1. Introduction

Anomaly detection manifests in tasks such as identifying fraudulent bank transactions, malicious bots on social networks, and cancerous cells living amongst healthy cells. However, in a sea of millions of transactions or billions of users, tasking a human with detecting anomalies is intractable. Thus, the problem of anomaly detection naturally lends itself to computation, but varying competitive computational methods have emerged in the field. Intuitively, these massive networks can be represented as graphs. In these graphs, individual data points are represented as nodes, and the connections between them are represented as edges (think points and lines connecting the points, respectively). In the financial example, nodes are individual transactions, and a hypothetical edge might represent two transactions made in the same hour. Graph anomaly detection (GAD) leverages graphical representations to some success, but no approaches have incorporated increasingly powerful and ubiquitous large language models (LLMs). The applicability and diversity of anomaly detection motivate augmenting the potential of current GAD approaches, and LLMs exhibit tremendous potential to this end.

2. Background and Literature Review

Complex systems can be modeled naturally by graphs. Specifically, past research with graph neural networks (GNNs), updateable computational representations of graphs, has demonstrated their success in applications ranging from traffic modeling to physical simulations [1,2]. Another field, graph anomaly detection, has emerged as a natural application for these potent GNNs. Recent research, however, has noted that GNNs alone fall short of the state-of-the-art GAD methods. Particularly, a comprehensive review by Tang et al. found that ensemble trees lead to the best performance on GAD tasks in their comprehensive benchmark testing [3]. Tang et al. also suggested that graph-structured information fails to consider "structure camouflage," the tendency for anomalous nodes to have few indications of their fraudulence, and "feature heterophily," the capability for different data points to have significant variations in their values. However, the authors also found that hyperparameter tuning in non-scarce environments, environments with abundant data, enabled competitive GNN performance when compared to tree ensemble methods. This hyperparameter tuning consists of updating model properties like learning rate and number of layers, distinct from updating model weights. This addition likely explains the results of Cheng et al., who found that GNNs expertly identified instances of financial fraud [4]. Their study includes applications that utilize hyperparameter tuning, like Cardoso et al.'s graph learning application for money laundering [6].

Recently, to much acclaim, large language models (LLMs) have had success in understanding and generating text. One of the forefront models, OpenAI's o1 outperforms Ph.D.-level students on the GPQA dataset, including questions from physics, chemistry, and biology [6]. Researchers broadly have begun incorporating LLMs in problems from modeling social media data to crystal structure generation [7,8]. To aid GNNs in modeling and understanding textual information, LLMs have been used in conjunction with GNNs on node classification tasks, tasks that involve classifying data points as one of several pre-defined types given a list of options. Two approaches have evolved in this field, as noted by Chen et al.: first, LLMs-as-enhancers, in which large language models are first used to augment data; second, LLMs-as-predictors. The former approach involves cost-heavy fine-tuning, updating of the LLM's internal values, which is often prohibitively expensive. The latter approach utilizes GNNs to craft optimal prompts for the LLM [9]. This approach was used by Hu et al. to successfully classify research papers using in-context learning (ICL), in which a GNN selected relevant examples to include in the LLM prompt [10].

Currently, GNN performance on GAD tasks seems to depend on the extent to which the GNN's hyperparameters are tuned. In the best, most optimally tuned models, GNNs document impressive performance. Given LLMs' inherent understanding and modeling capabilities and their success in conjunction with GNNs, the GNN-LLM approach should be applied to GAD tasks. Can LLMs augment non-hyperparameter-tuned GNNs enough to make them competitive with other state-of-the-art approaches in an LLM-as-predictors manner, and can the addition of hyperparameter tuning enable far superior performance to alternative approaches?

3. Methodology

In order to enable a wide-sweeping conclusion regarding LLM-enhanced GNNs applied to GAD tasks, I will collect and process several datasets. Fortunately, many datasets are already available [11]. These datasets, however, often include only the graph information, not the textual information which is critical for my project. To fix this, I will find the paper that curated the necessary datasets to repeat their approach.

With my datasets, I will need to train GNNs for the GAD task. To do this, I will write code in Python using the PyTorch and PyTorch Geometric libraries, which include helpful functionality for machine learning and graph tasks. The training phase will be based on that used by Hu et al.: a GNN will select example nodes, and an LLM will give these nodes a score based on their helpfulness [10]. Because I intend to test hyperparameter-tuned models, I will implement several of the most common hyperparameter-tuning methods, such as Bayesian Optimization, Grid Search, and Random Search [12]. Helpful libraries for these tasks include Scikit-learn and Optuna. Finally, with my trained GNN example selectors, I will write code to prompt various LLMs with examples of anomalous and non-anomalous nodes and a test node on which to make a prediction. With the LLM's answer, I will use its accuracy to answer my initial research question. Because the paper that found tree-ensembles superior to GNN approaches utilized the same datasets that I will use, I will be able to compare performance, answering whether LLM augmentation increases GNNs' competitiveness with state-of-the-art methods. Moreover, I can compare these results with the hyperparameter-tuned model to answer the second half of my research question.

4. Qualifications

Regarding the first facet of my proposed project, I have already demonstrated my experience manipulating datasets by creating labels for a dataset containing reviews of movie DVDs on Amazon (see Appendix A). I learned how to use Python to manipulate data when I did research for the Victor Fung Lab at The Georgia Institute of Technology, where I worked on a project using GNNs for materials science discovery. With the same group, I learned how to use PyTorch and PyTorch Geometric (see Appendix B). Finally, I will use either an application programming interface (a service offered that allows prompting of models like ChatGPT via code) or a small LLM that I download and host on Northwestern's Quest Computer to make predictions on test nodes. For the latter option, I will use models provided by Hugging Face. For this approach, I will utilize Hugging Face's ample courses and instruction.

Courses helpful to my proposed research include Honors Engineering Analysis 1, Fundamentals of Computer Programming 1, Fundamentals of Computer Programming 2, and Data Structures and Algorithms. Honors Engineering Analysis 1 includes linear algebra concepts that aid my understanding of machine learning literature. The latter three courses contribute to my general comfort with programming and computer science knowledge. The two Fundamentals courses provide comfort with object-oriented programming, and my project will include data structures that I have gained comfort with through Data Structures and Algorithms.

Budget

<u>OpenAI prompts</u>**:**

- Assuming that each LLM prompt contains 40 examples and each example contains ~100 words of raw data and ~100 words of metadata, each LLM prompt will contain 40*(100+100) = 8000 words. According to OpenAI, this is approximately 10,600 tokens. OpenAI prices its API at $15/1M input tokens. We will prompt the LLM to return just its determination of whether a test node is genuine or an anomaly (a 1 for genuine, a -1 for anomaly), which will contribute a negligible amount of tokens. Thus, each prompt costs ~16 cents. Assuming each dataset has an average of 10,000 data points, and we use a standard train split of 75% over 200 epochs, we will have 0.75*1,000 = 750 prompts per training round, and 1000*0.25= 250 per testing round. Thus the process beginning to end will take 750 + 250 = 1000 LLM prompts, costing 1000*.16 = $160 per complete round of training and inference. If we use 5 datasets, the total cost will be $160*5 = $800, leaving $200 to be used while building the model and testing it's functionality. All of these approximations came from [10] and [14].

Appendix

Appendix A:

Appendix B: https://github.com/walleio/RegeneronPaper/blob/main/regeneron_paper.pdf

Bibliography - ACM Format

[1] Sanchez-Gonzalez, Alvaro, et al. International Conference on Machine Learning, 2020.

[2] Learning to Simulate Complex Physics with Graph Networks. Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, Peter Battaglia; Proceedings of the 37th International Conference on Machine Learning, PMLR 119:8459-8468

[3] Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. 2024. GADBench: revisiting and benchmarking supervised graph anomaly detection. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23). Curran Associates Inc., Red Hook, NY, USA, Article 1289, 29628–29653

[4] Soroor Motie, Bijan Raahemi, Financial fraud detection using graph neural networks: A systematic review, Expert Systems with Applications, Volume 240, 2024, 122156, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.122156. (https://www.sciencedirect.com/science/article/pii/S0957417423026581)

[5] Mário Cardoso, Pedro Saleiro, and Pedro Bizarro. 2022. LaundroGraph: Self-Supervised Graph Representation Learning for Anti-Money Laundering. In Proceedings of the Third ACM International Conference on AI in Finance (ICAIF '22). Association for Computing Machinery, New York, NY, USA, 130–138. https://doi.org/10.1145/3533271.3561727

[6] OpenAI. 2024. Learning to Reason with LLMs. September (2024). Retrieved January 20, 2025 from https://openai.com/index/learning-to-reason-with-llms/.

[7] Social-LLM: Modeling User Behavior at Scale using Language Models and Social Network Data J Jiang, E Ferrara - arXiv preprint arXiv:2401.00893, 2023

[8] Mohanty, T., Mehta, M., Sayeed, H. M., Srikumar, V., & Sparks, T. D. (2024). CrysText: A Generative AI Approach for Text-Conditioned Crystal Structure Generation using LLM. *ChemRxiv*. https://doi.org/10.26434/chemrxiv-2024-gjhpq

[9] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. 2024. Exploring the Potential of Large Language Models (LLMs)in Learning on Graphs. SIGKDD Explor. Newsl. 25, 2 (December 2023), 42–61. https://doi.org/10.1145/3655103.3655110

[10] Zhengyu Hu, Yichuan Li, Zhengyu Chen, Jingang Wang, Han Liu, Kyumin Lee, Kaize Ding. 2025. Let's Ask GNN: Empowering Large Language Model for Graph In-Context Learning. Arxiv:2410.07074. Retrieved January 20, 2025 from https://arxiv.org/abs/2410.07074.

[11] FelixDJC. 2023. Awesome-Graph-Anomaly-Detection. (December 2023). Retrieved January 20, 2025 from
https://github.com/FelixDJC/Awesome-Graph-Anomaly-Detection/commits/main/.

[12] Amazon. What is Hyperparameter Tuning? Retrieved January 20, 2025 from https://aws.amazon.com/what-is/hyperparameter-tuning/.

[14] OpenAI. Pricing. Retrieved January 20, 2025 from https://openai.com/api/pricing/.