Short Paper

Graph Consistency Rule Mining with LLMs: an Exploratory Study

Hoa Thi Le
Lyon1 University, CNRS Liris
Lyon, France
hoalt.ptit@gmail.com

Angela Bonifati
Lyon1 University, CNRS Liris & IUF
Lyon, France
angela.bonifati@univ-lyon1.fr

Andrea Mauri
Lyon1 University, CNRS Liris
Lyon, France
andrea.mauri@univ-lyon1.fr

ABSTRACT

Graph data structures are essential for representing complex relationships in various domains, including life sciences, social media, healthcare, finance, security, and planning. With the increasing reliance on graph databases—particularly property graphs—for capturing semantic relationships, ensuring data integrity and quality has become crucial. Traditional methods for maintaining consistency, such as expert-defined rules and data-mined constraints like functional and entity dependencies, face challenges in scalability, adaptability, and comprehensibility. In this paper, we explore how Large Language Models (LLMs) can be utilized to automatically generate and refine consistency rules for property graphs through guided prompts. Leveraging the reasoning capabilities of LLMs over expressive graph models, we conduct an exploratory empirical study to assess the extent to which LLMs can extract rules that enforce data consistency. Our evaluation spans different real-world datasets and various graph encoding methods. Our results demonstrate that LLMs show promising abilities in extracting consistency rules, primarily identifying schema-based constraints such as primary keys, attribute uniqueness, and label enforcement. Additionally, LLMs occasionally capture more complex patterns, including temporal constraints where certain events cannot occur simultaneously.

KEYWORDS

Property Graphs, Rule Mining, Large Language Models

1 INTRODUCTION

Graph data has become widespread in various domains [24] such as life sciences, social media, healthcare, finance, security, and planning due to its ability to represent complex relationships between entities. With the growing reliance on graph data, ensuring data integrity and quality has become essential. Graph databases leveraging property graphs have been extensively adopted to capture the semantics of these complex relationships. However, ensuring consistency within large evolving property graphs is challenging. One of the approaches to ensure the quality of the graph is through consistency rules, such as functional dependencies [13] and entity dependencies [11]. These rules help maintain data integrity by enforcing specific constraints and relationships among the entities.

For example, consider a graph representing a social media platform like Twitter, where users, tweets, and hashtags are represented as nodes and edges represent relationships such as mentions, posts, follows, or tags. A consistency rule in this context could enforce temporal constraints—for instance, a retweet can occur only after the original tweet has been posted. Another rule might ensure that users cannot follow themselves or that every tweet must be associated with a valid user who posted it. These rules are crucial for maintaining the logical consistency of the data and preventing anomalies that could affect analytics and user experience.

Traditionally, these rules are either provided by domain experts [20], reflecting business logic related to the data, or mined directly from the data by considering the co-occurrence of elements [17, 26]. However, both approaches have limitations. Expert-defined rules may not cover all edge cases or adapt quickly to new data patterns, while data-mined rules can generate an overwhelming number of constraints, some of which may be redundant, irrelevant, or difficult to understand by the domain expert.

Given the recent advent of Large Language Models (LLMs), many works have started to investigate their capabilities to reason over structured data, both relational and graphs. In the context of graph data, LLMs have shown promising results in basic graph computational tasks on simple labeled graphs [14], graph mining [16], and reasoning [6]. They also enable non-experts to interact with these rich data structures through conversational interfaces [21].

For these reasons, in this paper, we explore how Large Language Models can be used to automatically generate and refine rules for property graphs by guiding them through designed prompts. By leveraging the capabilities of LLMs to understand and reason about graph data structures, we aim to provide an intuitive method to maintain data integrity in graph databases.

In this paper, we contribute with a **exploratory study** to investigate **to what extent** LLMs can **reason** over expressive graph models - such as property graph - to extract rules that can be used to enforce consistency in the data. To this end, we perform an **empirical study** evaluating how different LLMs perform in extracting consistency rules, on various real-world datasets using different graph encoding methods. Our preliminary results show that LLMs have promising capabilities in extracting consistency rules, mainly consisting of schema-based constraints (e.g, primary keys, uniqueness of attributes, or forcing specific node or edge labels), but sometimes also extracting constraints related to more complex patterns or considering the temporality of data (e.g., two events cannot happen simultaneously).

2 RELATED WORK

Rules Mining. Rule mining has been widely studied, especially in the context of knowledge bases (KBs) and relational databases [3]. AMIE [15, 17] is one of the most widely used systems to mine rules in large knowledge bases, such as DBpedia and Wikidata. It uses various optimization and pruning techniques that allow it to find exact rules without needing to estimate or sample. Specifically for property graphs, Cambria et al., [6] introduce operator embeds Cypher that works with Neo4j to process queries

efficiently. On the other hand, Xu et al., [26] use a transformer-based architecture to encode subgraph structures and generate reasoning rules, treating the rule mining process as a sequence generation task. Fan et al., [12] introduces the concept of Graph Association Rules to find relationships between elements in a graph. The proposed algorithms in this work focus on scalability and efficiency, providing polynomial speedups over traditional methods. While the previous methods are relevant to this work, they focus on either association or logical rules. However, in this paper, we are interested in mining *consistency rules* - i.e., rules enforcing quality constraints over data.

LLMs for Graph Processing. In recent years, the integration of Large Language Models with graph processing has gained significant attention. These models, originally designed for natural language tasks, are being adapted to enhance various graph-related applications. Ren et al., [23] provides a detailed overview of the current state-of-the-art on the use of LLMs in graph learning. The authors classify current methods into frameworks such as directly combining LLMs with graph tasks or prefixing them with Graph Neural Networks (GNNs). While some frameworks effectively leverage GNNs for tasks like link prediction, others struggle with data sparsity and generalization. The study also identifies promising avenues for future research, such as creating multimodal LLMs that can handle different kinds of data, which might greatly improve reasoning powers over graphs. Guo et al., [16] explore the capacity of LLMs to comprehend graph-structured information. The empirical study shows that although LLMs seem promising, there are still obstacles in their way when it comes to efficiently analyzing the relational and multi-dimensional nature of graph data. This poses significant issues on how to improve LLMs' comprehension and manipulation of network structures, especially in situations that call for complex thinking. Peng et al., [21] offers a compelling perspective on user interaction with graphs. By allowing users to query graphs through natural language, this LLM-based framework significantly lowers the barrier to entry for graph analysis. Traditional methods often require specialized programming skills or knowledge of query languages, but this work shows LLMs can facilitate a more intuitive interaction model. This democratization of graph analysis is particularly valuable, as it opens up opportunities for non-experts to engage with complex data structures. Another relevant study is [18]. This research presents a framework for using LLMs to generate logical rules based on the semantics and structure of knowledge graphs (KGs). The current approaches, which frequently suffer from computational inefficiencies and a lack of scalability, have a fundamental gap that our study fills. The proposed method allows for efficient rule generation and ranking, enhancing the interpretability and scalability of reasoning tasks on KGs. This has important implications for improving the interpretability of reasoning tasks within KGs, suggesting that LLMs might be key to unlocking deeper insights into knowledge representation. LLMs and conventional methods have been applied in a number of works that have investigated rule mining. However, most research employing LLMs in graph-based rule mining [18] has focused primarily on knowledge graphs, with an emphasis on generating logical rules and reasoning based on relationships between entities. Nonetheless, there has yet to be any comprehensive research exploring the potential of LLMs for rule mining in property graphs. This highlights a significant research gap and an opportunity to investigate the capabilities of LLMs in automating and enhancing rule-mining processes for this type of graph.

3 LARGE LANGUAGE MODEL FOR RULE MINING

A property graph [5] is a flexible and expressive way to represent complex data involving entities and the relationships between them. In this data model, both nodes and edges can have multiple labels, which are descriptive tags indicating the type or category of nodes and edges. In addition to labels, nodes, and edges can have properties, which are key-value pairs storing additional information.

Defining property graph consistency rules and translating them into Cypher queries can be a complex task requiring great human effort. Our research investigates using large language models to automate the generation and conversion of these rules. Figure 1 shows a pipeline to mine consistency rules from a property graph using an LLM. First, the property graph G is encoded into a text format that LLMs can understand, and then it is input along with a prompt. The latter is specifically crafted to ask the LLM to infer the consistency rules. Based on the structure and relationships of the graph, the LLM generates the consistency rules in natural language. In the second step, natural language rules are converted into Cypher queries. This two-step procedure can ensure clarity to those may not be familiar with Cypher [22, 25]. In the following, we detail each step of the pipeline.

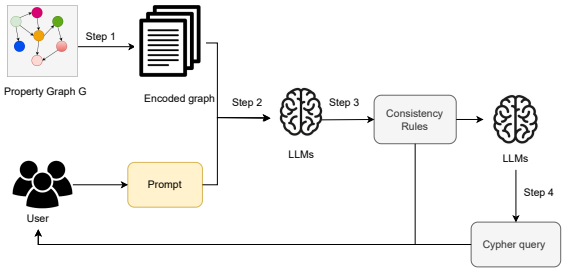

Figure 1: Overview of the pipeline.

3.1 Encoding the graph

The first step in our pipeline is to encode the property graph. Unlike typical encoding methods in deep learning models that convert data into numerical vectors, the graph is represented in text format that LLMs can understand in the prompt (First step - Figure 1). This encoding must ensure that the structure and important information of the property graph are preserved. In this study, we use the *incident encoder* based on its demonstrated effectiveness in prior research [14]. However, when using LLMs to generate consistency rules for property graphs, the size of the prompt of LLMs becomes an important challenge. To overcome the input size limitation, we propose two methods: sliding window attention and retrieval augmented generation (RAG).

3.1.1 Sliding Window Attention. As shown in Figure 2(a), after the graph has been converted into text, the text-encoded graph is divided into smaller sections called *windows*. In our experiments, we empirically chose a window size to reduce the probability of a pattern exceeding the windows. For example, when dividing a graph into windows, the last part of a window might contain the text *"Node node_id,"* while the next starts with *"with label Label has properties (key: value)"* (i.e., the node element is broken in two strings). Without overlap, such boundary splitting can lead to a loss of context, making it challenging for LLMs to fully understand the data. Overlapping between sections is necessary to ensure that the meaning of the data at the boundaries of the

windows is not lost. For this reason, we set the window size and the overlap as the maximum allowed by the LLMs limit, that is 8000 tokens for the window size, and 500 tokens overlap [4]. After the encoded windows have been created, each window is paired with a specific prompt that guides the LLMs on how to generate consistency rules. This prompt acts as an instruction, helping the LLMs focus on important aspects of the graph within each window. Next, the encoded windows along with the prompts are fed into the LLM for processing. Based on the information from the windows and the instructions from the prompts, the LLM generates consistency rules for the corresponding part of the graph. Finally, the rules generated from each window are combined to create a comprehensive set of rules that apply to the entire graph.

As stated before, this method may have limitations related to the fragmentation of context, which may lead to the loss of global information and boundary issues between windows, which could cause conflicts or omissions in the generated rules, a challenge that could be mitigated in future work by employing advanced NLP techniques to detect and merge patterns across windows effectively.

3.1.2 Retrieval Augmented Generation (RAG). As illustrated in Figure 2(b), the RAG framework combines information retrieval with text generation to enhance the accuracy and consistency of rule generation. While the graph is initially encoded as text in Step 1 (Figure 1), in this case, it undergoes a further transformation into vector embeddings. These embeddings are a critical component of the RAG method (Step 2 - Figure 2.b), as they enable storage in a vector database and retrieval of information. For this transformation, we utilized GPT4AllEmbeddings from the langchain_community library, ensuring high-quality embeddings tailored to the task. The prompting process occurs in two phases. In the first phase, a prompt requesting consistency rules directs the LLM to retrieve the most pertinent parts of the graph from the vector database. In the second phase, the LLM generates the rules by utilizing the graph data retrieved from the database.

3.2 Prompt Design

Prompts play an important role in guiding the LLM to generate relevant and accurate rules. In this paper, we used two types of prompts to generate consistency rules for property graphs: *zero-shot* and *few-shot prompts*. Each approach serves a different purpose and impacts how the LLMs generate the rules.

In the zero-shot method, shown in Figure 3(a), the LLM receives the encoded graph and instruction to generate consistency rules (in terms of graph functional and entity dependency rules). With the *few-shot* methods - shown in Figure 3(b) - instead, the LLM is also provided with rule examples. Once the consistency rules are generated, the LLM is prompted to generate the corresponding Cypher queries. The prompt included generated rules and information about the property graph including nodes edge labels, and properties, and asked the LLM to write the Cypher query matching the rule in natural language.

4 EXPERIMENTS

In the experimental study, we evaluate the capability of LLMs in generating consistency rules considering the two methods to encode the graph (Sliding Attention Window and RAG), and the two types of prompting (zero-shot and few-shot). To explore the ability to generate consistency rules for property graphs, we use `Mixtral` [7] and `LLaMA-3` [19]. We opted for these models because they are open-source large language models and can be deployed locally, unlike other non-open-source LLMs.

We implemented the project using Python 3.10 and Neo4j was used for the underlining graph database operations. The experiments were executed on a MacBook M2 with 16GB of RAM and a 512GB SSD. All the code to replicate the experiments is available at this GitHub repository [1]

4.1 Datasets

	Nodes	Edges	Node Labels	Edge Lagels
`WWC2019`	2468	14799	5	9
`Cybersecurity`	953	4838	7	16
`Twitter`	43325	56493	6	8

Table 1: Size of the datasets in term of number of nodes, edges, and labels

We conduct experiments on three different property graphs, summarized in Table 1. `WWC20219` [8]: This graph depicts information related to the 2019 Women's World Cup, including nodes such as teams, persons, matches, tournaments, squads, and the relationships between them. `Cybersecurity` [9]: This graph is related to cybersecurity, including systems, security events and their interconnections. It represents an active directory environment with users, groups, domains, policies, and computers. `Twitter` [10]: This dataset represents the interaction between users on the Twitter social network. Nodes represent entities such as users, tweets, hashtags, links, and sources, while edges represent various interactions and relationships between these entities.

4.2 Metrics

To evaluate the effectiveness of the consistency rules generated for property graphs we use some ranking measures used in the state-of-the-art of rules mining [15], **support**, **coverage** and **confidence**, and adapted it for property graph. **Support** measures the number of elements in the graph that satisfy a given rule. A higher support indicates that the rule is applicable to more facts. **Coverage** evaluates the proportion of facts related to the rule's head relation that are covered by the rule. It normalizes the support by the total number of facts for the relation in question. **Confidence** assesses the reliability of the rule by comparing the number of facts that satisfy the rule to the number of times the rule's body conditions are met. This measure indicates the rule's accuracy and how often the rule leads to the expected outcomes. The metrics for a given rule were computed by executing the corresponding Cypher query generated by the LLM, as explained in Section 3.2. For the Cypher queries that were generated inaccurately, we manually corrected them to ensure an accurate evaluation of the generated rules. More details in Section 4.4.

4.3 Rules Generation

In this section, we present the results in terms of the metrics obtained by the rules generated by the LLMs. For space reasons, the complete list of rules extracted by the LLMs is reported in the supplementary materials [2]. Tables 2, 3 and 4 report the score after correcting the Cypher queries. The details on how the correction was done are described in Section 4.4.

The results from applying consistency rule generation methods on the `WWC2019` property graph dataset reveal a clear distinction between the models `LLaMA-3` and `Mixtral`, as shown in

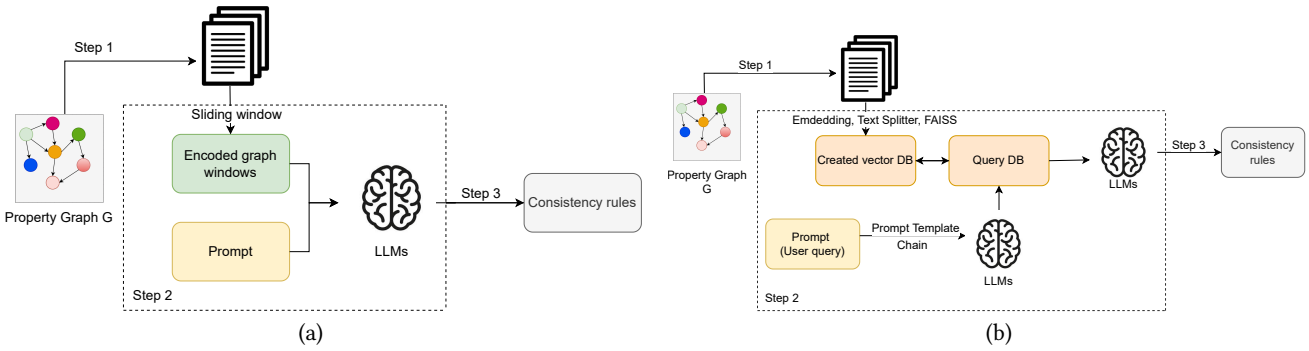

Figure 2: Sliding Window Attention Flow (a) and RAG (b)

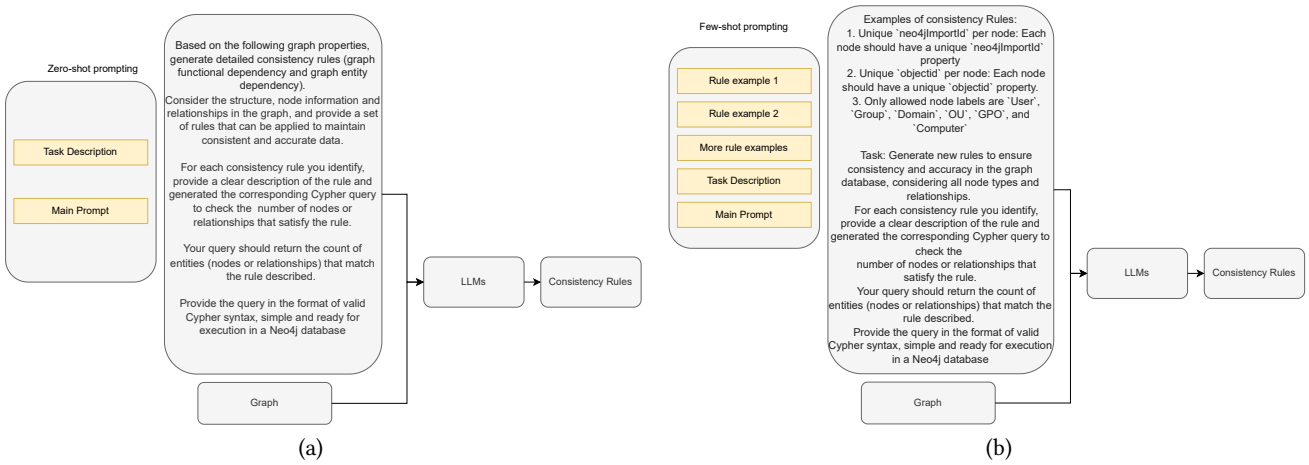

Figure 3: Zero-shot (a) and Few-shot prompting (b)

	Slide Window Attention				RAG			
	#rules	Supp%	Cov%	Conf%	#rules	Supp%	Cov%	Conf%
Zero-shot								
Llama-3	12	513	98.67	98.69	7	1604	100	91.57
Mixtral	9	1257	97.81	89.5	6	923	73.75	67.83
Few-shot								
Llama-3	8	716	92.22	100	6	1002	73.75	73.5
Mixtral	8	622	92.81	95.75	5	6364	63.12	80

Table 2: Support, coverage and confidence score for the `WWC2019` dataset with Zero-Shot and Few-Shot Prompts

	Sliding Window Attention				RAG			
	#rules	Supp%	Cov%	Conf%	#rules	Supp%	Cov%	Conf%
Zero-shot								
Llama-3	10	406	92.6	100	7	159	45.64	45.50
Mixtral	10	351	67.36	67.6	6	315	52.7	51.5
Few-shot								
Llama-3	9	1113	89.38	97.86	7	651	75.43	99.71
Mixtral	5	635	71.9	91.96	5	1501	99.96	100

Table 3: Support, coverage and confidence score for the `Cybersecurity` dataset with Zero-Shot and Few-Shot Prompts

Table 2. In the Zero-Shot approach, `LLaMA-3` performs better with higher scores for support, coverage, and confidence compared to `Mixtral` using both Slice Window Attention and RAG methods. Specifically, `LLaMA-3` achieves a support score of 513 with Sliding Window Attention and 1604 with RAG, coverage of 98.67% and 100%, and confidence of 98.69% and 91.57% respectively. In contrast, `Mixtral` shows lower support scores but demonstrates the ability to discover more interesting rules that consider more complex patterns, such as "A player should be associated with a squad, and that squad should belong to the tournament for which the player has played a match". In the Few-Shot approach, LLAMA-3 continues to excel with superior coverage and confidence, with support scores of 716 and 1002 for Sliding Window Attention and RAG respectively, coverage of 92.22% and 73.75%, and confidence of 100% and 73.5%. `Mixtral` achieves support scores of 622 and 6364 for the two methods, with coverage of 92.81% and 63.12% and confidence of 95.75% and 80%.

Table 3 presents the results for the `Cybersecurity` dataset. With Zero-Shot, `LLaMA-3` performs better with Sliding Window Attention, achieving a support of 406, coverage of 92.6%, and confidence of 100%, compared to `Mixtral`'s support of 351, coverage

	Sliding Window Attention				RAG			
	#rules	Supp%	Cov%	Conf%	#rules	Supp%	Cov%	Conf%
Zero-shot								
Llama-3	8	12177	72.27	86.14	8	981	70.62	78.75
Mixtral	10	10789	81.20	81.20	7	7698	67.3	76
Few-shot								
Llama-3	7	25201	85.72	85.72	9	8994	71.34	77.78
Mixtral	7	15262	78.79	83.25	8	11593	100	100

Table 4: Support, coverage and confidence score for the `Twitter` dataset with Zero-Shot and Few-Shot Prompts

of 67.36%, and confidence of 67.6%. However, with RAG, Mixtral shows a much higher support compared to LLaMA-3, though the latter still maintains a slightly better coverage (45.64%) and confidence (45.50%) than Mixtral. In the Few-Shot setting, LLaMA-3 continues to lead in terms of coverage and confidence using both Sliding Window Attention (89.38% coverage, 97.86% confidence) and RAG (75.43% coverage, 99.71% confidence). Mixtral, on the other hand, shows higher support with RAG but lower coverage and confidence compared to LLaMA-3. RAG shows slight improvements in the Few-shot scenario over Zero-shot. Table 4 shows the results for the Twitter dataset. LLaMA-3 outperforms Zero-shot in terms of support, coverage, and confidence compared to Mixtral. Meanwhile, both models demonstrate significant improvement in Few-shot. Specifically, Mixtral shows a great improvement with RAG, achieving an average of 100% of coverage and confidence. LLaMA-3 still dominates the other cases with coverage and confidence values ranging from 70% to 85%.

Rule Mining Time Analysis: We analyzed the time required for the LLMs to mine rules. Table 5 summarizes the results across all datasets, considering the two types of prompting and the methods used for encoding the graph. Our analysis revealed that, while the Sliding Window Attention encoding method produces better rules, it incurs significant mining times for larger graphs (e.g., Twitter), with times reaching approximately 500 seconds. This is because the Sliding Window Attention approach requires the LLM to be prompted multiple times, depending on the number of windows. We noticed that Few-Shot prompting increases the performance of the Sliding Window method in the case of larger graphs. In contrast, the RAG method offers substantial improvements, as the LLM is prompted only once with a partial representation of the graph. Future research on efficient rule mining with LLMs should focus on parallelizing the prompting process (e.g., distributing different parts of the graph to multiple LLMs) or developing methods to prompt a single LLM using subgraphs most relevant to the mining task.

Model	Sliding Window Attention		RAG	
	Zero-shot	Few-shot	Zero-shot	Few-shot
		WWC2019		
Llama-3	251.36	227.74	5.16	4.39
Mixtral	219.28	216.38	3.10	4.34
		Cybersecurity		
Llama-3	445.66	336.67	5.15	5.87
Mixtral	497.66	315.25	4.65	4.16
		Twitter		
Llama-3	525.25	410.20	3.67	4.24
Mixtral	517.32	392.18	3.54	4.17

Table 5: LLMs rule mining times (seconds)

4.4 Cypher Generation

In this section, we discuss the performances of the model in generating the Cypher queries related to the rules. We consider a **query** not correct if it has syntax errors or if its formulation does not match the data model. As shown in Table 6, both LLMs tend to correctly generate the queries (with a minimal accuracy of 70%). In addition, we also analyzed the types of errors the LLMs make and we found three categories. The first category corresponds to misinterpreting the direction of the relationship in the property graph. There were 5 cases where the LLMs misinterpreted the direction of the relationship. For example, with

Model	Sliding Window Attention		RAG	
	Zero-shot	Few-shot	Zero-shot	Few-shot
		WWC-2019		
Llama-3	11/12	7/8	7/7	5/6
Mixtral	8/9	7/8	5/6	4/5
		Cybersecurity		
Llama-3	8/10	7/9	6/7	7/7
Mixtral	7/10	4/5	4/6	4/5
		Twitter		
Llama-3	7/8	5/7	7/8	8/9
Mixtral	9/10	6/7	6/7	8/8

Table 6: Number of correctly generated Cypher queries

the rule *Unique Match identifier within a Tournament*, the correct query should ensure that each match within a tournament has a unique identifier. However, the LLM generated the following query:

```
MATCH (t:Tournament)-[:IN_TOURNAMENT]->(m:Match)
WITH t.id AS tournament_id, m.id AS match_id, COUNT(*) AS count
WHERE count = 1
RETURN COUNT(*) AS support;
```

This query incorrectly identifies the relationship direction from *Tournament* to *Match*. The second type of error correspond to misunderstand the properties between nodes or relationships, sometimes even creating properties that do not exist. For instance, the following rule generated by Mixtral *"Ensure that no two SCORED_GOAL relationships between a Person and a Match should have the same minute property"* resulted in the following Cypher query:

```
MATCH (p:Person)-[:SCORED_GOAL]->(m:Match)
WITH m.id AS match_id, p.id AS person_id,
COLLECT (DISTINCT p.name + ':'
 + tostring(m.score) + ':' + tostring(m.penaltyScore)
 + ':' + tostring(m.minute)) AS minutes WHERE Size(minutes) > 1
RETURN match_id, person_id, minutes;
```

However, this Cypher query is incorrect as it includes non-existent properties like *score*, *minutes*, and *penaltyScore* for the *Match* node. The third category of errors is related to syntax issues in the Cypher query. An example of this is when the rule *"The domain property should be a string value matching domain format"* generated the following incorrect query:

```
MATCH (n)
WHERE n.domain IS NULL AND n.domain = '^([a-zA-Z0-9-]+\\.)+
[a-zA-Z]{2,}$'
RETURN COUNT(*) AS valid_domains
```

In this case, the LLM used the "=" operator for the regular expression match, but the correct operator should have been "=~" to properly compare a string with a regular expression

In addition to these error categories, another factor contributing to the decrease in LLM performance is the generation of **inaccurate rules** (i.e., the rule itself is not correct). To ensure a fair evaluation of the LLM's ability to generate consistency rules, we corrected the queries in case of syntax errors or wrong edge directions, but we left them as they were the queries with additional non-existing properties, because those errors corresponded to hallucination at rule generation level, rather than the translation to Cypher.

4.5 Discussion

In our study, we initially aimed to extract specific GFD and GED rules. However, we observed that the LLMs struggled to distinguish between these concepts effectively. In general, for all the datasets the extracted rules seem to relate to the schema of the graph (e.g. enforcing that nodes are connected with edges having specific labels, or specifying some values for the properties). For instance, an example of a rule for the WWC20219 is *"Each match*

node should have a date and stage property" emphasizes the necessity of defining essential attributes for the integrity of the graph; while another, for the `Cybersecurity` one is *The owned property should only be True or False*. Even though Few-Shot prompting results in a higher confidence score, it doesn't seem to change the type of rules generated.

`LLaMA-3` generates rules with higher support, coverage, and confidence than `Mixtral`. However, this could be explained by LLM's tendency to focus on simple rules regarding the uniqueness of elements. For example, in the `Twitter` dataset, a generated rule is *Each tweet node should have a unique id property*.

In contrast, `Mixtral` appears to generate more complex rules. For instance, for the `WWC20219` dataset, it specifies that a player must be associated with a squad that belongs to the tournament in which the player has participated. Another notable rule indicates that each match must have a score for both teams if the score has been determined. Additionally, this rule stipulates that a player cannot score two goals in the same minute of the same match. This complexity could explain its lower scores, as there may be fewer elements in the graph satisfying these rules. Also, LLMs - as reported in Section 4.4 - may encounter issues when converting these complex rules into Cypher queries.

Interestingly, the sliding window method yields better results in capturing crucial information, even though there could be the risk of patterns being split in different windows (hence information relevant to mine rules could be lost). Probably this effect is attenuated by the fact that the LLM sees the whole graph. Also, we noticed that the number of patterns broken in this way was relatively small, 6 for the `WWC2019`, 11 for `Cybersecurity`, and 6 for `Twitter graph`.

On the other hand, the RAG encoding method does not perform as expected. This is probably because the LLM is not able to retrieve from the graph the specific information needed to extract the rules adapted for each dataset since the prompt itself indicates only the request to generate consistency rules. Incomplete or irrelevant context retrieval can significantly impact the performance of the LLM. The retriever might not obtain enough pertinent information from the vector database, leaving the LLM without the necessary context to generate accurate rules. Additionally, irrelevant chunks that appear semantically similar to the query may be retrieved, introducing excessive or irrelevant information. This noise can distract or mislead the LLM, leading to the generation of incorrect or unrelated rules. As a result, both the lack of relevant context and the presence of irrelevant information can undermine the rule-generation process. Using Few-Shot prompting improves the performance only in the `Twitter` dataset. Human intervention was necessary to address inaccuracies in the query generation while maintaining the intended meaning of the rules, in the future the correction of syntax errors may be automated. While the current pipeline is not fully automated, the LLMs nature has the opportunity to design rule mining pipelines that are inherently interactive, allowing also domain expert (who may not possess technical knowledge) to refine the rules to their needs.

5 CONCLUSIONS AND FUTURE WORK

In this research, we introduced a new method using LLMs to automatically generate consistency rules for property graphs, improving data quality. This method not only expands the scope of LLMs beyond knowledge graphs but also offers a new approach for mining consistency rules in property graphs.

While this work shows promising in using LLMs for rule mining in property graphs, it also highlights challenges related to LLMs limitations regarding input size and processing time, especially in case of bigger graph and prompts.

Although our approach is primarily designed for property graphs, it is also applicable to flat relational data. Relational data can be seen as a graph structure, especially when organized following key-foreign key relationships. In this case, nodes represent entities, and edges represent relationships between them. To apply our method to relational data, the rules and the encoding process using LLMs would need to be adapted accordingly.

Future work will focus on these key areas. First, a more detailed empirical study - with more complex prompting strategies - is required to thoroughly evaluate the performance of LLMs across different graph structures, which will provide deeper insights into rule generation patterns. Second, as mentioned in Section 4.3, we will investigate efficient rule mining methods, either based on parallelism or graph summarization. Third, developing interactive rule mining techniques could allow users to engage in the rule extraction process, offering real-time feedback to refine the rules. Finally, enabling LLMs to explain the rationale behind the rules they generate would improve transparency and provide valuable insights into the underlying data patterns.

REFERENCES

[1] 2024. *Code repository.* Retrieved October 08, 2024 from https://github.com/LeHoa98ptit/Rule-mining-with-llms

[2] 2024. *Supplementary materials.* Retrieved December 19 , 2024 from https://github.com/LeHoa98ptit/Rule-mining-with-llms/blob/main/Rules-Generated.pdf

[3] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (Washington, D.C., USA) *(SIGMOD '93)*. Association for Computing Machinery, New York, NY, USA, 207–216. https://doi.org/10.1145/170035.170072

[4] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[5] Angela Bonifati, George Fletcher, Hannes Voigt, and Nikolay Yakovets. 2018. Querying Graphs. In *Synthesis Lectures on Data Management*. Morgan & Claypool Publishers, Chapter 4, 03–13.

[6] Francesco Cambria, Francesco Invernici, Anna Bernasconi, and Stefano Ceri. 2024. MINE GRAPH RULE: A New Cypher-like Operator for Mining Association Rules on Property Graphs. *arXiv preprint arXiv:2406.19106* (2024). https://doi.org/10.48550/arXiv.2406.19106 Submitted on 27 Jun 2024.

[7] Albert Q. Jiang et al. 2024. Mixtral of Experts. *arXiv preprint arXiv:2401.04088* (2024). https://doi.org/10.48550/arXiv.2401.04088 Both the base and instruct models are released under the Apache 2.0 license.

[8] Neo4j Graph Examples. 2020. Women's World Cup 2019 Graph. In *https://github.com/neo4j-graph-examples/wwc2019/tree/main.*

[9] Neo4j Graph Examples. 2021. Cyber Security Graph. In *https://github.com/neo4j-graph-examples/cybersecurity.*

[10] Neo4j Graph Examples. 2021. Twitter-V2 Graph. In *https://github.com/neo4j-graph-examples/twitter-v2.*

[11] Wenfei Fan and Ping Lu. 2017. Dependencies for Graphs. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Chicago, Illinois, USA) *(PODS '17)*. Association for Computing Machinery, New York, NY, USA, 403–416. https://doi.org/10.1145/3034786.3056114

[12] Wenfei Fan, Xin Wang, Yinghui Wu, and Jingbo Xu. 2015. Association Rules with Graph Patterns. *Proc. VLDB Endow.* 8 (2015), 1502–1513. https://api.semanticscholar.org/CorpusID:7639544

[13] Wenfei Fan, Yinghui Wu, and Jingbo Xu. 2016. Functional Dependencies for Graphs. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 1843–1857. https://doi.org/10.1145/2882903.2915232

[14] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a Graph: Encoding Graphs for Large Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR) 2024.* International Conference on Learning Representations, Vienna, Austria. https://iclr.cc/Conferences/2024/AuthorGuide

[15] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*. 413–422.

[16] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking. *arXiv preprint arXiv:2305.15066* (2023). https://doi.org/10.48550/arXiv.2305.15066 Version 2.

[17] Jonathan Lajus, Luis Galárraga, and Fabian Suchanek. 2020. Fast and Exact Rule Mining with AMIE 3. In *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings* (Published: 31 May 2020). Springer, 36–52. https://doi.org/10.1007/978-3-030-49461-2_3

[18] Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. ChatRule: Mining Logical Rules with Large Language Models for Knowledge Graph Reasoning. *arXiv preprint* (2023). arXiv:2309.01538 [cs.AI] https://doi.org/10.48550/arXiv.2309.01538 Submitted on 4 Sep 2023 (v1), last revised 22 Jan 2024 (this version, v3).

[19] Meta AI. 2024. Introducing Meta Llama 3: The Most Capable Openly Available LLM to Date. https://ai.meta.com/blog/meta-llama-3/

[20] Victoria Nebot and Rafael Berlanga. 2012. Finding association rules in semantic web data. *Knowledge-Based Systems* 25, 1 (2012), 51–62. https://doi.org/10.1016/j.knosys.2011.05.009 Special Issue on New Trends in Data Mining.

[21] Yun Peng, Sen Lin, Qian Chen, Shaowei Wang, Lyu Xu, Xiaojun Ren, Yafei Li, and Jianliang Xu. 2024. ChatGraph: Chat with Your Graphs . In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Los Alamitos, CA, USA, 5445–5448. https://doi.org/10.1109/ICDE60146.2024.00424

[22] Niroop Channa Rajashekar, Yeo Eun Shin, Yuan Pu, Sunny Chung, Kisung You, Mauro Giuffre, Colleen E Chan, Theo Saarinen, Allen Hsiao, Jasjeet Sekhon, et al. 2024. Human-Algorithmic Interaction Using a Large Language Model-Augmented Artificial Intelligence Clinical Decision Support System. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–20.

[23] Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A Survey of Large Language Models for Graphs. In *KDD '24: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 6616–6626. https://doi.org/10.1145/3637528.3671460 Published: 24 August 2024.

[24] Yuanyuan Tian. 2022. The World of Graph Databases from An Industry Perspective. arXiv:2211.13170 [cs.DB] https://arxiv.org/abs/2211.13170

[25] Ziang Xiao, Xingdi Yuan, Q. Vera Liao, Rania Abdelghani, and Pierre-Yves Oudeyer. 2023. Supporting Qualitative Analysis with Large Language Models: Combining Codebook with GPT-3 for Deductive Coding. In *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) *(IUI '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 75–78. https://doi.org/10.1145/3581754.3584136

[26] Zezhong Xu, Peng Ye, Hui Chen, Meng Zhao, Huajun Chen, and Wen" Zhang. 2022. Ruleformer: Context-aware Rule Mining over Knowledge Graph. In *Proceedings of the 29th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Gyeongju, Republic of Korea, 2551–2560. https://aclanthology.org/2022.coling-1.225