Linking Software Engineering Methodologies to Graph Neural Networks

Sofiane ENNADIR
KTH Royal Institute of Technology, Sweden
Assignment for WASP Software and Engineering Course Module 2 - 2023

For August 31st, 2023

1 Introduction

Graph-structured data is ubiquitous in many domains such as in chemoinformatics, bioinformatics and social network analysis. The multitude of machine learning applications in these domains motivated the development of neural network models that can operate on graphs, known as Graph Neural Networks (GNNs). In the past years, GNNs have emerged as a powerful tool for learning both node and graph representations. These models have been applied to several different problems including drug design [1] and session-based recommendation [2]. Nevertheless, as a generalization of classical Neural Networks, GNNs inherited the susceptibility to adversarial attacks which are perturbations subtly introduced into input data. These attacks pose a critical threat to the reliability of GNNs, particularly in safety-critical applications such as for example in finance and healthcare. From this perspective, my main research is related to graph adversarial attacks and defense schemes where I investigate and develop new attack/defense methods to make GNNs more robust. Additionally, my second area of research is related to investigating applications of GNNs for different areas such as Bioinformatics.

Graph adversarial attacks refer to a type of attack on GNNs where an adversary tries to introduce perturbations in the input graph to mislead the network's output. The aim of my research in this area is to develop methods to prevent these attacks, or to improve the GNN's ability to detect and correct them. For instance, we have adapted the famous Graph Convolution Neural network (GCN) by forcing the layer's weight to be orthogonal matrices enhancing their robustness for both structural and node features-based adversarial attacks. From another perspective, our work focused on developing new attacks to investigate the GNNs robustness. For instance, we proposed an unbounded adversarial attacks scheme for graph-based protein models where rather than perturbing an existing graph, we generate a new one from scratch that have the same characteristics and semantics as the training set and fools the model.

In addition to graph adversarial attacks and defense schemes, my research on the application of GNNs to different areas such as bioinformatics. Given the protein's form, we can represent them as graphs and therefore applying different approaches such as Graph Neural Networks is possible. For instance, we proposed to represent the 3D structure of a protein (generated using AlphaFold) as a graph and we combined the different graph with other features extracted from Language models applied to their sequences to predict the Antibiotic Resistance. Our proposed model is performing better than available benchmark methods and is faster to train and use.

2 Two concepts from the lectures:

In this section, I will try to connect both the concept of Software Quality and Boundary value testing to my research related to adversarial attacks on GNNs.

Software quality constitutes a sub-domain of the software engineering field focused on scrutinizing the dependability and security of underlying software frameworks. This is highly related to the field of studying GNN's robustness. The latter field revolves around identifying the imperceptible and inconceivable adjustments to input data, capable of inducing a shift in the model's output. Consequently, this manipulation misleads the model into rendering erroneous predictions, thereby shedding light on its susceptibility to adversarial perturbations. A corollary to this is observed in the sphere of software engineering, wherein analogous principles apply. Consider, for instance, the scrutiny applied by safety teams when investigating how software systems react in the face of perturbations affecting their outputs. This facet gains bigger importance in scenarios involving critical software applications, such as those used within the financial domain, where even minor software malfunctions can yield substantial financial repercussions. Consequently, while the attacks in the context of Graph Neural Networks are not directly applicable to the case of software given the difference in input, similar developed concept can be considered and adapted accordingly.

Looking from a different perspective, Boundary Value Testing have an significant role within the broader context of software engineering by focusing on assessing the behavior of values that exist at the boundaries of distinct partitions. This aspect is highly important due to the potential divergence between the normal expected behavior and the actual behavior when confronting values positioned at these boundaries. Similar to adversarial attacks, empirical evidence lends confirm this notion, as practical instances of adversarial points typically emerge in close proximity to the computed boundaries of the model. Additionally, a noteworthy and shared aspect between the domain of adversarial attacks and boundary testing is their reliance on the selection of appropriate metrics to quantify the input and output relationships. In fact, the main assumption is that we consider the input and output space to be metric spaces with a tractable and usable metric defined on the space. In the context of adversarial attacks, an important observation establishes a crucial foundation: the robustness of a model is inherently tied to the metric chosen for evaluation. This observation has lead to showing the equivalence between different metrics, thus paving the way for a generalized approach to quantifying robustness. This result should be interesting to be extended to the domain of Software Boundary Testing. By drawing a parallel, it becomes conceivable that this equivalence could potentially alleviate the complexity of testing procedures. In this context, the adoption of simpler metrics does not necessarily restrict the empirical efficacy of the results, opening solution for more streamlined and efficient testing methodologies.

3 Two ideas from the guest lectures

Automated Machine Learning , often referred to as AutoML in literature, is focusing on automating the process of picking, training, and tweaking models. We can think of it like having a helper that takes care of the technical side of the project. For instance, when it comes to Graph Neural Networks (GNNs), and similar to other Deep Neural Networks, there are many settings like how many layers to use, how quickly the model learns, and more. Picking the right settings is like tuning an instrument for the best performance. Trying out all the possible combinations by hand is very tedious and costly. AutoML offers efficient and optimized techniques that can speed up the process of finding the best settings. For example, when using GNNs to recommend movies to people based on their preferences, with AutoML, the used can specify the important metric to be considered (accuracy for instance), and it will quickly test out different settings until it finds the ones that work best. This saves you time and effort compared to testing everything manually.

Semantic and sentiment analysis is another field related to Graph Neural Networks. The majority of available used techniques are built on text-based methods related Natural Language Processing. Recently, Graph Neural Networks (GNNs) have achieved excellent performance in various NLP tasks. Hence, applying GNNs to sentiment analysis have been investigated recently in the literature specially for online social platforms where the topology of the network is relevant to the task [3, 4]. The use of

GNNs for sentiment analysis is also very relevant for product reviews in marketplaces such as Amazon. This is the case since in these specific cases, the user is interested in detecting aspect and opinion relationships that are not only dependent on the considered product but also takes into account similar product or other product of interest to the considered client giving therefore a global and relevant context-based sentiment analysis.

4 Two topics from the given list:

4.1 Automated Software Testing:

Automated Software Testing is a research field related to automating the process of validating the well execution of software and it meets requirements before being pushed to production. This is very relevant from a point of view of robustness and adversarial attacks in general and also in the context of GNNs. Verifying the robustness of machine learning models is crucial since data can be noisy, incomplete, manipulated by an adversary, or simply different from what was previously observed. Ideally, giving a black-box trained model, we ideally would like to find a method that can directly test its robustness to different attacks and gives an input's maximum-neighborhood in which the model is stable and robust. This is very similar to the work related to robustness certificates that aim to provide provable guarantees that no perturbation regarding a specific threat model will change the prediction of an instance.

4.2 Quality assurance:

Ensuring Software quality is an important field interested in studying the reliability, maintainability, and scalability. This may be considered for the concept evoked in the previous section which is robustness certification. In fact, there exist a trade-off between the certificate's false positives rate and its scalability, where the less false positives are usually related to a higher complexity since these methods should be based on computationally expensive method. Hence, ensuring the quality of these certificates is important to trust their robustness assessments. In this perspective, rigorous testing may help discover potential issues and uncertainties in these certification and their process. Furthermore, method from this domain, specially related to monitoring, may be helpful when dealing with real-world datasets and challenges.

4.3 Areas of Opportunity

Software engineering is essential and important in the modern world. While the actual frameworks and process used have done great achievement, the field can can benefit more from advancements in AI such as large language models and diffusion models, which can generate and produce text and images with high quality. I believe that using tools from NLP to generate code is very relevant and can help enhance the code quality of software. Unfortunately, while this may seem as a thread to software engineers, it is actually an opportunity since it's really hard to substitute engineers but these tools can help alleviate their workload and make their daily tasks easier.

Given the impressive capability of Graph Neural Networks (GNNs) to capture and represent node and graph structures, their applicability extends clearly into the field of software engineering. Imagine a software script as a sequence of algorithms working together in harmony to yield the final desired output of the software. Each algorithm communicates with others, where the output of one becomes the input of the next. This dynamic interplay between algorithms can be effectively transcribed into a graph-based framework. Here, each algorithm is represented as a node in the graph, while the edges are built based on the nature of interactions between these algorithms. By harnessing this distinctive topological representation, different techniques based on GNNs can be employed to extract valuable insights. These insights, in turn, lay the foundation for simplifying and streamlining the algorithmic pipeline and, by extension, the entirety of the script. Moreover, this novel perspective opens the door to an interesting prospect

which is the pre-training of models capable of generating new scripts for diverse tasks. To exemplify, picture a scenario where a GNN-powered model learns the intricacies of existing scripts. Subsequently, using this representation, it becomes proficient at crafting entirely new scripts to accomplish a range of tasks. Hence, the combination of GNNs and software engineering can have two initial advantages: optimizing the existing script infrastructure and empowering the creation of novel, task-specific scripts. This convergence represents the possiblity of using GNNs beyond their conventional domains and a possible reshaping into the landscape of software engineering.

From another perspective, in my research, I think some concept that are related to software testing and boundary value testing may be relevant to my work in adversarial attacks and can be easily applicable and extended to GNNs. I believe that the available methods in the software engineering fields have another perspective which could be very useful and can enhance our understanding of AI model's robustness to adversarial perturbations.

References

[1] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, 2016.

[2] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based Recommendation with Graph Neural Networks," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019, pp. 346–353.

[3] Y. Li and N. Li, "Sentiment analysis of weibo comments based on graph neural network," *IEEE Access*, vol. 10, pp. 23 497–23 510, 2022.

[4] W. Liao, B. Zeng, J. Liu, P. Wei, X. Cheng, and W. Zhang, "Multi-level graph neural network for text sentiment analysis," *Computers Electrical Engineering*, vol. 92, p. 107096, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790621001051