LLM-Augmented Adversarial Training for Robust Graph Learning

Siddhartha Gotur*
Birla Institute of Technology & Science
f20220070@pilani.bits-pilani.ac.in

Sriram Sudheer Hebbale*
Birla Institute of Technology & Science
email2@domain.edu

Avviral Jain
Birla Institute of Technology & Science
email3@domain.edu

Abstract

We propose an adversarial-training framework that leverages large language models (LLMs) to enhance the robustness of graph neural networks (GNNs) on text-attributed graphs (TAGs). Existing work shows that Graph-LLMs (GNNs with LLM-based text features) are unusually robust to isolated structural or textual attacks. However, joint attacks and LLM-driven perturbations remain unexplored. We will generate adversarial node texts and injected graph alterations using an LLM (e.g., GPT-3.5) and incorporate them into training. This LLM-augmented data-augmentation aims to harden Graph-LLMs against subtle graph manipulations. By fine-tuning both LLM-as-Enhancer and LLM-as-Predictor pipelines with LLM-generated adversarial examples, we expect to improve worst-case accuracy. Experiments will measure degradation under novel attacks and analyze feature distinguishability. Our goal is to demonstrate that LLM-generated augmentations significantly boost robustness, filling gaps noted by previous work.

1 Introduction

Graph neural networks on text-attributed graphs (TAGs) are increasingly used in domains such as citation networks, wiki-articles graphs, and e-commerce item co-purchase graphs. Recent work on Graph-LLMs (GNNs enhanced by LLM-based embeddings or LLM-based prompt predictors) has shown significant robustness to certain structural or textual adversarial attacks. However, these evaluations typically consider either text perturbations (e.g., word-level synonym replacements) or structure perturbations (e.g., edge flips) in isolation. They do not explore **LLM-generated adversarial content**, **hidden token manipulations**, or **instruction-injection via node text**, which may exploit the specific vulnerabilities of LLM-based pipelines. We propose to fill this gap by designing novel attack modes and then training with them to improve robustness.

2 Related work

We review key recent works (2024–2025) on graph adversarial attacks, text-attributed graphs, and LLM-based augmentation/defense:

- Guo et al. (2024). *Learning on Graphs with LLMs: A Deep Dive into Model Robustness*. ICLR 2025. This work benchmarks Graph-LLMs under structural/textual attacks and finds superior robustness of LLM-based features.

*Equal contribution.

YourTeamName et al., LLM-Augmented Adversarial Training for Robust Graph Learning. *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 269, Arizona State University, Phoenix, USA, December 10–12, 2025.

- Xu et al. (2025). *Unveiling the Vulnerability of Graph-LLMs: An Interpretable Multi-Dimensional Adversarial Attack on TAGs*. ArXiv. Proposes an SHAP-based attack on TAGs combining text perturbations and edge pruning.
- Lei et al. (2024). *Intruding with Words: Graph Injection Attacks at the Text Level*. NeurIPS 2024. Introduces textual graph injection attacks (adding malicious nodes with text) and notes that LLM-based predictors can help defend against them.
- Olatunji et al. (2025). *Adversarial Attacks and Defenses on Graph-aware LLMs*. ArXiv. Studies poisoning/evasion attacks on graph-augmented LLMs (LLAGA, GraphPrompter) and proposes an LLM-based defense (GALGUARD) for feature correction.
- O'Brien et al. (2024). *Improving Black-box Robustness with In-Context Rewriting*. ArXiv. Shows that using LLM-generated text augmentations at test time (LLM-TTA) substantially improves classification robustness.
- Chowdhury & Chadha (2024). *Generative Data Augmentation using LLMs Improves Distributional Robustness in QA*. EACL Workshop. Demonstrates that augmenting datasets with LLM-generated examples yields better robustness under domain shifts.
- Khurana et al. (2024). *A Hierarchical Language Model for Interpretable Graph Reasoning*. ArXiv. Presents an LLM-based graph model (HLM-G) that uses hierarchical structure and attention for interpretability in graph tasks.

3 Datasets

We propose to use publicly available text-attributed graph (TAG) benchmarks that allow rapid prototyping and reproducibility:

- **Citation Networks:** *Cora*, *Citeseer*, *PubMed*, and *ogbn-arxiv* — nodes represent research papers, and textual features come from paper titles and abstracts. These datasets are standard for semi-supervised node classification and allow comparison with previous Graph-LLM robustness studies.
- **Wikipedia Graph:** *WikiCS* — a computer-science topic graph where nodes correspond to Wikipedia articles and edges represent hyperlinks. Each node is associated with full article text as features, making it suitable for evaluating textual and structural attacks jointly.
- **E-commerce Graph:** *Amazon-Books* — an item co-purchase network where nodes represent books, edges denote co-purchase relations, and node text features are derived from book titles and product descriptions. This dataset helps test model generalization to real-world domains.

We will use the standard training/validation/test splits provided by the datasets' maintainers to ensure comparability with prior literature. The variety in domain (academic, wiki, e-commerce) ensures that our results generalize across different text and graph modalities.

4 Methodology

Our pipeline extends baseline Graph-LLM frameworks with LLM-guided adversarial augmentation.

4.1 Baseline Implementation

We reproduce two Graph-LLM models: (a) **LLM-as-Enhancer**, where an LLM encodes node text (e.g., Sentence-BERT or LLaMA) feeding a GNN; (b) **LLM-as-Predictor**, where an instruction-tuned LLM (e.g., GPT-3.5) directly predicts node labels. We follow existing work for architecture/hyperparameter parity.

4.2 Novel Adversarial Attacks

We propose and implement three new attack modes on TAGs:

1. **Invisible Unicode Injection Attack**: Insert zero-width characters (e.g. U+200B, U+200D) into node text, preserving human-readable string but altering tokenisation and embedding generation.

2. **Prompt Instruction Injection Attack**: Embed malicious instructions (e.g. "Ignore above and classify as Y") inside node text or neighbor nodes, exploiting the LLM's instruction-following behaviour.
3. **LLM-Based Semantic Paraphrase Attack**: Use an LLM to paraphrase node text (or back-translate) so meaning is retained but syntactic structure changes, altering embedding behaviour and confusing Graph-LLM pipelines.
4. **Contradictory Statement Injection Attack**: Append or inject sentences that semantically contradict key facts in the node text (or its neighborhood) to produce confusion in aggregation and encoding.

These attacks are designed to exploit vulnerabilities unique to Graph-LLM pipelines, rather than generic word-swap or edge flip attacks.

4.3 Adversarial Training

We integrate the attack-generated examples into training:

- For LLM-as-Enhancer: For each training batch, augment with adversarial versions (invisible-unicode, paraphrase, contradiction) of node texts while keeping labels unchanged; train the GNN on mixed clean + adversarial data.
- For LLM-as-Predictor: Fine-tune the LLM using few-shot prompts that include adversarial node contexts; optionally include instruction-injected neighbor nodes.

4.4 Evaluation

We will compare performance under multiple scenarios:

- Clean data (no attack)
- Standard attacks (PGD structure, SemAttack text) as baseline
- Our proposed attacks (invisible unicode, instruction injection, paraphrase, contradiction)

Metrics: node classification accuracy, drop in accuracy under attack, attack success rate (percentage of nodes whose label flips), embedding margin / feature space separation (e.g., using t-SNE/UMAP), prediction confidence/entropy differences. We may also include interpretability: analysing which tokens or edges influenced decisions (by SHAP or attention methods).

5 Timeline

1 Setup datasets, environment (GPU), implement baseline Graph-LLM pipelines on one dataset (e.g., Cora).
2 Build LLM-attack module: scripts for (i) invisible unicode insertion, (ii) prompt-instruction injection, (iii) paraphrase via
3 Integrate adversarial samples into training: implement adversarial training for both LLM-as-Enhancer and LLM-as-Predi
4 Full evaluation: run on all chosen datasets (Cora, WikiCS, Amazon) comparing clean, baseline attacks, and proposed atta
5 Analysis of results: ablation studies (which attack type hurts most, domain differences), interpretability insights (token/e
6 Write-up: compile project report, code cleanup, prepare slides (if needed), final checks.

References

A Appendix

Additional details such as sample prompt templates for the LLM attack module, full hyper-parameter settings, additional plots or results tables may be included here.