granite) as open-source LLMs. Although DeepSeek and Granite are not exactly 7B LLMs, we include them in our evaluation since their sizes are comparable to 7B LLMs.

Benchmarking these models consists in the following three steps (Fig. 1):

1) Define the database schema format for LLM prompting.
2) Determine which database schema format is the best for LLM Cypher generation.
3) Evaluate LLMs efficacy in generating Cypher queries from English text for IYP using functional correctness.

To measure functional correctness we use $pass@k$ [16], [17]. This metric is computed by generating n code samples for a prompt and estimating the likelihood that the LLM provides at least one correct code sample in $k \leq n$ samples. Consequently, a $pass@1 = 1$ means that the LLM is likely to answer correctly every time, and a $pass@100 = 1$ means that the LLM is likely to generate at least one correct answer when asked 100 times.

A. Defining the Neo4J Schema Format

LLMs are generally fine-tuned to function as general assistants [18], which limits their performance in specialized tasks [11]. To overcome this limitation, we use prompt engineering to configure the LLM specifically for this task [19], i.e., generating Cypher queries from English text. We utilize a standard prompt template for code generation as proposed by [20], [21], which includes the database schema to give the LLM an understanding of the relationships between entities within the database. However, LLMs are not efficiently using the database schema if it is presented in a different format from what the LLM was trained on [22], which is unfortunately not disclosed for the four studied LLMs [23].

To address this, we create five schema formats[3] to represent the database schema. The first two formats are inspired by data structures used in the bibliography [24]: JSON and CSV. The remaining formats are generated by prompting well-known LLMs (i.e., GPT-3.5, CodeLlama, DeepSeek, Qwen) to generate Neo4J schema formats. While the LLMs produce broadly similar outputs, their responses differed slightly in structure and phrasing. We grouped these variations into three distinct schema formats that reflect visibly different structural patterns rather than minor textual differences. By using these five schema formats, we aim to determine which format the LLMs best understand [19], [21], enabling them to generate accurate Cypher queries.

B. Determining the Impact of Schema Format and LLMs

Experimenting with CypherEval is challenging due to the rapid increase in computation time, which is closely tied to the size of the input prompt. Larger prompts result in longer inference times for LLMs, while shorter prompts yield faster responses [25]. Since CypherEval relies on the IYP database, each LLM prompt must include the IYP schema alongside the CypherEval prompt, that significantly increases the total LLM prompt length and, consequently, the inference time.

To address this issue, we base our experiments on the Movies database [26]. The Movies database is the example Neo4J database used in the documentation of the Cypher syntax [27]. This database is widely used in tutorials and educational materials, making it likely to be part of the training data of the LLMs we evaluate. As a result, we expect the models to generate more accurate Cypher queries when using this familiar database [28].

[2]https://codeberg.org/dimitrios/CypherEval

[3]https://codeberg.org/dimitrios/CypherEval/src/branch/main/Neo4J-Schema-Formats.md

TABLE I

A SNAPSHOT FROM THE CYPHEREVAL DATASET SHOWING FOUR UNIQUE CYPHER QUERIES. THE SECOND COLUMN SHOWS THE DIFFICULTY. EACH CYPHER QUERY CONTAINS TWO PROMPTS, ONE TECHNICAL AND ONE GENERIC (COLUMN THREE). THE FOURTH COLUMN INCLUDES THE CANONICAL SOLUTION USED DURING THE FUNCTIONAL CORRECTNESS STEP.

Task ID	Difficulty Level	Prompt	Canonical Solution
1.1	Easy technical prompt	Find the IXPs' names where the AS with asn 2497 is present.	`MATCH (:AS{asn:2497})-[:MEMBER_OF]->(ixp:IXP) RETURN DISTINCT ixp.name`
1.2	Easy general prompt	Give me the names of the IXPs where AS2497 is member.	`MATCH (:AS{asn:2497})-[:MEMBER_OF]->(ixp:IXP) RETURN DISTINCT ixp.name`
3.1	Medium technical prompt	For AS with asn 2497 find all rank values and their corresponding Ranking name for Rankings corresponding to Country with country_code "JP".	`MATCH (:AS{asn:2497})-[r1:RANK]-(r2:Ranking)--(:Country{country_code:"JP"}) RETURN r1.rank, r2.name`
3.2	Medium general prompt	Find the rank and the corresponding rankings name for AS2497 in Japan.	`MATCH (:AS{asn:2497})-[r1:RANK]-(r2:Ranking)--(:Country{country_code:"JP"}) RETURN r1.rank, r2.name`
4.1	Hard technical prompt	Find all the AS nodes that have peer to peer relationship with the AS with asn 2497.	`MATCH (:AS{asn:2497})-[:PEERS_WITH{rel:0}]-(a:AS) RETURN a`
4.2	Hard general prompt	Which ASes have a settlement-free peering with AS2497?	`MATCH (:AS{asn:2497})-[:PEERS_WITH{rel:0}]-(a:AS) RETURN a`

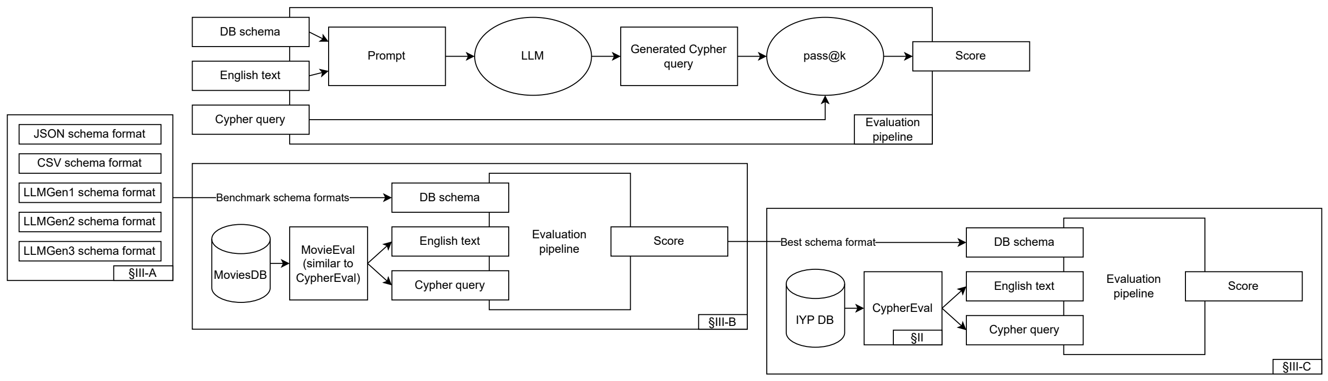

Fig. 1. Presents the three steps of our benchmarking methodology introduced in §III. The "Evaluation Pipeline" illustrates how we utilize LLMs to evaluate generated responses, as applied in §III-B and §III-C.

Using the Movies database allows us to evaluate how different schema formats influence Cypher query generation while minimizing factors from unfamiliar content. This setup helps us assess which schema formats LLMs handle best and which are most effective at generating Cypher queries [29].

We first run CodeLlama, DeepSeek, Qwen, and Granite on 36 natural language queries with the same structure as those used in CypherEval, but for the Movies database [26]. For each of the 36 natural language queries and five different Neo4J schema formats, we generate 100 Cypher queries. This results in a total of 18,000 Cypher queries, which we evaluate for functional correctness. The results, presented in Fig. 2, reveal several key findings.

All models can generate Cypher queries that return correct results for the given prompts, demonstrating a baseline level of overall correctness. When examining performance, CodeLlama, Qwen, and Granite show similar performance, with a $pass@20$ above 80%, with Qwen being the most practical (higher $pass@1$ score). DeepSeek, however, stands out with superior performance, surpassing 85% in $pass@20$.

In terms of schema evaluation, the LLMGen3 schema format provides the best overall results for CodeLlama, Qwen, and Granite, while DeepSeek achieves its best results when using the CSV schema format. Additionally, DeepSeek converges faster than the other LLMs, achieving the same $pass@k$ score with fewer iterations. Based on these results, we select Qwen and DeepSeek for further study as the most suitable for generating Cypher queries.

C. Evaluate Best Neo4J Schema Format and LLMs with CypherEval

Based on the above findings, we evaluate the ability of DeepSeek and Qwen to generate correct IYP Cypher queries by generating 100 Cypher queries per English prompt using LLMGen3 schema format for Qwen and CSV schema format for DeepSeek. This results in a total of 66,400 Cypher queries, which we evaluate using the $pass@k$ metric (Fig. 3).

Our analysis reveals that Qwen significantly outperforms DeepSeek on prompts classified as easy. When prompts include technical details, Qwen achieves a $pass@20$ above 55%, whereas DeepSeek scores below 45%. Even for more general prompts, Qwen maintains a $pass@20$ above 25%,

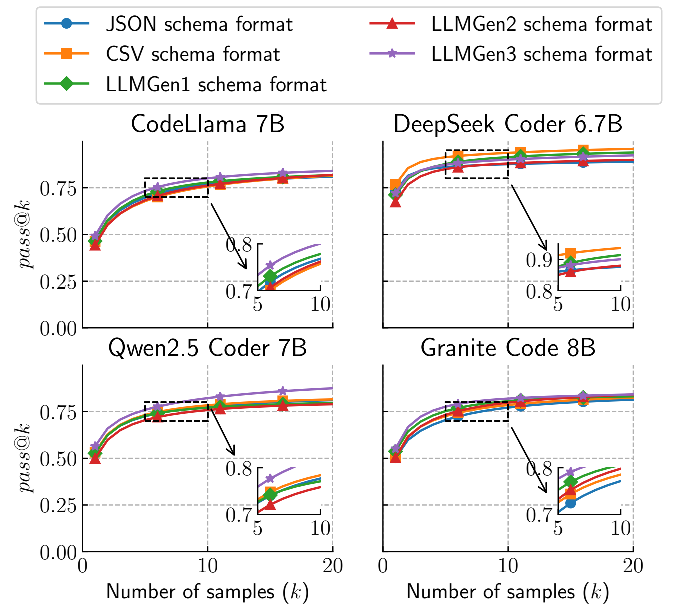

Fig. 2. $pass@k$ vs k plot for the Movies database. The figure shows the performance of the CodeLlama, DeepSeek, Qwen, and Granite using the different schemas and zooms in on the first 20 generated responses to better identify the optimal schema.

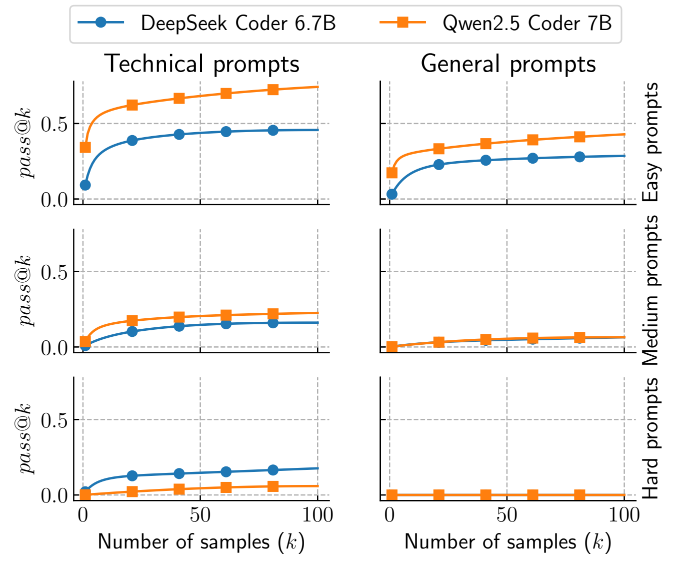

Fig. 3. $pass@k$ vs k plot for the IYP database using CypherEval (both variations have similar results). The figure shows the performance of DeepSeek and Qwen using the CSV and LLMGen3 schema format.

while DeepSeek falls below this threshold. This contrasts with DeepSeek's performance on the Movies database (§III-B), where it achieved $pass@20$ above 85%.

The level of detail in the prompt also plays a crucial role in model performance. For prompts that include technical details, such as references to specific nodes and relationships, the performance of both models improves. Qwen, in particular, reaches a $pass@80$ score of up to 75% for easy prompts. This indicates that unlike the Movies database, where LLMs performed well regardless of prompt style, providing explicit schema information in the prompt becomes essential when dealing with new databases.

However, both models face significant challenges when dealing with prompts of medium and hard difficulty. The majority of generated Cypher queries for these categories fail to produce correct results. This difficulty is amplified in the absence of training exposure to the target database. While the LLMs handle varying difficulty levels prompts successfully in the Movies database, their inability to do so for IYP indicates a strong dependence on prior familiarity with database structure and content.

These observations highlight a notable contrast in model behavior across databases. Qwen demonstrates relative proficiency in generating accurate Cypher queries for easy, well-described prompts in unfamiliar databases, which is promising for our end goal. DeepSeek, despite its strong performance in a known database, underperforms in the IYP, with $pass@k$ below 50% regardless of prompt difficulty. Together, these findings underscore the importance of schema familiarity and prompt detail in guiding LLMs to generate functionally correct Cypher queries.

IV. Pythia

In order to improve LLMs performance and provide a practical solution to IYP users, we introduce Pythia an open-source system[4] for generating IYP Cypher queries. Pythia combines few-shot prompting with a pre-trained LLM for code generation. This approach is particularly useful in highly specialized contexts or when dealing with domain-specific terminology that may not be well-represented in the general-purpose training data of LLMs [18].

A. *Pythia's Architecture*

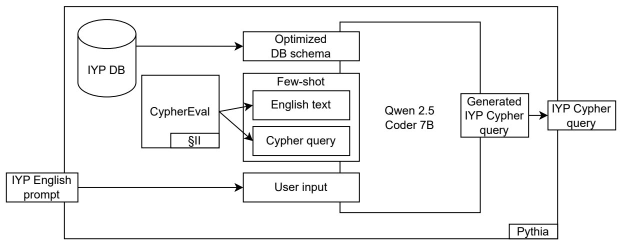

Fig. 4. Pythia architecture: Using CypherEval for few-shot prompting and Qwen for Cypher generation.

In Fig. 4, we present Pythia's architecture. Pythia is built on top of Qwen, and features a hidden input layer inaccessible to the end user, distinguishing it from other LLMs that allow users to access the entire input layer freely. This hidden layer utilizes an optimized IYP schema version and CypherEval. As a result, our system is designed exclusively for generating IYP Cypher queries from English text, with the user only able to

[4]https://codeberg.org/dimitrios/Pythia

provide a prompt related to the desired Cypher query for the IYP database. Pythia then returns the generated IYP Cypher query as output.

Optimized IYP Schema[1]. As discussed in §I, IYP combines various open-source Internet data datasets. To preserve all features from these datasets, such as the number of IPv4 prefixes, domain names, ASes, Facebook URLs associated with AS organizations, etc., IYP contains numerous node and relationship attributes. While this provides a wealth of information, it also introduces significant schema complexity. Furthermore, as noted in §III-B, longer prompts increase inference time and lead to hallucinations by the LLM [30].

Notably, not all attributes are necessary for querying the database, as the query results remain consistent regardless of their presence. For example, Facebook URLs from PeeringDB are unnecessary for Internet data analyses. To optimize the schema, we identify and remove attributes from nodes and relationships that do not influence the query outcomes. This reduction results in smaller LLM prompts, which improve inference speed and reduce the likelihood of hallucinations.

Throughout this refinement process, we collaborated with the authors of IYP to ensure that the new, compact, and task-focused schema maintains the integrity of the query results. It is worth noting that this optimization must be repeated whenever the IYP schema changes. The process cannot be fully automated, as future updates to the open-source Internet data datasets may introduce new features whose relevance to query results cannot be determined in advance.

CypherEval As Few-shot Examples. Few-shot prompting is an efficient alternative to LLM post-training [18]. Rather than updating LLM weights, we provide a curated set of input-output examples during inference time to guide LLM's behavior. As introduced in §II, we developed CypherEval, a real-world dataset for evaluating Cypher query generation from English text. Since CypherEval is based on the IYP schema, it serves as an ideal source of examples for our system. By including one CypherEval variation into our hidden layer, we provide the model with domain-specific knowledge.

B. Evaluation

To evaluate Pythia's effectiveness, we conduct leave-one-out cross-validation using CypherEval. Since this dataset is part of the hidden layer, we remove CypherEval's Cypher queries and their corresponding English text (both technical and general prompts) from the hidden layer that have high similarity with the ones that we intend to test. This ensures that Pythia has no prior knowledge of the Cypher query to be generated from CypherEval, allowing a fair evaluation.

When compared to Qwen (Fig. 5), we observe a significant improvement in $pass@k$ score when using Pythia. In terms of overall correctness, Pythia achieves notably higher $pass@1$ scores for easy, medium, and hard IYP queries. For general prompts of easy difficulty, Pythia reaches up to 75% $pass@20$, outperforming Qwen, which only reaches 55%. This suggests a more robust generalization capability in Pythia, particularly for easier queries.

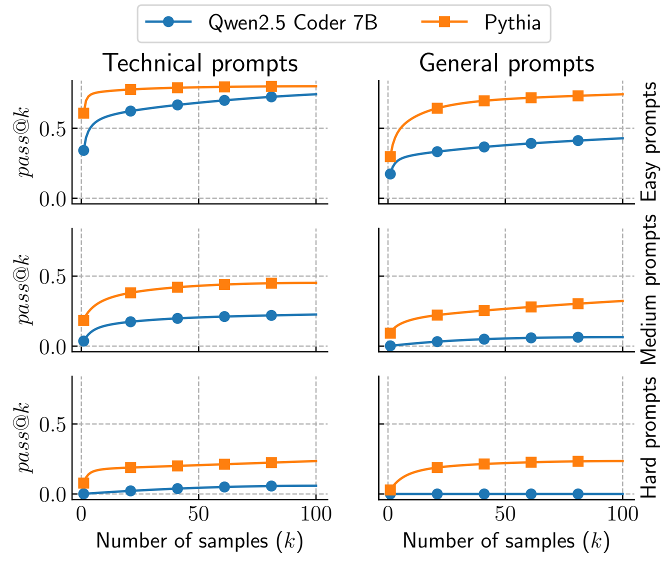

Fig. 5. Prompts' types vs $pass@k$ vs k plot for the CypherEval database. The figure shows that Pythia has better overall correctness for IYP Cypher queries generation.

In addition to better correctness, Pythia demonstrates faster convergence than Qwen. It is able to generate accurate IYP Cypher queries with fewer sampling iterations, making it more practical and efficient for real-world applications where compute cost is a concern.

However, despite this improvement, generating correct Cypher queries for medium and hard difficulty prompts remains challenging. For medium prompts, although most generated queries still fail to return correct results, Pythia achieves an overall $pass@k$ score close to 50%, outperforming Qwen and showing better reliability when technical details are included in the prompt. For hard prompts, the task remains difficult, but Pythia's performance still improves with an overall $pass@k$ score of 25%, whereas Qwen's performance remains significantly lower.

These findings show Pythia's enhanced ability to generate IYP Cypher queries. While challenges persist with complex queries, Pythia consistently outperforms Qwen across different levels of query difficulty.

V. Limitations and Discussion

Pythia can streamline workflows by eliminating the need to understand the underlying IYP structure or stay updated with schema changes. However, like other LLM based systems, using Pythia requires cautions from the users. It does not always generate IYP Cypher queries that align with the user's intent, and it may produce misleading or incorrect results after query execution on IYP. By discussing the risks associated with using Pythia, we aim to highlight areas of concern.

Over-reliance. A recent survey [31] showed that users increasingly tend to over-rely on LLMs, often accepting their responses without validation, which poses a significant risk. In Pythia's case, IYP Cypher queries will be generated using correct Cypher syntax and node and relationship attributes that align with the IYP schema. However, executing these queries may yield unexpected or incorrect results (e.g., nodes can exist in the IYP schema but can be irrelevant to the prompt). This can have unintended consequences, particularly for users exploring IYP for the first time, such as students or enthusiasts entering the networking domain.

Misalignment. Pythia generates Cypher queries based on the distribution of examples encoded in its hidden layer. As a result, if a user attempts to generate a highly specialized IYP Cypher query that is not similar to these examples, Pythia may produce a query that appears correct but is actually wrong.

Also, users might attempt to improve results by providing their own IYP example as part of their input. While this seems reasonable, since Pythia relies on example-based learning, there is a crucial distinction. The examples embedded in Pythia's hidden layer, as discussed in §II, have been reviewed by experienced IYP users and are considered error-free. In contrast, user-provided examples are not validated and may introduce errors, which can lead to the generation of incorrect queries that still appear accurate.

Risk Mitigation. While systems like Pythia offer valuable capabilities, it is important to evaluate the potential consequences of incorrect responses. It must be remembered that Pythia is a system and does not possess critical thinking skills. Its outputs are influenced by the training distribution of its base model (with respect to Cypher proficiency) and the IYP examples in its hidden layer (for IYP database expertise). Future work should explore the extent of the risks discussed above and how they correlate with the system's architecture.

VI. Related Work

Inspiring Work. Chen et al. [17] introduced Codex, an LLM capable of generating Python code from natural language. To evaluate Codex, they developed HumanEval, a dataset specifically designed to assess the functional correctness of LLMs that produce Python code. This dataset is also used to compare Codex with other LLMs unfamiliar with the dataset.

Zhong et al. [24] developed an LLM that generates Cypher queries from natural language. They presented a methodology that includes a phase for creating a prompt template that utilizes a Neo4J database schema format. They evaluated their results based on accuracy metric and emphasized the importance of extensive exploration to identify efficient prompt templates, as a prompt template can have numerous variations, such as different instructions, tones, etc.

LangChain Tool. LangChain [32] combines LLMs with various components like datasets and APIs, enabling the creation of various natural language applications. While it has been successfully used for tasks like translation, summarization, and Q&A, this tool relies on existing LLMs that are not trained for a specific task.

NLP Models. Existing NLP models trained specifically for Cypher query generation are available in open-source repositories [33]–[36]. However, these models are limited to generating queries for simple schemas or existing Neo4J databases. They struggle with generating accurate Cypher queries for IYP due to the database's specialized and complex schema, which is enriched in networking terminology.

Few-shot Prompting. Few-shot prompting is a method for improving the performance of LLMs by including a small set of examples in the model's prompt [18]. Results [18], [37], [38] showed that optimizing the prompt alone, without any additional training, can significantly improve the performance of LLMs, offering an alternative approach to optimization using only text-based prompts.

Uniqueness Of Our Work. Our work differs from existing studies by introducing CypherEval, the first real-world Cypher dataset designed for evaluating LLMs and benchmarking LLMs' performance in Cypher query generation. Furthermore, we propose Pythia to address the unique challenges of the IYP database. While training an LLM is a common practice in the literature, Pythia utilizes few-shot prompting to adapt to the IYP database without the need for extensive LLM training. Lastly, although we do not use LangChain in this research, as it is not essential for our current objectives, our findings can inform the integration of the best-performing LLMs or Pythia into LangChain for developing future tools.

VII. Conclusion

This paper investigates the generation of IYP Cypher queries from English text using LLMs to assist users, such as network operators, who want to use IYP but are unfamiliar with Cypher. We introduce CypherEval, a real-world dataset used in our benchmarking methodology to benchmark LLMs for IYP. Our results highlight that LLMs performance is highly related to the difficulty of the prompt, as they could not generate functionally correct Cypher queries for difficult prompts. We propose Pythia, a system that uses few-shot prompting, which significantly improves Cypher query generation for the IYP database.

Our study could be extended to Cypher-like querying languages. This opens a space for assisting policymakers and regulators unfamiliar with other related tools, and highlights the potential for LLMs to facilitate the use of complex querying languages in various domains.

Acknowledgment

We thank the Cloud Division - Cloud Infrastructure of Internet Initiative Japan Inc. for lending us an AI server, and Pierre-Louis Aublin for providing us an AI workstation to run our experiments for this research.

References

[1] D. Freedman, *The internet of rules*. Routledge, Feb. 2016, p. 28. [Online]. Available: http://dx.doi.org/10.4324/9781315695624-4
[2] CAIDA, "The CAIDA Dataset," https://www.caida.org/data/, 09 2024.
[3] R. NCC, "Routing Information Service (RIS)," https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/, 09 2024.
[4] U. of Oregon, "RouteViews Project," https://www.routeviews.org/routeviews/, 09 2024.
[5] R. NCC, "RIPE Atlas," https://atlas.ripe.net/, 09 2024.
[6] PeeringDB, "PeeringDB Dataset," https://www.peeringdb.com/, 09 2024.
[7] IHR, "IHR Archive," https://ihr-archive.iijlab.net/, 09 2024.

[8] R. Fontugne, M. Tashiro, R. Sommese, M. Jonker, Z. S. Bischof, and E. Aben, "The Wisdom of the Measurement Crowd: Building the Internet Yellow Pages a Knowledge Graph for the Internet," in *Proceedings of the 2024 ACM on Internet Measurement Conference*, ser. IMC '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 183–198. [Online]. Available: https://doi.org/10.1145/3646547.3688444

[9] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, "Cypher: An Evolving Query Language for Property Graphs," in *Proceedings of the 2018 International Conference on Management of Data*, ser. SIGMOD/PODS '18. ACM, May 2018. [Online]. Available: http://dx.doi.org/10.1145/3183713.3190657

[10] H. Naveed, U. Asad Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A Comprehensive Overview of Large Language Models," 07 2023. [Online]. Available: https://arxiv.org/pdf/2307.06435.pdf

[11] L. Chen, Q. Guo, H. Jia, Z. Zeng, X. Wang, Y. Xu, J. Wu, Y. Wang, Q. Gao, J. Wang, W. Ye, and S. Zhang, "A Survey on Evaluating Large Language Models in Code Generation Tasks," 08 2024. [Online]. Available: https://arxiv.org/pdf/2408.16498.pdf

[12] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, "A Survey on Evaluation of Large Language Models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, p. 45, Mar. 2024. [Online]. Available: http://dx.doi.org/10.1145/3641289

[13] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li *et al.*, "DeepSeek-Coder: When the Large Language Model Meets Programming–The Rise of Code Intelligence," *arXiv preprint arXiv:2401.14196*, 2024.

[14] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu *et al.*, "Qwen2. 5-coder technical report," *arXiv preprint arXiv:2409.12186*, 2024.

[15] M. Mishra, M. Stallone, G. Zhang, Y. Shen, A. Prasad, A. M. Soria, M. Merler, P. Selvam, S. Surendran, S. Singh *et al.*, "Granite code models: A family of open foundation models for code intelligence," *arXiv preprint arXiv:2405.04324*, 2024.

[16] S. Kulal, P. Pasupat, K. Chandra, M. Lee, O. Padon, A. Aiken, and P. Liang, "SPoC: Search-based Pseudocode to Code," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019, p. 12.

[17] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating Large Language Models Trained on Code," 07 2021.

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020, p. 25.

[19] L. Reynolds and K. McDonell, "Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 7.

[20] (2023, 11). [Online]. Available: https://github.com/neo4j/NaLLM

[21] J. Peng, W. Yang, F. Wei, and L. He, "Prompt for extraction: Multiple templates choice model for event extraction," *Knowledge-Based Systems*, vol. 289, p. 111544, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705124001795

[22] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala, "Overview and Importance of Data Quality for Machine Learning Tasks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. Association for Computing Machinery, Aug. 2020. [Online]. Available: http://dx.doi.org/10.1145/3394486.3406477

[23] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A Review of Machine Learning Interpretability Methods," *Entropy*, vol. 23, no. 1, p. 18, Dec. 2020. [Online]. Available: http://dx.doi.org/10.3390/e23010018

[24] Z. Zhong, L. Zhong, Z. Sun, Q. Jin, Z. Qin, and X. Zhang, "SyntheT2C: Generating Synthetic Data for Fine-Tuning Large Language Models on the Text2Cypher Task," 06 2024.

[25] C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters," 2024. [Online]. Available: https://arxiv.org/abs/2408.03314

[26] (2024, 09). [Online]. Available: https://github.com/neo4j-graph-examples/movies

[27] (2024, 09). [Online]. Available: https://neo4j.com/docs/getting-started/appendix/tutorials/guide-build-a-recommendation-engine/

[28] W. W. Baraki, "Leveraging large language models for accurate Cypher query generation : Natural language query to Cypher statements," Master's thesis, , School of Informatics, 2024.

[29] B. Rozière, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, R. Sauvestre, T. Remez, J. Rapin, A. Kozhevnikov, I. Evtimov, J. Bitton, M. Bhatt, C. Canton Ferrer, A. Grattafiori, W. Xiong, A. Défossez, J. Copet, F. Azhar, H. Touvron, L. Martin, N. Usunier, T. Scialom, and G. Synnaeve, "Code Llama: Open Foundation Models for Code," 08 2023. [Online]. Available: https://arxiv.org/pdf/2308.12950.pdf

[30] S. Liu, K. Halder, Z. Qi, W. Xiao, N. Pappas, P. M. Htut, N. A. John, Y. Benajiba, and D. Roth, "Towards Long Context Hallucination Detection," 2025. [Online]. Available: https://arxiv.org/abs/2504.19457

[31] H.-P. H. Lee, A. Sarkar, L. Tankelevitch, I. Drosos, S. Rintel, R. Banks, and N. Wilson, "The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers," ser. CHI '25. New York, NY, USA: Association for Computing Machinery, 2025. [Online]. Available: https://doi.org/10.1145/3706598.3713778

[32] J. Jeong, D. Gil, D. Kim, and J. Jeong, "Current Research and Future Directions for Off-Site Construction through LangChain with a Large Language Model," *Buildings*, vol. 14, no. 8, 2024. [Online]. Available: https://www.mdpi.com/2075-5309/14/8/2374

[33] (2024, 4). [Online]. Available: https://github.com/Chirayu-Tripathi/nl2query

[34] (2024, 6). [Online]. Available: https://huggingface.co/lakkeo/stable-cypher-instruct-3b

[35] (2024, 4). [Online]. Available: https://huggingface.co/vickarrious/code-llama-7b-text-to-cypher

[36] (2023, 8). [Online]. Available: https://huggingface.co/diegomiranda/text-to-cypher

[37] T. Gao, A. Fisch, and D. Chen, "Making Pre-trained Language Models Better Few-shot Learners," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, 2021, pp. 3816–3830. [Online]. Available: https://aclanthology.org/2021.acl-long.295/

[38] X. Ye and G. Durrett, "The Unreliability of Explanations in Few-shot Prompting for Textual Reasoning," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 30 378–30 392. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/c402501846f9fe03e2cac015b3f0e6b1-Paper-Conference.pdf