Generation of Consistency Rules for Property Graph Data Using Large Language Models

Presented by

Thi Hoa LE

Supervised by:
Prof. Angela BONIFATI
Prof. Andrea MAURI

Thesis submitted for the degree of:
Master 2 Data and Intelligence for Smart Systems
(DISS)

August 23, 2024

Table of Contents

Acknowledgements

I would like to express my sincere gratitude to my supervisors Professor Angela Bonifati and Professor Andrea Mauri for their invaluable guidance and support throughout this project. As a new student starting out in research, I am still lacking in many areas. Their insightful advice, experience and constant encouragement have been crucial in the successful completion of this work. I am very grateful their dedication to providing me with the best possible conditions to carry out my research.

I would also like to thanks to the LIRIS Laboratory at Lyon 1 University for providing the resources and an excellent research environment. The support and facilities provided by the laboratory have been instrumental in the progress and success of my project.

Moreover, I would love to thanks to my family especially my parents. Their love and belief in me have been a constant source of strength. Special thanks to all my friends and relatives for their continuous support and understanding throughout this journey.

Thank you all for your significant contributions and support.

List of Figures

List of Tables

1 Context

Graph data is used in many different fields, including life sciences, finance, security, logistics and planning. It is an important role in these areas because it helps to understand and analyze complex relationships and patterns. The quality of graph data is very important because it directly impacts the accuracy and reliability of the insights and decisions made from it. High-quality graph data leads to more accurate conclusions and better decision-making, while poor-quality data can result in misleading information and errors.

However, graph data often faces challenges such as errors, inconsistencies and inaccuracies. These problems can affect the properties or attributes of graph nodes such as individual entities and edges the connections between entities. These errors can be do to mistakes data entry, inconsistencies when merging data from different sources and noise or errors in data collection processes.

Due to these issues, cleaning and verifying the quality of graph data have become essential steps in the data preparation process. Data cleaning involves identifying and correcting errors and inconsistencies to improve the data's overall quality. Quality control measures help ensure that the data is accurate and reliable before it is used for analysis or decision-making. Proper data cleaning and quality control are crucial for achieving accurate results and making informed decisions based on graph data.

2 Introduction

Traditional methods [10] for cleaning and ensuring the quality of graph data often rely on predefined constraints and rules. These methods require expertise in the field because they depend on specific knowledge about the data and its intended use. For example, experts in a particular area need to determine which rules should be applied to identify and fix errors in the graph. This process can be time-consuming and labor-intensive especially when dealing with large and complex graphs. Additionally, these traditional methods assume that the structure or schema of the data is well understood and known in advance, which is not always the case.

One of the major challenges with predefined constraints is that they may not cover all possible errors or inconsistencies, particularly in dynamic or evolving data environments. As a result, maintaining and updating these rules becomes a continuous process, which can be complicated as the data changes.

Recently, the strong development of large language models (LLMs) in many fields has led to increasing interest in using LLMs to automatically generate consistency rules from graph data. LLMs are advanced algorithms capable of analyzing and understanding patterns and relationships in the data. Moreover, LLMs are notable for their ability to generate rules in natural language, making them more accessible and easier for people to understand. This shows great potential for automatically generating consistency rules directly from graph data.

Using LLMs to generate consistency rules for graph data is a significant ad-

Link code: github.com/LeHoa98ptit/These$_{M}aster_{2}/tree/main$

vancement, opening up the possibility of improving the efficiency and accuracy of the data quality control process. Instead of relying on fixed rules, LLMs can generate rules automatically, reducing manual work and enhancing the reliability of the data.

3 Problem Statement

Although AI-based methods especially LLMs have great potential to automatically generate consistency rules for graph data, several challenges remain. Traditional methods for ensuring consistency in graphs often rely on manually defining and enforcing rules, which is time-consuming and labor-intensive. This approach is not only time-consuming but also prone to errors and inconsistencies, especially in complex or evolving domains.

Previously, many studies [8, 14] have focused on mining rules for graphs using deep learning methods. More recent research has explored logical rule mining from knowledge graphs [12] using LLMs, which has advanced the automatic generation of rules from graph data. While these methods have made significant strides, they still have limitations. They often depend on pre-existing assumptions about data structure or require specific domain knowledge, which can reduce the flexibility and scalability of the generated rules.

The main problem this thesis addresses is how to use LLMs to automatically create consistency rules for property graphs, especially when schemas are not clearly defined or constraints are not well known. The goal is to understand how LLMs can be trained to understand and model the natural relationships within graph data and then generate rules that help identify and correct inconsistencies. This approach aims to reduce reliance on manual work, speed up data quality management, and improve the overall reliability of graph-based systems.

Research Question: Can LLMs be used to generate consistent rules for property graph data?

4 Literature Review

4.1 StructGPT: A General Framework for Large Language Model to Reason over Structured Data

Recent advancements in reasoning over structured data with large language models (LLMs) have focused on integrating traditional structured data processing techniques with the powerful capabilities of LLMs. Traditional methods for knowledge graphs (KGs), tables and databases often rely on specialized algorithms and domain-specific approaches, but these methods struggle with generalization and handling complex queries. For knowledge graphs, traditional reasoning methods include graph traversal and logic-based approaches, while neural network embeddings have been explored but require extensive training data. In the context of tables, techniques such as TableQA and Text-to-SQL

have shown promise in bridging natural language and structured table data, yet they often struggle with complex queries and require significant training data. For databases, recent advancements have focused on generating SQL queries from natural language inputs using LLMs, but challenges remain in handling complex schemas and ensuring query accuracy. Efforts to combine LLMs with structured data processing include the development of specialized interfaces and hybrid models; however, direct interaction with structured data formats remains a challenge. StructGPT [9] addresses these limitations by providing a framework that separates the reading and reasoning processes. It uses specialized interfaces for KGs, tables, and databases, and employs an "Invoking-Linearization-Generation" procedure to enhance reasoning capabilities. Experimental results show that StructGPT significantly improves zero-shot performance on KGQA, TableQA and Text-to-SQL datasets, achieving results comparable to fully supervised methods.

4.2 ChatRule: Mining Logical Rules with Large Language Models for Knowledge Graph Reasoning

ChatRule [12] is a framework developed to enhance the process of logical rule mining in knowledge graphs (KGs) by utilizing large language models (LLMs). The primary goal of ChatRule is to generate semantically rich and accurate logical rules that improve reasoning tasks within KGs, addressing the limitations of traditional methods that often struggle with scalability and semantic integration. ChatRule employs a few-shot prompting technique, where LLMs are provided with a small number of examples to generate candidate logical rules by analyzing both the structural patterns and semantic content of the KG. These rules are then ranked based on their quality and relevance, with a ranking module evaluating them against existing KG facts to ensure accuracy. The top-ranked rules are subsequently applied to the KG to infer new knowledge and fill in missing information. Results indicate that ChatRule significantly outperforms traditional rule mining methods, offering better scalability and more precise reasoning in large-scale KGs, making it a valuable tool for advancing knowledge graph reasoning.

4.3 ChatGraph: Chat with Your Graphs

The **ChatGraph** [13] framework is proposed by Peng et al. This research introduces a novel approach to graph analysis by leveraging LLMs to allow users to interact with graphs through natural language prompts. ChatGraph distinguishes itself by integrating three core modules:

- **API Retrieval Module:** This module searches for relevant APIs based on the user's prompts, utilizing an embedding-based search mechanism.
- **Graph-aware LLM Module:** This component enables the LLM to understand and process graph structures by sequentializing graph data into paths that the LLM can interpret.

- **API Chain-oriented Finetuning Module:** It fine-tunes the LLM to generate accurate sequences of API calls that address the user's query, guided by a node matching-based loss function and a search-based prediction mechanism.

The framework was demonstrated to be effective in four key scenarios: graph understanding, comparison, cleaning, and API chain monitoring, showcasing its flexibility and efficiency in real-world applications.

4.4 Synthesis and Implications for Generating Consistency Rules with LLMs

The use of Large Language Models (LLMs) for creating consistent rules for property graphs is still in the early stages, with limited research directly addressing this area. However, existing research on LLMs and their integration with structured data offers useful insights that could guide future developments. **StructGPT** presents a framework that separates reading from reasoning, improving how LLMs handle structured data like knowledge graphs, data tables and databases. Although StructGPT does not focus specifically on property graphs, its method of combining LLMs with traditional techniques could be applied to property graphs. This approach could enhance how LLMs generate consistent rules by managing complex structures and ensuring accurate rule application. **ChatRule** is another framework that uses LLMs to work with logical rules in knowledge graphs, especially through few-shot prompting. It creates precise and meaningful logical rules, which improves reasoning within knowledge graphs. While not yet used for property graphs, ChatRule's methods could be adapted to generate consistent rules for property graphs, even with limited data. Additionally, **ChatGraph** combines natural language interaction with graph data, showing how rule creation can become more intuitive and user-friendly. By integrating ideas from these approaches, future research could improve how LLMs create and apply consistent rules for property graphs.nerating and applying consistency rules for property graphs

5 Methodology

5.1 Graph Encoding

To enable Large Language Models (LLMs) to process property graphs, it is essential to encode these graphs into a format that the models can understand. In this research, I use an incident encoder to achieve this. The incident encoder is designed to capture both the structure of the graph and the attributes of its nodes and edges, enabling LLMs to generate consistency rules effectively.

A property graph is a type of graph modelwhere relationships not only are connections but also carry a name (type) and some properties [5]. The challenge is encoding this detailed information so the LLM can analyze and draw meaningful conclusions.

The choice of the incident encoder is based on its demonstrated effectiveness in prior research [3]. They showed it provides the best results for translating complex graph relationships into a format that LLMs can interpret. This encoding ensures that the LLM receives a comprehensive representation of the graph.

Once encoded, this information is fed into the LLM, which uses it to generate rules that ensure the consistency of the graph. By employing the incident encoder, the LLM can effectively generate onsistency rules to property graphs.

5.2 Types Of Rules

In this research, Graph Functional Dependencies (GFDs) and Graph Entity Dependencies (GEDs) are employed as foundational elements for generating consistency rules using Large Language Models (LLMs). These types of dependencies play an important role in guiding LLM to generate consistency rules.

GFDs is a class of dependencies for property graphs that sub-sume Functional Dependencies (FDs) and Conditional Functional Dependencies (CFDs). Their expressive power allows to capture inconsistencies on properties of the same vertex (edge, resp.) or across different vertices (edges, resp.). As opposed to relational FDs, that only involve dependencies among properties, GFDs for property graphs allows the specification of two constraints: topological constraint and property dependency. [4]

GFDs do not cover many practical classes of database constraints. As an example, they cannot express key constraints on graphs, imposing that two identical vertices cannot exist and should be merged into one. Graph Entity Dependencies is a class of dependencies that extend GFDs by adding equalities of vertex identities. [4]

5.3 Rule Generation

When using large language models (LLMs) to generate consistent rules for graph data, dealing with the input size limitation is a challenge. There are three approaches to address this challenge. They are fine-tuning LLMs, slide window attention and retrieval augmented generation (RAG). Fine-tuning LLMs, while effective in improving model performance and tuning it for specific tasks, is however expensive, time-consuming and requires retraining whenever new data is added. Therefore, in this study, the focus will be on using the slice window attention and RAG methods.

5.3.1 Slice Window Attention

The first step in the **Slice Window Attention** method is to encode the property graph. Unlike the typical encoding methods in deep learning models that convert data into numerical vectors. The graph is represented in text format that Large Language Models (LLMs) can understand and process. This encod-

ing must ensure that the structure and important information of the graph are preserved without losing its original meaning.

After the graph has been converted into text, the next step is to divide this text into smaller sections called "windows". When dividing the data into windows, overlapping between sections is necessary to ensure that the meaning of the data at the boundaries of the windows is not lost. This is an important step to handle the input size limitation of LLMs while still maintaining the integrity of the information.

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 LLMs for processing. Based on the information from the windows and the instructions from the prompts, the LLMs generate 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.

The **Slice Window Attention** method offers several key benefits. It allows efficient processing of large datasets without encountering input size issues with LLMs. Additionally, this method can be easily scaled for complex datasets, while enabling a focus on the local context of each part of the graph, resulting in precise and detailed rule generation. However, some limitations of this method include 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.

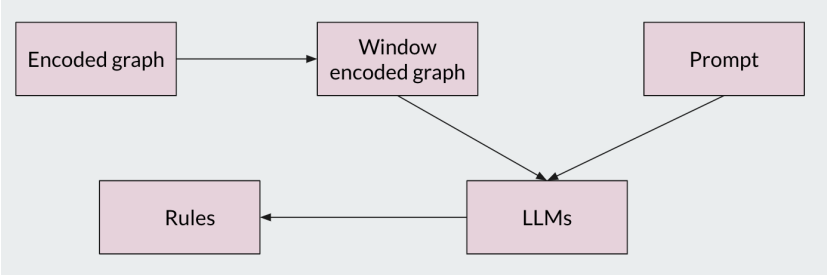


Figure 1: Slice Window Attention Flow

5.3.2 Retrieval Augmented Generation

Retrieval Augmented Generation combines information retrieval and text generation to improve the accuracy and consistency of the rules.

The process starts by converting graphs containing structural and relational information into text that LLMs can understand. Just like Slice Window Attention, this helps ensuring the meaning of the data and LLMs do the work.

The encoded data is then converted into vector embeddings and stored in a vector database. This database helps to find relevant information quickly and efficiently.

Once the vector database is set up, prompts are created and sent to the LLMs. These prompts are specific questions or requests used to search for information in the database. The prompts are not converted into embeddings but are used to query the database and retrieve relevant data. Then the LLMs use this information to generate responses or new rules. The generation process uses both the retrieved data and the model's ability to create text to ensure that the rules are accurate and consistent.

Finally, the responses from the LLMs are reviewed to make sure the generated rules meet the required standards of consistency and quality. This method effectively combines knowledge retrieval and text generation, while also providing flexibility in handling information. However, it's important to consider that the quality of the retrieval and the complexity of the process can affect the overall effectiveness and processing time.

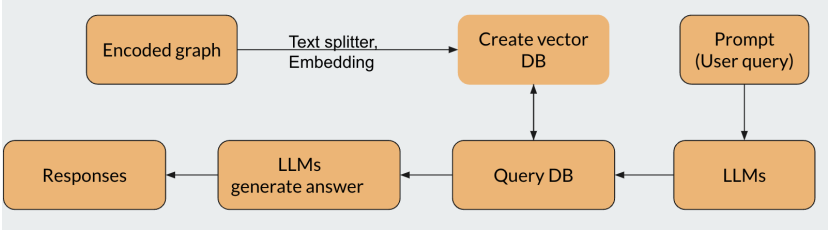


Figure 2: Retrieval Augmented Generation Flow

6 Experiment

6.1 Experiment Design

6.1.1 Properties Graph Data Selection

The main goal of the experiments in this research is to evaluate the ability of large language models (LLMs) to generate consistency rules for property graphs. To do this, I performed experiments with two specific property graphs:

Property Graph 1: Women's World Cup 2019 Graph [6]

- Nodes: 2,468
- Edges: 14,799

This graph depicts information related to the 2019 Women's World Cup, including nodes such as teams, person, matches, tournament, squad and the relationships between them. It was created from data covering all World Cups from

1991 to 2019, detailing matches, lineups and goal scorers. This graph's complex structure with a large number of nodes and edges. This graph's complex structure with a large number of nodes and edges. It provides a rich context for testing the accuracy and applicability

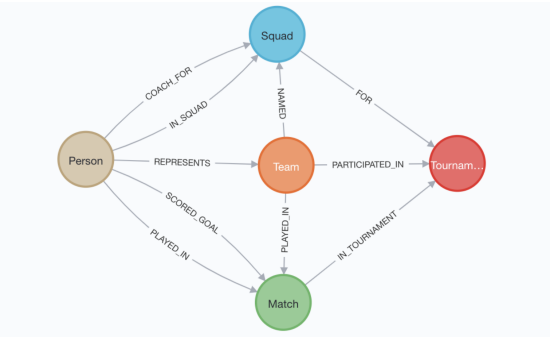


Figure 3: Women's World Cup 2019 Graph Example

Property Graph 2: Cybersecurity Graph [7]

- Nodes: 953
- Edges: 4,838

This graph is related to cybersecurity, including systems, security events and their interconnections.Computer security, cybersecurity or information technology security (IT security) is the protection of computer systems and networks from information disclosure, theft of or damage to their hardware, software, or electronic data, as well as from the disruption or misdirection of the services they provide. The field is becoming increasingly significant due to the continuously expanding reliance on computer systems, the Internet and wireless network standards such as Bluetooth and Wi-Fi, and due to the growth of "smart" devices. It provides insights into the challenges of the field and allows us to assess the effectiveness of consistency rules in a complex and rapidly changing domain.

Selecting these two graphs ensures that the testing methods can be applied and evaluated across different types of graphs with varying complexities. One contains fairly common, user-friendly and easy-to-understand data. While the other pertains to a specific specialized field. This approach helps accurately evaluating the generation of consistency rules in diverse real-world contexts.

6.1.2 Graph Encoding Methods

I use the **incident** encoder as I described above to encode property graphs. This method helps to convert the structure and properties of nodes and edges in the graph into a format that large language models (LLMs) can understand. The incident encoder focuses on encoding information related to events and related

Figure 4: CyberSecurity Graph Example

properties, helping the LLM model to recognize and process the information more effectively.

6.1.3 Zero-Shot and Few-Shot Prompts

To generate consistent rules for attribute graphs using LLMs, prompts play an important role in guiding the model to generate relevant and accurate rules. I used two types of prompts to guide the Large Language Models (LLMs) in generating 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.

Zero-Shot Prompt: The zero-shot prompt approach that the LLM to generate consistency rules without providing specific examples. This method leverages the model's general understanding of graph structures and consistency requirements. This prompt does not include specific examples but instead asks the LLM to generate rules based on its understanding of the graph's structure and relationships. It requires the model to provide detailed rules and Cypher queries to ensure data consistency and accuracy.

Few-Shot Prompt: The few-shot prompt approach that providing the LLM with a few examples of consistency rules to guide its generation process. This method helps the model understand the specific format and type of rules that are expected.This prompt includes specific examples and a task description, guiding the LLM to produce rules that align with the provided examples. It asks for new rules to maintain consistency and accuracy, along with a clear

description and a Cypher query for validation.

Using both zero-shot and few-shot prompts allows for a comprehensive evaluation of the capabilities of LLM. In the experiments, these prompts help evaluate the effectiveness of LLM in generating consistent rules and understand its strengths and limitations in handling different types of prompts.

In this experiment, I will keep the prompt the same for both graphs to assess the impact of the prompt on rule generation for two data sets with different characteristics.

6.1.4 Techniques for generating consistency rules

In the experiment, two techniques were used to generate consistency rules: **Slice Window Attention** and **Retrieval-Augmented Generation (RAG)**. Both techniques are discussed in detail in the methodology section and were chosen because of their complementary strengths in processing and understanding graph data.

The LLMs I employ are **LLAMA-3** [11] and **Mixtral** [2], which are used to compare the performance between these two models. **LLaMa 3** and **Mixtral** are two large language models from Meta. Each model has a different structure and purpose.

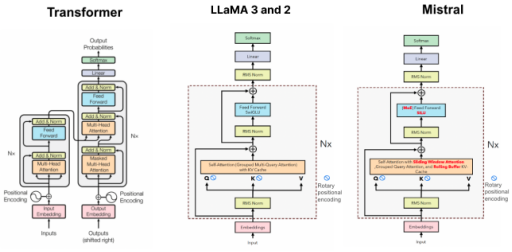
LLaMa 3 uses the Transformer architecture, which is well-known for handling natural language processing tasks. This model features self-attention mechanisms that help it understand and generate text more effectively. **LLaMa 3** builds on earlier versions by increasing the number of layers and attention heads. It improves its ability to handle complex language tasks. It can be fine-tuned for specific applications like answering questions, text generation, and translation, providing high accuracy in these tasks.

On the other hand, **Mixtral** takes a different approach by combining various machine learning techniques. It merges deep learning with traditional machine learning methods, creating a more flexible structure that can handle diverse and complex problems. **Mixtral** can be used for many different tasks, not just language processing but also data analysis and pattern recognition. This integration allows **Mixtral** to address a wide range of challenges effectively.

6.1.5 Ranking Measures

To evaluate the effectiveness of the consistency rules generated for property graphs. I use some ranking measures: **Support**, **Coverage** and **Confidence**. These measures provide a comprehensive assessment of the rules' performance and reliability.

Support [13] measures the number of facts in the graph that satisfy a given rule. It helps determine how frequently the rule applies within the graph. A higher support indicates that the rule is applicable to more facts.



Feature	Plain Transformers	LLaMA 3 and 2	Mistral
Architecture	Encoder-Decoder	Single Layer	Single Layer + Enhancements
Positional Encoding	Standard	Rotary	Rotary + Additional Innovations
Normalization	LayerNorm	RMS Norm	RMS Norm + Enhanced Mechanisms
Attention Mechanism	Self-Attention	Grouped Multi-Query Attention	Sliding Window Attention
Memory Efficiency	N/A	KV Cache	Rolling Buffer KV Cache
Activation in Feedforward	RELU	SwiGLU	SwiGLU + Mixture of Experts

Figure 5: Architecture of LLaMa and Mixtral [1]

$$\text{supp}(\rho) := \#\{(e, e') \mid \exists (e_1, r_1, e_2) \wedge \cdots \wedge (e_{L-1}, r_L, e') : \text{body}(\rho) \wedge (e, r_h, e') \in G\} \tag{1}$$

Coverage [13] 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. Coverage provides insight into how well the rule addresses the relevant facts.

$$\text{cove}(\rho) := \frac{\text{supp}(\rho)}{\#\{(e, e') \mid (e, r_h, e') \in G\}} \tag{2}$$

Confidence [13] 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.

$$\text{conf}(\rho) := \frac{\text{supp}(\rho)}{\#\{(e, e') \mid \text{body}(\rho) \in G\}} \tag{3}$$

6.2 Results And Evaluations

6.2.1 Women's World Cup 2019 Graph

The results from applying consistency rule generation methods on the **WWC-2019** property graph dataset reveal a clear distinction between the models LLAMA-3 and Mixtral. 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 545 with Slice Window Attention and 1604 with RAG, coverage of 98.56% and 100%, and confidence of 98.58% and 91.57% respectively. In contrast, Mixtral shows lower support scores but demonstrates the ability to discover more interesting rules, such as "Each Match node should have a unique date and stage within a specific tournament," despite lower coverage and confidence. You can refer to this demo [1] for more details.

Zero-shot / WWC	Slice Window Attention			RAG		
	support	**coverage %**	**confidence%**	**support**	**coverage%**	**confidence%**
LLAMA-3	545	98.56	98.58	1604	100	91.57
Mixtral	1257	97.81	89.5	923	73.75	67.83

Table 1: Results on the WWC-2019 Property Graph with Zero-Shot Prompt

In the Few-Shot approach, LLAMA-3 continues to excel with superior coverage and confidence, with support scores of 716 and 1002 for Slice 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 6354 for the two methods, with coverage of 92.81% and 63.12% and confidence of 95.75% and 80%. Although Mixtral does not achieve the highest scores for performance metrics, it provides more detailed and interesting rules, demonstrating its capability to generate valuable rules in the context of complex graph data.

Few-shot / WWC	Slice Window Attention			RAG		
	support	**coverage %**	**confidence%**	**support**	**coverage%**	**confidence%**
LLAMA-3	716	92.22	100	1002	73.75	73.5
Mixtral	622	92.81	95.75	6354	63.12	80

Table 2: Results on the WWC-2019 Property Graph with Few-Shot Prompt

6.2.2 CyberSecurity Graph

In a Zero-Shot setting, RAG (Retrieval-Augmented Generation) shows significantly lower performance compared to other models. This can be explained by the nature of the data and RAG's approach. The Cybersecurity Property Graph data is specialized unlike the WWC-2019 dataset, which is quite user-friendly, In a zero-shot context, LLMs (Large Language Models) do not have specific information about this data. This makes RAG, which relies on finding similar information and generating rules from stored data, less effective. RAG searches for similar information and creates rules based on that, but in this case,

[1] `https://these-master-2.onrender.com/`

the specialized data and lack of specific details limit RAG's ability. As a result, RAG has lower support, coverage, and confidence in the zero-shot environment, indicating that it struggles to apply meaningful rules from the available information.

Zero-shot / CyberSecurity	Slice Window Attention			RAG		
	support	**coverage %**	**confidence%**	**support**	**coverage%**	**confidence%**
LLAMA-3	402	82.6	100	23	20.25	38.4
Mixtral	439	71.4	71.4	57	25	26.5

Table 3: Results on the Cybersecurity Property Graph with Zero-Shot Prompt

In contrast, when moving to a Few-Shot environment, where additional examples and specific guidance are provided. RAG's performance improves significantly. Providing information and specific examples helps the model better understand data patterns and apply rules more accurately. With concrete examples, RAG can leverage this information to improve accuracy and create rules that better match the specialized data. As a result, in a few-shot setting, RAG shows significant improvements in support, coverage, and confidence, demonstrating its ability to use specific examples to generate better rules.

Zero-shot / CyberSecurity	Slice Window Attention			RAG		
	support	**coverage %**	**confidence%**	**support**	**coverage%**	**confidence%**
LLAMA-3	1113	89.38	97.86	660	89.57	99.7
Mixtral	575	91.9	91.69	1499	79.69	80

Table 4: Results on the Cybersecurity Property Graph with Few-Shot Prompt

6.2.3 Discussion

The results from evaluating the **Cybersecurity Property Graph** and **WWC-2019** datasets reveal distinct differences in model performance under zero-shot and few-shot settings.

For the Cybersecurity Property Graph, RAG performs poorly in the zero-shot setting due to the dataset's specialized nature and lack of relevant examples. Without specific context, RAG struggles to effectively retrieve and generate meaningful rules. However, in the few-shot setting, RAG shows significant improvement as it benefits from additional examples, leading to better performance in terms of support, coverage, and confidence.

In contrast, the WWC-2019 dataset, being more general and user-friendly, allows for better zero-shot performance across models, including RAG. This is due to the dataset's broader applicability, which provides more context for the

models. The few-shot setting further enhances performance, as the availability of specific examples helps refine rule generation.

Overall, domain-specific datasets challenge models in zero-shot scenarios but benefit from few-shot learning, highlighting the importance of context and examples in improving model accuracy.

7 Conclusion

This research has taken the initial steps in exploring the potential of Large Language Models (LLMs) for automatically generating consistency rules for property graph data. Unlike traditional methods that rely on manual rule-setting and domain expertise, LLMs offer a more adaptable and efficient approach by learning directly from the data. While the study shows promising results in identifying and correcting errors in graph data, it also highlights the challenges of applying these models to highly specialized datasets.

However, There are challenges remain especially in handling highly specialized datasets and ensuring that the models can generalize well across different domains. Future research should focus on refining these models to better understand and process complex and dynamic graph data, thereby further enhancing their ability to automate data quality control in various fields. Developing a framework to repair dirty graphs, further refining these models to better handle complex and dynamic graph data and enhancing their effectiveness in automating data quality control across various domains.

References

[1] Architecture of llama and mixtral. In *https://lightning.ai/fareedhassankhan12/studios/building-llama-3-from-scratch*.

[2] Alexandre Sablayrolles Albert Q. Jiang. Mixtral of experts. *arXiv*, 2401.04088(v1), 2024. Comments: See more details at this URL.

[3] Bryan Perozzi Bahare Fatemi, Jonathan Halcrow. Talk like a graph: Encoding graphs for large language models. *ICLR 2024*, 2024.

[4] Angela Bonifati, George Fletcher, Hannes Voigt, and Nikolay Yakovets. Querying graphs. In *Synthesis Lectures on Data Management*, chapter 4, pages 37–54. Morgan & Claypool Publishers, 2018.

[5] Dataversity. What is a property graph? In *https://www.dataversity.net/what-is-a-property-graph/*, 2021.

[6] Neo4j Graph Examples. Women's world cup 2019 graph. In *https://github.com/neo4j-graph-examples/wwc2019/tree/main*, 2020.

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

[8] Kewen Wang Pouya Ghiasnezhad Omran Jiangmeng Li Hong Wu, Zhe Wang. Rule learning over knowledge graphs: A review. *ACM Transactions on Knowledge Discovery from Data*, 1(1):7, 2023. Category: Survey; ACM Subject Classification: Computing methodologies → Knowledge representation and reasoning; Information systems → Data mining.

[9] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*, 2023. Submitted on 16 May 2023 (v1), last revised 23 Oct 2023 (this version, v2).

[10] Alexander Lauer. Rule-based graph repair using minimally restricted consistency-improving transformations. *arXiv*, 2307.09150(v1), 2023. Subjects: Software Engineering (cs.SE).

[11] AI @ Meta Llama Team. The llama 3 herd of models. *arXiv*, 2407.21783(v2), 2024. Submitted on 31 Jul 2024, last revised 15 Aug 2024.

[12] Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. *arXiv preprint*, 2023. Submitted on 4 Sep 2023 (v1), last revised 22 Jan 2024 (this version, v3).

[13] Yun Peng, Sen Lin, Qian Chen, Lyu Xu, Xiaojun Ren, Yafei Li, and Jianliang Xu. Chatgraph: Chat with your graphs. *arXiv preprint*, 2024. Submitted on 23 Jan 2024.

[14] Hui Chen Meng Zhao Huajun Chen Wen Zhang Zezhong Xu, Peng Ye. Ruleformer: Context-aware rule mining over knowledge graph. In *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, October 2022.