Towards Semi-Supervised Data Quality Detection in Graphs

Rubab Zahra Sarfraz
rubabzsarfraz@gmail.com
BridgeLinx Technologies
Lahore, Pakistan

ABSTRACT

Graph databases have emerged as a powerful tool for representing and analyzing complex relationships in various domains, including social networks, healthcare, and financial systems. Despite their growing popularity, data quality issues such as node duplication, missing nodes or edges, incorrect formats, stale data, and misconfigured topology remain prevalent. While there are numerous libraries and approaches for addressing data quality in tabular data, graph-structured data pose unique challenges of their own. In this paper, we explore an automated approach for detecting data quality issues in graph structured data which focuses on both node attributes and relationships. Since data quality is often governed by pre-established rules and is highly context-dependent, our approach seeks to balance rule-based control with the automation potential of machine learning. We investigate the capabilities of graph convolutional networks (GCNs) and large language models (LLMs) at detecting data quality issues using a few-shot learning approach. We evaluate the data quality detection rates of these models on a graph dataset and compare their effectiveness and potential impact on improving data quality. Our results indicate that LLMs exhibit robust generalization capabilities from limited samples while GCNs offer distinct advantages in certain contexts.

VLDB Workshop Reference Format:
Rubab Zahra Sarfraz. Towards Semi-Supervised Data Quality Detection in Graphs. VLDB 2024 Workshop: 13th International Workshop on Quality in Databases (QDB'24).

1 INTRODUCTION

The rapid digitization of the world has led to the generation, storage, and utilization of vast amounts of data every day. In order to effectively utilize such a data-rich environment, it is crucial to manage data quality meticulously to minimize irrelevant or erroneous information. To this end, graph data structures have been effectively utilized to lay the foundation of many fields such as transportation, genome analysis, and neuroscience applications. More recently, they have found an interesting position as central units in modeling complex problems such as image detection, recommendation systems, and sequence analysis.

In this paper, we focus on the unique challenges posed by graph-structured data that powers many database today. Unlike traditional databases, graph databases offer significant flexibility, often with limited schemas, leading to potential issues with data validation. The inherent flexibility of graph databases allows for dynamic and evolving schemas, but this flexibility comes with the cost of potentially introducing data quality issues that are harder to detect and correct. The goal of this study is to find ways to automate the detection of these data quality issues without the need to manually inspect each node and edge, thereby enhancing the efficiency and reliability of graph-structured data.

Data quality and corruption issues can manifest in various forms. Investigating and capturing these issues can be a time- and compute-intensive task with varying reliability levels. We aim to devise an approach to facilitate the detection of such issues with limited human intervention. For this study, we limit the scope of data quality issues to missing, incorrect, and inconsistent data. Our approach is based on the hypothesis that data quality is context-dependent: for example, route estimation requires high-quality spatial and temporal data, while the quality of textual data may be less critical to its functioning.

We use large language models (LLMs) and graph convolutional networks (GCNs) due to their semi-supervised capabilities, which are advantageous in scenarios with limited labeled data. LLMs excel in few-shot learning scenarios, making them well-suited for applications where annotated data is scarce. GCNs, on the other hand, are inherently suited for graph-structured data due to their ability to capture the spatial dependencies and relationships between nodes. By leveraging these models, we aim to improve the automated detection of data quality issues in graphs.

Our contributions are as follows:

- We present a semi-supervised machine learning approach to detect data quality issues in graph-structured data.
- We utilize LLMs and GCNs to address this specific problem, leveraging their strengths in few-shot learning and graph representation, respectively.
- We demonstrate promising results on a graph dataset, highlighting the effectiveness of our approach.
- We discuss the potential for further research in this direction, exploring additional data quality dimensions and refining our models for better accuracy and scalability.

Through our research, we highlight the importance of maintaining high data quality in graph-structured data and show how advanced machine learning techniques can serve as promising solutions for automating and enhancing data quality assessment, ultimately supporting the robust performance of graph-based systems across various applications. We make our code available to the community[1].

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment. ISSN 2150-8097.

[1]https://github.com/rubabzs/graph-data-quality

2 RELATED WORK

2.1 Data Quality

Data quality is crucial for data-driven applications, impacting everything from assessment methodologies to machine learning performance and monitoring tools. This section reviews key contributions in these areas.

Batini et al. [1] provide a foundational survey on data quality assessment and improvement methodologies, categorizing various approaches and emphasizing tailored solutions for specific challenges. Some advocate for empirical methods to better understand and improve data quality, calling for rigorous evaluation of interventions [13]. A comprehensive review of tools for data quality measurement and monitoring by Ehrlinger and Wöß [6] discusses various tools' strengths and weaknesses and suggests best practices for putting them in practice.

From a systems perspective, tools like CLAMS [7] focus on ensuring data quality in large-scale data lakes by managing heterogeneous data sources appropriately. Schelter et al. [14] leverage machine learning to automate large-scale data quality verification, and complement this with Deequ [15], an automated tool for validating data quality in machine learning pipelines, ensuring it meets predefined standards before use.

2.2 Graph Convolutional Networks

Graph convolutional networks (GCNs) have become essential for modeling various data problems, enabling advances across many domains. Dwivedi et al. [5] benchmarked different GCN architectures, providing valuable insights into their strengths and weaknesses across multiple datasets and tasks, aiding in the selection of suitable models for specific applications, including data quality measurement.

Studies have aimed to enhance GCN efficiency and simplicity, such as Wu et al. [18], which focused on reducing computational complexity while maintaining performance, and Chen et al. [4], who developed Simple Graph Convolutional Networks (SGC), demonstrating how a linear model can match more complex architectures in performance, facilitating easier interpretation and application. We incorporate these findings by using a simple architecture in our research.

Lastly, Garcia and Bruna [10] explored GCNs in few-shot learning, showing their effectiveness with limited training data. This is particularly relevant for data quality tasks, where labeled data is scarce, and GCNs' ability to generalize from limited examples is crucial. To our knowledge, no prior work has applied GCNs to data quality detection tasks, which we address in this study.

2.3 Large Language Models

The integration of large language models (LLMs) into data management has shown considerable promise, transforming traditional tasks and enhancing data quality across various applications. Borisov et al. [2] provide a comprehensive survey on the application of deep neural networks, including LLMs, to tabular data, offering insights into state-of-the-art techniques and key challenges. Fernandez et al. [9] highlight the disruptive potential of LLMs in data management, emphasizing improvements in efficiency and accuracy.

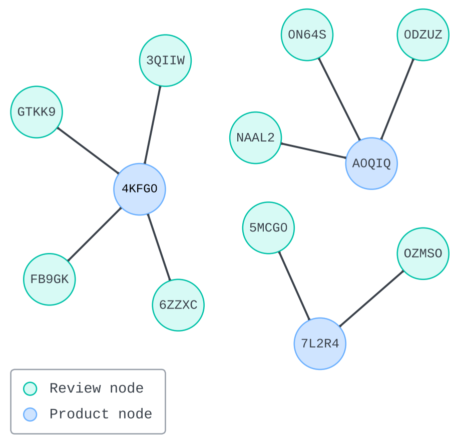

Figure 1: An illustration of the graph-structured data model used for data quality assessment in our experiments. The graph has a bipartite structure with nodes denoted by their unique IDs and edges representing relationships between product and review nodes.

Frameworks like Embdi [3], which generate embeddings for relational data integration, are crucial for maintaining data quality across disparate sources. Narayan et al. [12] examine foundation models in automating data wrangling tasks, suggesting significant improvements in data preparation efficiency and accuracy.

While literature on LLMs for graph-related tasks is limited, Fatemi et al. [8] present findings on encoding graphs for LLMs, exploring the impact of graph encoding functions and tasks on performance. Wang et al. [16] design a graph benchmarking task for language models, providing further insights. We build on these works to explore LLMs' potential in assessing data quality in graph-structured data.

3 METHODOLOGY

To detect data quality issues in graphs, we first formalize a measure of Data Quality Detection (DQD) which we evaluate our models with. We then present approaches for using a small set of labeled examples to predict detect data quality issues in an unseen data point in a few-shot setting. The objective is to perform minimal modifications to the models and utilize their intrinsic ability to generalize from a small amount of data. We use evaluate two classes of models: an off-the-shelf pre-trained large language model (LLM), specifically GPT-4o, and a graph convolutional network (GCN) that we train ourselves. We experiment with different numbers of few-shot samples and evaluate the performance of the models on the DQD rate.

Concretely, our problem can be defined as follows: given a set of vertices and edges $(V, E) \in G$, we introduce data quality issues to create a faulty version $(V', E') \in G'$, where V' and E' contain data quality issues. We treat our predictive models as a function f and aim to evaluate:

$$\text{DQD Rate} = \frac{1}{n}\sum_{i=1}^{n} f((V_i, E_i), S)$$

where:

$$G = \{(V_1, E_1), (V_2, E_2), (V_3, E_3), \ldots\}$$

$$G' = \{(V_1', E_1'), (V_2', E_2'), (V_3', E_3'), \ldots\}$$

$$S = \{(V_{1s}, E_{1s}), (V_2s, E_{2s}), (V_{3s}, E_{3s}), \ldots\}$$

$$f((V_1', E_1'), S) = \begin{cases} 1 & \text{if issue is detected} \\ 0 & \text{otherwise} \end{cases}$$

$$n = \text{number of data quality issues introduced}$$

Our goal is to evaluate the models' ability to accurately detect the faulty versions of V and E given a set of good and bad samples $(V_s, E_s) \in S$, which is a subset of the ground truth graph G'.

As an ablation study, we vary how many few-shot labeled examples are given to the models and evaluate the impact of this change on their detection rate. Additionally, we set the scope of our study based on some factors for simplicity. First, we restrict the types of data quality issues to missing, incorrect, and inconsistent data. Second, we minimize modifications to the existing models to ensure the focus remains on evaluating their inherent "off-the-shelf" capabilities. In the next section, we describe our experimental setup.

4 EXPERIMENTAL SETUP

Our experiments involved converting a graph dataset into a format, corrupting some nodes and edges in the dataset, and then performing DQD operations. We performed our experiments on a single node Linux-based machine using only the CPU for all local computation. All code was written in Python and we incorporated multiple standard graph processing and machine learning libraries to increase reproducibility. To ensure robustness, we repeated all experiments twice and report the average in the results section along with the variation observed. The following sections describe the setup in detail.

4.1 Dataset

We use the Amazon Fine Food Reviews dataset [11] for our experiments, which contains over a decade of reviews from the popular ecommerce website. This dataset contains comprehensive data about the products, reviews, and the reviewing users, and is well-suited for modeling as a graph due to the inherent relationship between products and their reviews. The dataset spans over 10 years and comprises 74,258 product nodes, 568,453 review nodes, and 256,059 user nodes. For our experiments, we extracted a subset of 4,000 nodes, focusing only on product and review nodes. The detailed features of this subset are shown in Figure 2. We preprocessed the dataset from its original text file format into a graph model and established edges between each product and its corresponding reviews to accurately represent the relationships within the data. The final model of the graph used can be seen in Figure 1.

```
product/productId: B001E4KFGO
review/userId: A3SGXH7AUHU8GW
review/profileName: delmartian
review/helpfulness: 1/1
review/score: 5.0
review/time: 130386240
review/summary: Good Quality Dog Food
review/text: I have bought several of the Vitality canned dog food
products and have found them all to be of good quality. The product
looks more like a stew than a processed meat and it smells better.
My Labrador is finicky and she appreciates this product better than
most.
```

Figure 2: An example of the graph attributes derived from the Amazon Fine Foods Reviews dataset.

4.2 Data Quality Issues

After creating the graph, we introduced data quality issues in our dataset. We corrupted 10% of our initial dataset (n=400, e=79) and introduced the following issue types:

- **Missing Node Features:** We walked through the graph and unset randomly chosen node attributes in both the product nodes and the review nodes, e.g. setting the `text` feature in review node to be blank or a product `name` to `None`.
- **Incorrect Node Features:** For the numerical columns, we added incorrect values in our dataset which were out of bounds, e.g. replacing the correct value of the `score` feature in the review node to a randomly chosen number in the range [-100000, 100000].
- **Inconsistent Node Features:** We introduced inconsistencies related to the formats and mismatched values, e.g. setting the nodes with positive `summary` to have a negative corresponding `text` feature in the review node. In addition, we also set some values to be numerical when the expectation was of textual data.
- **Inconsistent Relationships:** We introduced self-loops and misconfigured relationships in our graph to test e.g. a review node should not be linked with another review node, similarly a product node should not point to itself.

4.3 Crafting Samples for Few-shot Learning

To perform in-context learning, we provide the models with a small number of labeled examples at inference time that are carefully chosen to maximize generalizability, which is crucial in this task. We propose an approach where models are trained on a balanced sample of both, valid nodes and invalid nodes (in terms of attributes and relationships). For our 4-shot input samples, we include 2 examples per node type, one valid and one invalid. Specifically, this means providing our models with one valid product node, one invalid product node, one valid review node, and one invalid review node. We show one example of an invalid review node in Figure 3.

In the 12-shot scenario, we increase the number of examples to 3 per node type per validity status. This allows us to examine the impact of the training data size on model performance. By experimenting with both 4-shot and 12-shot samples, we aim to capture the potential differences in model performance that arise

```
{
  "id": "B001E4KFG0_A5GH6H7AUHU8GW_1303862400",
  "type": "Review",
  "properties": {
    "userId": "A3SGXH7AUHU8GW",
    "profileName": "jkhhjknmnmn",
    "helpfulness": "1/1",
    "score": -11115.0,
    "time": "1afaadasafa",
    "summary": "fajhjahjh2q8fuhaj",
    "text": "None"
  },
  "relationships": [
    {
      "type": "HAS_REVIEW",
      "target_id": "B001E4KFG0_A5GH6H7AUHU8GW_1303862400"
    }
  ],
  "label": "Invalid"
}
```

Figure 3: An example of an invalid review node where the `score` feature has an implausible value and the node is marked accordingly with the `label` property.

due to variations in the amount of training data provided. This approach helps us understand how the models generalize from a few examples and adapt to the intricacies of graph data. It is also pertinent to encode the data quality patterns that any user would like to filter out from their graph in this step, as the models will evaluate the incoming data on these samples.

4.4 Large Language Models (LLMs)

We first evaluated the ability of large language models (LLMs) for detecting data quality issues in graph-structured data. Specifically, we used GPT-4o where its objective was to predict whether, given a set of reference samples, an unseen node has invalid data or not. We encode both the samples and the graph's nodes and edges into text format to provide as input to the LLM and enable it to reason over them. The results were aggregated after the predictions were made and compared against our ground truth data to measure the Data Quality Detection (DQD) rate. The following sections outline this process in detail.

Prompt Engineering: To evaluate the inherent "off-the-shelf" capabilities of the models, we designed a minimalistic prompt free of dataset-specific details that could bias the results. We asked the LLM to learn data quality patterns from the samples provided, use them to predict the quality of the unseen nodes, and return its response as an array of JSON objects, one for each input node, containing a tuple of the node ID and the predicted label (node_id, predicted_label). The only explicit instruction provided to the LLM was:

```
"Learn the valid and invalid data quality patterns from
the samples given in the input."
```

Graph Encoding: To convert the node and edge data from graph format into a textual format that can be accurately understood by an LLM, we design a graph encoding function that maps attributes and relationships into key-value pair representation. For each instance, we include the node ID, type, feature values, a natural language description of its relationships with other nodes, and its data quality label in the following format:

```
Node X, Type: Product/Review
Node Features: time: "12287913", name: "Bob", ...
Relationships Linked: Node X is linked with Node Y
Label: Valid/Invalid
```

Batching: Having an LLM label one node at a time can be expensive, both in terms of cost and time, as the LLM has to be provided the few-shot examples for every input. In order to be more efficient, we batched inference requests by asking the LLM to label multiple input nodes simultaneously in the same prompt. In the 4-shot prompt, we used a batch size of 5 input nodes. For the 12-shot prompt, we used a batch size of 2 since the few-shot examples themselves took up a significant number of tokens in the context:

```
4-shot Token Size: ~350 tokens per sample
12-shot Token Size: ~2,100 tokens per sample
```

4.5 Graph Convolutional Networks (GCNs)

Graph convolutional networks (GCNs) are well-suited for this task due to their ability to effectively learn from graph-structured data with relatively limited training samples. Their semi-supervised learning capabilities also allow them to leverage both labeled and unlabeled data, which is greatly increases their applicability in real-world contexts where labeled data is limited.

Graph Pre-processing: We performed multiple steps of pre-processing to transform the graph into the appropriate format to feed it into the GCN model. We first used the NetworkX framework[2] to convert it into a format compatible with the Deep Graph Library (DGL), which facilitates efficient computation on graph-structured data.

We then encoded the node's textual features as dense embeddings using the SentenceTransformers library[3] which offers a unified framework for generating high-quality embeddings that capture the semantic meaning of the text. We use the `all-MiniLM-L6-v2` model which is based on a compressed transformer developed by Wang et al. [17] and use it to generate vector representations for both the few-shot examples and test nodes.

Model Training: We train a GCN on the node embeddings and use the relationship information provided by the DGL representation. We perform a grid search over a range of hyperparameter values, including the learning rate (0.001 to 0.01), the number of hidden layers (5 to 7), the number of epochs (10 to 20), and the early stopping patience (3 to 5). We used the Adam optimizer and the categorical cross-entropy objective function. We trained our model in two settings: one with 4-shot examples and another with 12-shot

[2]https://networkx.org
[3]https://www.sbert.net

Task Type	GCN		LLM	
	4-shot	**12-shot**	**4-shot**	**12-shot**
Missing Node Features	77.89%	**87.96%**	87.50%	76.00%
Incorrect Node Features	78.02%	87.50%	**94.95%**	**95.00%**
Inconsistent Node Features	75.12%	**85.30%**	77.23%	76.10%
Inconsistent Relationships	82.00%	**91.22%**	46.63%	46.80%

Table 1: Comparison of GCN and LLM on various tasks with different few-shot configurations.

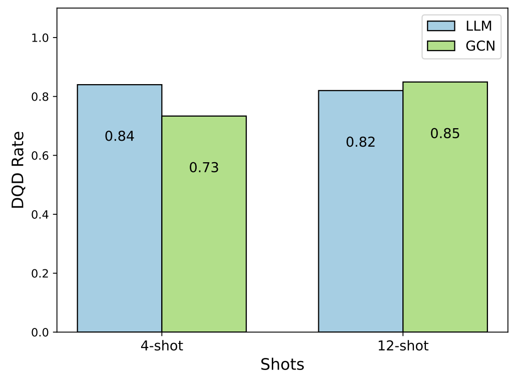

Figure 4: DQD Rate comparison between LLM and GCN with 4 and 12 few-shot learning samples.

examples. This approach allowed us to evaluate the model's performance with varying amounts of training data, and we performed this evaluation on a separate test set.

The training process can be summarized as follows:

- **Graph Conversion:** Transform the NetworkX graph into a DGL graph.
- **Feature Encoding:** Use SentenceTransformers to generate embeddings for node features.
- **Model Initialization:** Set up the GCN model architecture with the appropriate hyperparameters.
- **Training:** Train the model using a categorical cross-entropy loss function and the Adam optimizer.
- **Evaluation:** Evaluate the model's performance on a separate test set and measuring the detection rate on it.

Evaluation We evaluated the GCN model by comparing its predictions against the ground truth labels. The primary metric for performance assessment was the Data Quality Detection (DQD) rate, which was defined in Section 3. The results were aggregated and analyzed to determine the effectiveness of the GCN in detecting data quality issues in graph-structured data.

5 RESULTS

Our major takeaway from the experiments was the superior ability of LLMs to generalize from a few examples compared to GCNs. This makes LLMs particularly suitable for consistently evaluating data quality with minimal training data. However, if a sufficient amount of labaled data is available then GCNs can be employed instead. Below, we summarize our key findings:

Performance Comparison: The Data Quality Detection (DQD) rate for 4-shot LLMs came out to be 84%, which is notably higher compared to 73% for GCNs as shown in Figure 4. This highlights the effectiveness of LLMs in learning from very limited examples. It is also important to mention here that the loss for 4-shot GCN dropped from 0.7086 to 0.4100 while the validation accuracy fluctuated between 90% to 95%, which indicates that although the model is learning, it has the potential for overfitting. We recommend employing techniques like regularization or early stopping to prevent this.

Impact of Few-shot Examples: For 12-shot GCNs, the DQD rate improved significantly to 85%, slightly exceeding that of LLMs, indicating that additional training samples greatly increase the model's performance. However, the same cannot be said for the LLMs as their performance did not improve, which may be due to the prompt becoming very long.

Task-Specific Performance: A detailed breakdown of DQD rates per task and issue type presented in Table 1 shows that LLMs outperformed GCNs on certain tasks, while GCNs performed better on others. It's interesting to note that LLMs achieved an impressive rate of 95% on detecting incorrect node features, whereas GCN showed significantly higher detection rate of 91.22% for detecting inconsistent relationships. These inconsistent relationships included wrong self-loops as well as wrongly introduced relationships, e.g. replacing `HAS_REVIEW` with `REVIEW_OF`. LLMs performed significantly worse on this specific task, which requires further probing. One of the potential reasons could be LLMs not being able to capture the overall structure of the graph like GCNs do.

These findings underscore the importance of selecting the appropriate model based on the specific requirements and constraints of the data quality evaluation task.

6 DISCUSSION

In this section, we address three critical aspects of detecting data quality with our framework: the deteriorating performance of LLMs with increased few-shot examples, the selection criteria for models, and crafting effective samples for few-shot learning.

Deteriorating Performance of LLMs: The observed decline in the LLM's performance as we go from 4 to 12 few-shot examples warrants further investigation. It is possible that the increased prompt length is testing the limits of the LLMs' context window, which suggests that a delicate balance between maximizing in-context examples and preventing an overly large prompt is needed. Future experiments with alternative graph encoding functions may help in this regard by keeping only relevant information for each node. It is also reasonable to assume that with a more complex dataset, a richer representation of good and bad quality samples might be required. In such scenarios, GCNs might be preferable, particularly if a labeled dataset is available. An alternative approach to mitigate performance barriers could be fine-tuning the LLMs so they can learn from more examples without burdening the prompt.

Selection Criteria for Models: The choice of model should be guided by the specific task requirements, budget constraints, and available resources, particularly in terms of manual effort. While we tried to evaluate the potential of using advanced machine learning algorithms to reduce the manual effort at scale, if a simple rule-based approach can provide deterministic results with less labor-intensive work then it makes sense to opt for it instead. Monetary cost is another factor: closed-source LLMs can become fairly expensive to use compared to GCNs but demonstrate superior off-the-shelf performance with less manual intervention, especially when it comes to detecting issues with node features. GCNs on the other hand require more manual effort for model training, validation, testing, and hyperparameter tuning.

Crafting Effective Samples for Training: Regardless of the model chosen, the quality of few-shot examples is crucial in helping them generalize better. High-quality samples should comprehensively represent the expected data quality issues to to help detect data quality issues on unseen data accurately. Simultaneously, the few-shot examples must also be balanced to prevent any biases from impacting the models' predictions. While crafting new samples for every node type may appear to be a complex operation, this approach can be especially useful when working at scale where manual effort needs to be minimized.

7 CONCLUSION & FUTURE WORK

Managing data quality in graph-structured datasets is crucial, whether in offline databases or real-time analysis systems. In this paper, we explored the potential of semi-supervised learning for automating data quality assessment. By leveraging samples of both good and bad data, we evaluated the performance of large language models (LLMs) and graph convolutional networks (GCNs) in a few-shot setting. Our preliminary findings show a promising potential in further research on this topic. While LLMs excel in off-the-shelf data quality assessment, GCNs demonstrate significant performance gains when more labeled data is available and when graph relationships are needed to be tested. Additionally, we show that context can be effectively encoded with machine learning to detect data quality issues with minimal manual intervention. The choice of model is crucial in obtaining good results and should be informed by the specific graph and task at hand, as evidenced by our experiments. Since both approaches have their limitations, the decision must also factor in the monetary costs, time constraints, and technical capability needed for a particular task. Future work in this direction should focus on several key areas to further improve automated data quality assessment in graph databases:

- **Extensive Experiments:** Perform further evaluations on bigger datasets with more detailed graph-structured issues.
- **Fine-Tuning LLMs:** Investigate the impact of fine-tuning LLMs on domain-specific graph data to improve their performance.
- **Graph Encoding Functions:** Explore more methods to encode graphs for LLMs and observe their impact on the models' performance.
- **Hybrid Models:** Explore hybrid models that combine the strengths of LLMs and GCNs to leverage the benefits of both approaches.
- **Real-Time Analysis:** Extend the current approaches to real-time data quality monitoring and correction in dynamic graph databases.

We hope that this research encourages the community to explore automated methods for managing data quality in graph-structured data, ultimately leading to more robust and reliable data-driven systems.

REFERENCES

[1] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. 2009. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–52.

[2] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[3] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2021. Embdi: generating embeddings for relational data integration. In *Proceedings of the 29th Italian symposium on advanced database systems, SEBD*. 5–9.

[4] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International conference on machine learning*. PMLR, 1725–1735.

[5] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research* 24, 43 (2023), 1–48.

[6] Lisa Ehrlinger and Wolfram Wöß. 2022. A survey of data quality measurement and monitoring tools. *Frontiers in big data* 5 (2022), 850611.

[7] Mina Farid, Alexandra Roatis, Ihab F Ilyas, Hella-Franziska Hoffmann, and Xu Chu. 2016. CLAMS: bringing quality to data lakes. In *Proceedings of the 2016 International Conference on Management of Data*. 2089–2092.

[8] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560* (2023).

[9] Raul Castro Fernandez, Aaron J Elmore, Michael J Franklin, Sanjay Krishnan, and Chenhao Tan. 2023. How large language models will disrupt data management. *Proceedings of the VLDB Endowment* 16, 11 (2023), 3302–3309.

[10] Victor Garcia and Joan Bruna. 2017. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043* (2017).

[11] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[12] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. 2022. Can foundation models wrangle your data? *arXiv preprint arXiv:2205.09911* (2022).

[13] Shazia Sadiq, Tamraparni Dasu, Xin Luna Dong, Juliana Freire, Ihab F Ilyas, Sebastian Link, Miller J Miller, Felix Naumann, Xiaofang Zhou, and Divesh Srivastava. 2018. Data quality: The role of empiricism. *ACM SIGMOD Record* 46, 4 (2018), 35–43.

[14] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. 2018. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment* 11, 12 (2018), 1781–1794.

[15] Sebastian Schelter, Philipp Schmidt, Tammo Rukat, Mario Kiessling, Andrey Taptunov, Felix Biessmann, and Dustin Lange. 2018. Deequ-data quality validation for machine learning pipelines. (2018).

[16] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems* 36 (2024).

[17] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems* 33 (2020), 5776–5788.

[18] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.