FLAG: Fraud Detection with LLM-enhanced Graph Neural Network

Chengdong Yang
13811080302@163.com
Beijing University of Posts and Telecommunications
Beijing, China

Hongrui Liu
wlgoodboys@163.com
Independent Researcher
Beijing, China

Daixin Wang
daixinwang@qq.com
Independent Researcher
Beijing, China

Zhiqiang Zhang
zzqsmall@gmail.com
Independent Researcher
Hangzhou, China

Cheng Yang
yangcheng@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Chuan Shi*
shichuan@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Abstract

Graph-based methods have proven effective in financial fraud detection by modeling relationships between entities, yet they often fail to leverage the rich textual information present in real-world data. With the ability to understand semantic information, large language models (LLMs) offer a promising solution to enhance fraud detection by incorporating textual data, such as user profiles and transaction descriptions. However, integrating LLMs with graph-based methods introduces two key challenges: (1) the neighborhood camouflage problem, where fraudulent nodes disguise themselves within normal network structures, and (2) the input size constraints of LLMs, making it difficult to process large, complex graphs with extensive textual data. In this paper, we propose a novel framework, Fraud Detection with LLM-enhanced Graph Neural Networks (FLAG), to address these challenges. FLAG integrates LLMs with graph-based fraud detection by introducing two main modules: semantic similarity neighbor sampling, which reduces the input size and further alleviates the influence of camouflaged neighbors by selecting neighbors having high semantic similarity with the target nodes, and LLM-based node enhancement, which extracts discriminative textual features by LLM to enhance node robustness against camouflaged neighbors. To further improve the model, we design a fine-tuning approach that enables the LLM to extract discriminative text more closely related to the node labels, enhancing the model's ability to differentiate between fraudulent and normal nodes. Extensive experiments on public datasets highlight the superiority of FLAG, showing average improvements of 3.14% in F1-macro and 6.97% in AUC. Furthermore, we have deployed FLAG in Alipay's credit risk assessment system and evaluated its performance on a real-world dataset. The results indicate a 0.9% improvement in the KS criterion, further underscoring FLAG's effectiveness.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**.

Keywords

Fraud Detection, Graph Neural Networks, Large Language Model

ACM Reference Format:
Chengdong Yang, Hongrui Liu, Daixin Wang, Zhiqiang Zhang, Cheng Yang, and Chuan Shi. 2025. FLAG: Fraud Detection with LLM-enhanced Graph Neural Network. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3737220

1 Introduction

With the rapid development of Internet services, fraudulent activities have become increasingly sophisticated and widespread [2, 19]. Fraudsters often disguise themselves as legitimate users to bypass antifraud mechanisms, spread misinformation, or exploit sensitive user data. Graph-based methods have emerged as an effective approach for financial fraud detection, modeling entities and their relationships in a graph to identify suspicious patterns [35, 42]. These methods are particularly effective when fraudsters form distinct structures, such as clusters, within the graph.

Recently, large language models (LLMs) have shown remarkable abilities in understanding textual semantics [4, 38], leading to the integration of LLMs with graphs for various tasks on text-rich graphs [5, 21]. For example, in the recommendation domain, LLM-Rec [43] utilizes text-generated profiles to filter noisy interactions and enrich node embeddings which significantly improves recommendation performance. Similarly, in graph contrastive learning, GAugLLM [10] leverages LLMs to generate enhanced features and graph structures, addressing the limitations of traditional methods that overlook textual semantics. These advancements demonstrate the potential of LLMs in improving graph-based approaches. In the context of financial fraud detection, textual data, such as user profiles or transaction descriptions, also provides critical information for identifying fraudulent behaviors [3, 15, 45]. However, existing

*Corresponding author

KDD '25, August 3–7, 2025, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737220

methods largely fail to incorporate the expressive textual information into their frameworks. This raises a crucial question: *how can we effectively integrate LLMs with graph-based methods to enhance fraud detection in text-rich graphs?*

Integrating LLMs with fraud detection on graphs presents unique challenges. **Firstly**, integrating LLMs with graphs can exacerbate the *neighborhood camouflage* problem, which is inherent to graph-based fraud detection tasks. Specifically, as shown in Figure 1, fraudsters often camouflage themselves by employing sophisticated tactics to mimic regular network patterns [27, 36], *e.g.*, interacting with normal users in the social networks or transferring funds to normal accounts in transaction networks. This behavior conceals fraudulent nodes within normal heterophilous neighborhoods, resulting in heterophily for fraudulent nodes and homophily for normal ones, which violates the homophily [12] inductive bias of GNNs. Current methods that combine LLMs with graph-based tasks struggle to address this challenge. For instance, InstructGLM [47] uses a *LLMs-as-Predictors* approach with instruction prompts to predict node labels. However, due to the dataset's dominance of homophilous normal nodes, it struggles to identify anomalous, heterophilous nodes. GAugLLM adopts an *LLMs-as-Enhancers* strategy, generating explanations from node text and graph structure to enhance the GNN predictor. However, the effectiveness of the LLM is limited by the GNN predictor, especially when the GNN is inherently homophilic, hindering the LLM's performance gains. Similarly, GraphTranslator [49] aligns graph models with LLMs through a translator module, which faces challenges in fraud detection when aligning LLMs to homophilic GNNs. More details about related work can be found in Appendix B. **Secondly**, LLMs have inherent input size constraints [1, 39], making it infeasible to process the whole ego-graph directly, especially when dealing with highly connected nodes and extensive textual data. Particularly, financial fraud detection graphs often exhibit complex relationships and extensive textual attributes [24, 32], necessitating strategies to reduce the input size while preserving critical information. However, most current LLM-based gnns use sampling strategies like GraphSAGE [16]'s random sampling [14, 44], but could lose valuable homophilic neighbors and exacerbate the neighbor camouflage issue.

In this paper, we propose a novel framework Fraud Detection with LLM-enhanced Graph Neural Network (FLAG) designed to integrate LLM into graph-based fraud detection while addressing the aforementioned challenges. The framework consists of two key modules: semantic similarity neighbor sampling and LLM-based node enhancement. The semantic similarity neighbor sampling module reduces the influence of camouflaged neighbors by selecting neighbors having high semantic similarity with the target nodes. This approach ensures that the node's neighborhood retains critical connections while filtering out those that might dilute its representation, particularly for fraud nodes camouflaged by normal neighbors. Moreover, this sampling strategy helps reduce the input size, facilitating the application of the LLM. To further enhance the robustness against camouflaged neighbors, we introduce an LLM-based node enhancement strategy to extract *discriminative text*—the portions of a node's text most strongly correlated with its label. This discriminative text is then encoded using a skip-GNN, where we introduce a skip connection into the traditional GNN to enhance

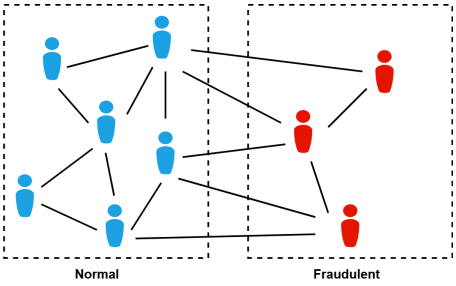


Figure 1: The illustration of neighborhood camouflage.

personalized information. To optimize the LLM's ability to extract such discriminative text, we design an auxiliary task to generate residual text, which complements discriminative text by being less correlated with the label. By encouraging the discriminative text to exhibit stronger label relevance and the residual text to remain label-independent, LLM effectively removes noisy information from raw text and highlights critical discriminative features. Extensive experiments on the public datasets validate the effectiveness of FLAG in fraud detection, with average improvements of 3.14% in F1-macro and 6.97% in AUC. In light of the superior performance, we have deployed FLAG in Alipay's credit risk assessment system and evaluated its performance on a real-world dataset. The results indicate a 0.9% improvement in the KS criterion, 1.4% in F1-macro and 0.47% in AUC, further underscoring FLAG's effectiveness. In summary, the main contributions are highlighted as follows:

- We are the first to integrate LLM with GNN for fraud detection, utilizing textual semantic information to enhance GNN.
- To address the challenges of neighborhood camouflage and input size constraints, we design two modules: semantic similarity neighbor sampling to filter camouflaged neighbors while reducing the input size and LLM-based node enhancement to extract discriminative text for enhancing node robustness. Additionally, we fine-tune the LLM to improve its performance, making nodes more distinguishable in fraud detection.
- Extensive experiments on public and industry datasets demonstrate the superiority of FLAG, in detecting fraudulent nodes, especially on text-rich graphs.

2 Preliminary

2.1 Graph Neural Network

Consider a text graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ with node texts $\mathcal{T}$, where $\mathcal{V}$ represents the set of nodes, $\mathcal{E}$ represents the set of edges, and $\mathcal{X}$ represents the node features. Each node $v \in \mathcal{V}$ is associated with a feature vector $x_v \in \mathcal{X}$ and ground-truth label $y_v \in \mathcal{Y} = \{1, ..., K\}$. Each edge $(v, u) \in \mathcal{E}$ represents the relation between v and u.

GNN is a parametric model designed to compute node representations by aggregating information from neighbors. At each layer l, the representation h_v^l of node v will be updated by combining the representations of v and its neighbors in the previous layer:

$$h_v^l = \psi^l(h_v^{l-1}, \bigoplus_{u \in \mathcal{N}(v)} \phi^l(h_v^{l-1}, h_u^{l-1})), \tag{1}$$

where $\mathcal{N}(v) = \{u | (u, v) \in \mathcal{E}\}$ is the neighbor set of node v, $\oplus$ denotes a differentiable, permutation-invariant function (*e.g.*, sum,

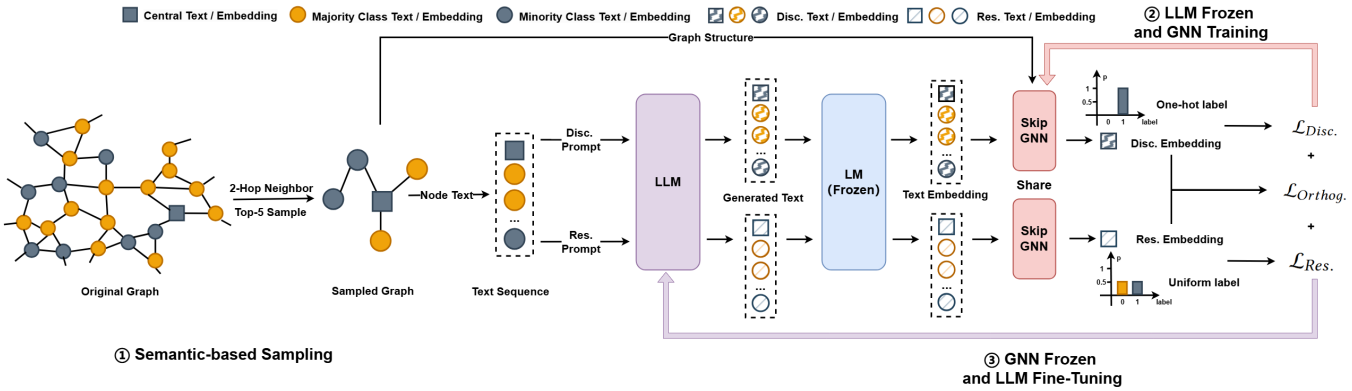


Figure 2: The framework of the proposed FLAG. Starting with a target node in the original graph, a k-hop subgraph is sampled using a semantic similarity neighbor sampling. The text of all nodes in the subgraph is concatenated into a text sequence, which is then processed by the LLM to extract both discriminative and residual texts. These texts are encoded with a frozen LM and, along with the subgraph structure, passed through shared-parameter skip-GNN modules to obtain target node representations. The framework is optimized using three loss functions: Disc. Loss, aligning discriminative text with one-hot labels; Res. Loss, aligning residual text with uniform distribution; and Orthog. Loss, ensuring the two representations remain orthogonal. Additionally, we give examples of discriminative and residual texts extracted from the original text in the Appendix C.

mean or max) and ψ, ϕ denote differentiable transformation functions such as multi-layer perceptrons (MLPs). The initial representation h_v^0 is the raw feature vector x_v. The final label predictions from a GNN are given by:

$$Z = \text{softmax}(\text{GNN}_\Theta(A, X)) \in \mathbb{R}^{|\mathcal{V}| \times K}, \tag{2}$$

where $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is the adjacency matrix, Θ denotes the set of trainable parameters, and each row of matrix Z corresponds the predicted label distribution of a specific node.

2.2 Neighborhood camouflage

In this paper, neighborhood camouflage refers to the phenomenon that the network exhibits distinguish patterns for various classes. The phenomenon often arises in the graph-based fraud detection scenario, which is a highly imbalanced binary classification task. In this scenario, as shown in Figure 1, the fraudulent users actively or passively camouflage themselves by interacting with normal nodes, leading to heterophily, while the normal users remains homophily. The neighborhood camouflage violates the general homophily assumption, becoming a significant challenge for fraud detection.

3 Method

In this section, we propose FLAG, a novel approach designed for fraud detection by integrating LLMs with graph-based methods. FLAG consists of two main components: a semantic similarity neighbor sampling module and an LLM-based node enhancement module. Given a text graph, FLAG first employs semantic similarity neighbor sampling to reduce input size and the influence of heterophily neighbors. The sampled graph is then processed by the LLM-based node enhancement module, which extracts discriminative text expected to strongly correlate with the associated labels. To further enhance the LLM's ability to extract such text, we propose an effective fine-tuning strategy aimed at improving the correlation between discriminative text and labels. Figure 2 illustrates the overall framework of our fraud detection method.

3.1 Semantic Similarity Neighbor Sampling

Given the increasing size of text and relationships in the industry [23], it is necessary to perform effective sampling from the entire text graph. Most current LLM-based graph methods use sampling strategies like GraphSAGE's random sampling [14, 44], but could lose valuable homophilic neighbors and exacerbate the neighbor camouflage issue. Moreover, previous methods for tackling neighbor camouflage also use sampling techniques, but they do not leverage the semantic information contained in the node text, which limits their ability to capture more meaningful relationships and contextual nuances. Therefore, we propose a sampling method that utilizes semantic information to selectively focus on the most relevant neighbors, thereby reducing the influence of irrelevant normal nodes and mitigating neighbor camouflage.

Mathematically, for each node $v \in \mathcal{V}$, with the pre-trained language models $\mathcal{B}$, we first obtain its representation of the text t_v. Then the semantic similarity between the node v and its neighbor $u \in \mathcal{N}(v)$ within a k-hop subgraph centered around v is defined by cosine similarity:

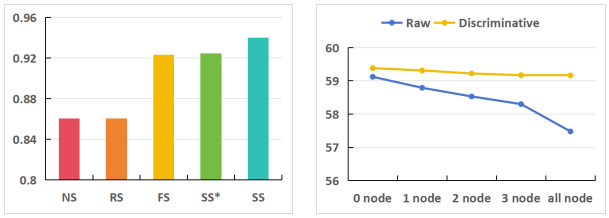
$$sim(v, u) = \frac{\mathcal{B}(t_v) \cdot \mathcal{B}(v_u)}{||\mathcal{B}(t_v)||\ ||\mathcal{B}(v_u)||}. \tag{3}$$

Afterwards, we rank the neighboring nodes based on their similarity to the center node v's text t_v. We then select the top-N neighbors $\mathcal{N}_{selected}(v)$ with the highest similarity that exceed a predefined threshold δ:

$$\mathcal{N}_{selected}(v) = \{u \in \mathcal{N}(v) \mid sim(v, u) \geq \delta\}. \tag{4}$$

The set of selected neighbors $\mathcal{N}_{selected}(v)$ is finally used to form a simplified subgraph $\mathcal{G}_v$ centered around node v.

To evaluate the effectiveness of our proposed text-based neighbor sampling strategy, we perform comparative experiments on Reddit [25] to compare the average subgraph edge homophily [51] with other sampling strategies, including no-sampling (NS), random sampling (RS), shallow feature similarity-based sampling (FS), semantic similarity sampling without threshold (SS*), and our proposed semantic similarity sampling (SS). Specifically, FS calculates



(a) Comparison of subgraph homophily across sampling strategies.

(b) Stability of discriminative text vs. raw text.

Figure 3: Semantic similarity neighbor sampling and LLM-based node enhancement for fraud detection.

the similarity between the central node and its neighbors based on shallow features, and then selects the top-N neighbors to form the final subgraph. SS*, compared to FS, replaces shallow features [30] with text-based features for similarity calculation. Detailed sampling settings are provided in Section 4.1. Moreover, the average subgraph edge homophily measures the fraction of edges in a graph which connects nodes that have the same class label, which is formalized as follows:

$$homo = \frac{\sum_{(u,v)\in\mathcal{E}} \mathbb{I}(y(u) = y(v))}{|\mathcal{E}|}. \tag{5}$$

where $\mathbb{I}(y(u) = y(v))$ is an indicator function that is 1 if the class labels of nodes u and v are the same, and 0 otherwise; $|\mathcal{E}|$ is the total number of edges in the graph.

The results are visualized in Figure 3(a). We observe that the homophily of RS is similar to that of the NS baseline. FS performs slightly better, but still falls behind SS*. And homophily of SS* is lower than that of our proposed SS. This demonstrates that: (1) RS is ineffective in addressing neighbor camouflage; (2) the raw node text contains more complete information than the shallow feature; and (3) adopting a threshold-based filtering strategy is beneficial for preserving critical information.

3.2 LLM-Based Node Enhancement

To further counter the effects of neighbor camouflage, we focus on enhancing the robustness of the node itself. In traditional GNNs, during feature aggregation, the features from neighboring normal nodes can overwhelm the features of a fraudulent node, causing a loss of the node's distinct identity. By strengthening the connection between a node's features and its label, we can more effectively combat neighbor camouflage. To achieve this, we utilize an LLM to extract *discriminative text*—the portions of a node's text most strongly correlated with its label. Additionally, we implement a skip-GNN that incorporates a bypass from the node features to the final representation to increase the personalized information.

3.2.1 Discriminative text extraction. To extract this discriminative text, we turn to LLM, which are well-suited for understanding and distilling meaningful semantic information from text. LLMs can identify and retain the most label-relevant aspects of a node's text, filtering out irrelevant text. Specifically, we apply the LLM to distill the raw node text t_v into a discriminative representation t_v^D that highlights the node's intrinsic properties, enabling a clearer distinction between fraudulent and normal nodes. Table 1 illustrates the format and an example of the prompt used for extracting discriminative text from node information.

Table 1: Format and example of the discriminative text prompt on Reddit dataset.

Disc. Prompt:
<Dataset Description>. <Task Description>. <Ensurance>.
Example:
You are provided with a list of Reddit users' posts. Each user is classified as either popular or normal based on their interactions and content. Your task is to generate a brief discriminative text for each user that directly relates to distinguishing them as either popular or normal. Ensure that the generated text highlights specific features that help differentiate the user's classification while avoiding any general or irrelevant information.

To validate the correlation between labels and the LLM-generated discriminative text, we compare its stability against the raw text under neighbor perturbations on Reddit [25]. Intuitively, for each targeted node, if its feature is strongly correlated with the label, it should still maintain superior performance even if most of its neighbors are replaced. Put simply, we first extract discriminative text and train two GNN models—one using the raw text and the other using discriminative text. We then perturb the test set by replacing the text of same-class neighbors with randomly selected text from nodes of the opposite class. We use the GCN model for this experiment, and detailed experimental settings are provided in the Section 4.1. The model's performance is finally evaluated on these disturbed datasets.

Results shown in Figure 3(b) indicate that, as node heterophily increases, the AUC of the model using raw text declines significantly after GNN aggregation, while the model using discriminative text remains more stable. This demonstrates that discriminative text is robustness against neighbor perturbation and strongly correlated with the node's label.

3.2.2 Skip-GNN. To further address the issue of heterophily of fraudulent users, we incorporate skip connections into the GNN. The skip connections allow the node feature to be included in the final representation without passing through the aggregation process, ensuring that the node's label-relevant features are preserved. This is crucial for distinguishing fraudulent nodes from normal nodes, as it helps prevent the "dilution" of the fraudulent node's features by its neighbors.

In the proposed skip-connected GNN, we add a direct connection to the node feature $\mathcal{X}$, so that the final node representation $\mathcal{Z}$ is a combination of both the aggregated features and the node features:

$$\mathcal{Z} = \text{GNN}(\mathcal{X}, \mathcal{A}) + \text{Linear}(\mathcal{X}). \tag{6}$$

This ensures that the node's own features are preserved, providing more stable representations for nodes, particularly fraudulent ones, even in the presence of neighborhood dilution.

3.3 Fine-Tuning Strategy

To enhance the discriminative power of the extracted text features, we propose a fine-tuning strategy for the LLM that is specifically tailored to the task of fraud detection in graph data. The core idea behind this approach is to introduce an auxiliary task that focuses on extracting the residual text—the portion of node text that is

complementary to the discriminative text and less relevant to the label. Since the discriminative and residual texts are inherently complementary, enhancing the model's ability to accurately extract residual text helps improve its capacity to extract discriminative text. Based on this complementary relationship, we design a process that optimizes both tasks simultaneously.

The fine-tuning process involves generating two distinct types of text for each node by utilizing separate prompts: one to produce discriminative text t_v^D and the other to generate residual text t_v^R. The format and an example of the prompt for residual text is shown in Table 2. The generated texts are then encoded into respective representations (x_v^D and x_v^R) using a fixed language model, and subsequently input by two shared-parameter skip-GNNs along with the corresponding graph structure. We finally obtain representations for discriminative text and residual text, respectively denoted as z_v^D and z_v^R. To supervise the fine-tuning, we introduce three loss functions, each focusing on different aspects of the task.

Table 2: Format and example of the residual text prompt on Reddit dataset.

Res. Prompt:
<Dataset Description>. <Task Description>. <Ensurance>.
Example:
You are provided with a list of Reddit users' posts. Each user is classified as either popular or normal based on their interactions and content. Your task is to generate a brief residual text for each user that captures the portions of their text that do not directly relate to distinguishing them as either popular or normal. The residual text should highlight non-discriminative features that are not helpful in differentiating the user's classification. Focus on generating text that includes general background details, unrelated observations, or information that does not influence the classification task.

The Discriminative Text Loss encourages the LLM to generate text strongly correlated with the node's label, improving classification performance. This is represented by Binary Cross-Entropy (BCE):

$$\mathcal{L}_{Disc.} = -\sum_{i \in \mathcal{V}} [y_i log(p_i) + (1 - y_i) log(1 - p_i)] \tag{7}$$

where y_i is the true label of node i (fraudulent or normal), and p_i is the predicted probability that node i is fraudulent. This loss ensures that the discriminative text generated by the LLM is aligned with the node's class.

The Residual Text Loss ensures that the residual text is as unrelated as possible to the node's label. This is modeled using Kullback-Leibler (KL) divergence:

$$\mathcal{L}_{Res.} = \sum_{i \in \mathcal{V}} D_{\mathrm{KL}}(\hat{p}_i || \hat{u}_i) = \sum_{i \in \mathcal{V}} \sum_{c \in \{0,1\}} \hat{p}_i(c) \log \left(\frac{\hat{p}_i(c)}{\hat{u}_i(c)} \right) \tag{8}$$

where $\hat{p}_i$ is the predicted probability distribution of node i generated by the LLM, and $\hat{u}_i$ is the uniform distribution over the classes (i.e., $\hat{u}_i = \frac{1}{C}$, with C being the number of classes). This loss encourages the model to produce text representations that do not over-rely on contextual features.

The Orthogonality Loss ensures the representations of discriminative and residual texts is distinct by minimizing their similarity:

$$\mathcal{L}_{Orthog.} = ||\mathcal{Z}_D \cdot \mathcal{Z}_R||_2^2 \tag{9}$$

where $\mathcal{Z}_D$ and $\mathcal{Z}_R$ are the representation matrices of the discriminative and residual text, respectively, and $\cdot$ denotes the dot product. This loss ensures that the two text representations are orthogonal, thereby preventing any interference between them.

The final loss function, which combines all three components, is defined as:

$$\mathcal{L} = \mathcal{L}_{Disc.} + \lambda_1 \cdot \mathcal{L}_{Res.} + \lambda_2 \cdot \mathcal{L}_{Orthog.} \tag{10}$$

where $\lambda 1$ and $\lambda 2$ are the hyper-parameters that control the contribution of each loss term. The goal is to minimize this total loss to fine-tune the LLM, ensuring that it generates effective and accurate discriminative text representations.

To ensure stable and effective optimization, we employ an alternating two-stage approach. In the first stage, the LLM is fixed while the skip-GNN is trained on the generated text and graph structure. In the second stage, the GNN is fixed, and the LLM is fine-tuned to better distinguish between discriminative and residual text. This alternating approach ensures both models iteratively improve, enhancing fraud detection performance.

3.4 Inference Phase

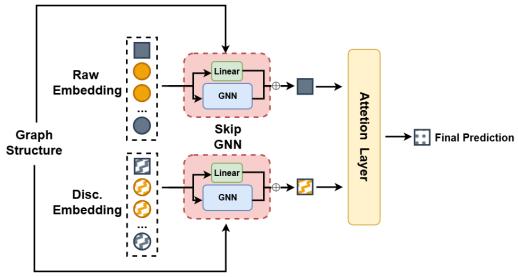


Figure 4: Inference phase workflow integrating raw and discriminative text via skip-GNN and attention mechanism.

In inference phase as shown in Figure 4, we utilize the fine-tuned LLM to extract discriminative text from each node's raw text. Then both the raw text and the discriminative text are encoded into text feature vectors using a frozen LM. These encoded features are independently processed through two shared-parameter skip-GNN modules. The outputs of the two skip-GNN modules are fed into an attention layer. The attention mechanism dynamically weighs the contributions of the raw text and discriminative text representations, generating a fused node representation that integrates information from both sources. The fused node representation is the final prediction. The pseudo code of FLAG is shown in Appendix A.

4 Experiment

In this section, we evaluate the performance of FLAG for financial fraud detection using both public and industry datasets. Specifically, we design a series of experiments to address the following research questions: **Q1:** How does FLAG perform under zero-shot and fine-tuning settings? **Q2:** How do the individual components of FLAG contribute to the overall performance? **Q3:** How does FLAG influence the distribution of node embeddings in the feature

space? **Q4:** What is the impact of hyper-parameters on the final performance of FLAG?

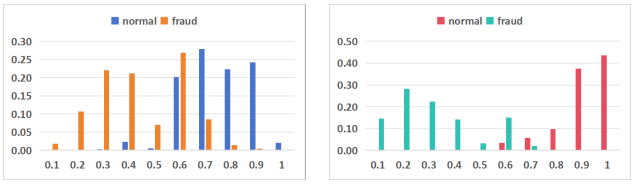
4.1 Experimental Setup

4.1.1 Datasets. In this paper, we utilize two types of datasets for our experiments. The first dataset is a real-world industrial dataset named *Huabei*, derived from the Huabei financial product in Alipay. This dataset comprises 13 million nodes, each representing a unique Alipay user. The associated attributes include user behaviors such as transfers, transactions, and credit activities. The constructed graph consists of 120 million edges, illustrating social connections between users. Additionally, the dataset is labeled based on whether users have overdue payments.

Additionally, for the public fraud detection datasets, we note that most of them, such as Yelp-Fraud [8], Amazon-Fraud [8], T-Finance [36] and T-Social [36], lack textual information, which poses a challenge for applying fraud detection methods that leverage rich-text graphs. To overcome this limitation, we construct a dataset tailored to fraud detection by utilizing existing social network datasets that contain textual data. Specifically, we use two social network datasets: Reddit [25] and Instagram [25]. The Reddit dataset consists of users as nodes and edges representing replies between users. The raw text associated with each node is derived from the content of the user's last three posts on Reddit. Each node is labeled as either popular or normal, based on the user's social interactions and engagement. The Instagram dataset consists of users as nodes and edges representing following relationships. The raw text for each node is derived from the user's personal introduction. Each node is labeled as either commercial or normal, based on the user's account type. Both datasets originally contain an approximately equal number of nodes in each class, but fraud detection tasks generally require imbalanced datasets. To address this, we treat the popular category in Reddit and the commercial category in Instagram as the minority class. The minority class nodes are randomly selected so that the final ratio between the minority and majority classes is about 1:10.

To validate that the datasets we constructed are well-suited for fraud detection tasks, we calculate the homophily score of every node on Reddit and Instagram, defined as the fraction of edges connected to nodes of the same class. The homophily scores are divided into ten bins, ranging from 0.1 to 1.0, to represent varying levels of homophily. For both normal and fraudulent nodes, we compute the proportion of nodes within each homophily bin relative to the total number of nodes in their respective classes. This analysis provides a detailed view of the neighborhood structure for each class, allowing us to assess whether the constructed datasets reflect the expected characteristics of fraud detection tasks.

The results in Figure 5 reveal a clear distinction in the homophily distributions of normal and fraudulent nodes across both datasets. Fraudulent nodes are predominantly found in lower homophily bins, indicating that they are often connected to normal nodes. In contrast, normal nodes are concentrated in higher homophily bins, confirming their tendency to form connections within their own class. This trend is especially pronounced in the Instagram dataset. Furthermore, similar experiments [35] conducted on the non-textual fraud detection datasets, Yelp-Fraud and Amazon-Fraud, also show



(a) Reddit. (b) Instagram.

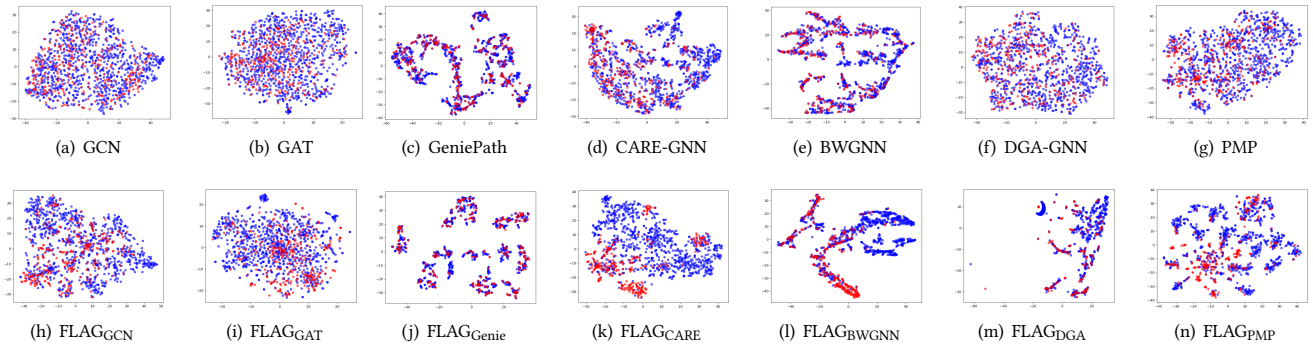
Figure 5: Homophily distributions of fraudulent and normal nodes on Reddit and Instagram.

comparable patterns, confirming that the network structures of the datasets we constructed are suitable for fraud detection tasks.

4.1.2 Baselines and Implementation. We compare FLAG against seven baseline GNN models, categorized into two groups: traditional GNN models (GCN [22] and GAT [40]) and GNN models specifically designed for fraud detection tasks (GeniePath [28], CARE-GNN [8], BWGNN [36], DGA-GNN [9], and PMP [52]). We follow the experimental setup used in BWGNN [36], using F1-macro and AUC [7] as evaluation metrics for public datasets. The model with the highest F1-macro score on the validation set is selected for testing on the test set. Additionally, we employ the Kolmogorov-Smirnov (KS) statistic as supplement metric, which is a widely used metric in the financial industry, particularly for evaluating the performance of credit scoring models. Mathematically, KS can be calculated by $KS = \max |F_1(x) - F_2(x)|$, which measures the maximum difference between the cumulative distribution functions of two sample distributions—typically the distribution of scores for the positive (e.g., defaulters, $F_1(x)$) and negative (e.g., non-defaulters $F_2(x)$) classes.

All models are configured with two layers and a hidden layer size of 64. To enable testing on graphs of varying sizes, we evaluate the models on 2-hop subgraphs of each node, employing a training strategy similar to GraphSAGE. During training, we accumulate the loss of 10 subgraphs before backpropagation. For sampling, we set the similarity threshold to 0 and select the top-10 most similar nodes to form each subgraph. We use the Adam optimizer with a fixed learning rate of 0.01, employing early stopping to prevent overfitting. For the public datasets, each model is evaluated 25 times with 5 random seeds and 5 random initializations to ensure robustness. For the industrial dataset, we conducted the training process only once instead of repeating it to report average performance due to the time constraint. Industry practice has also shown us that results from a single training session on large industrial datasets can be reliable. We use Gemma-9b-it [37] as LLM model with LoRA [18] used for fine-tuning and Sentence-BERT [34] as LM.

4.1.3 Model Deployment. The LLM fine-tuned in Section 3.3 has been deployed for use on Alipay, utilizing an A100 GPU with 80GB of memory. To ensure optimal online inference performance, the deployment uses a total of 8 GPUs across both preproduction and online environments. Notably, our model fully meets the needs of monthly online prediction updates. Additionally, the GNN module is deployed on a CPU cluster, capable of generating predictions within a day.



(a) GCN (b) GAT (c) GeniePath (d) CARE-GNN (e) BWGNN (f) DGA-GNN (g) PMP

(h) FLAG_{GCN} (i) FLAG_{GAT} (j) $\text{FLAG}_{\text{Genie}}$ (k) $\text{FLAG}_{\text{CARE}}$ (l) $\text{FLAG}_{\text{BWGNN}}$ (m) FLAG_{DGA} (n) FLAG_{PMP}

Figure 6: Visualization. The red and blue nodes represent fraud nodes and normal nodes respectively.

Table 3: The performance of FLAG on the industrial dataset (KS, AUC, F1-macro in percentage). GNN† refers to GNN trained using raw text embeddings and FLAG* corresponds to the performance of FLAG after fine-tuning.

Datasets	Huabei		
Metric	KS	F1-macro	AUC
GCN†	76.42	22.76	94.80
GAT†	76.59	22.89	94.87
Geniepath†	76.91	23.28	94.97
CARE-GNN†	76.92	23.28	94.98
BWGNN†	76.85	23.12	94.93
DGA-GNN†	76.84	23.15	94.92
PMP†	76.96	23.32	95.02
FLAG	77.78	24.48	95.41
FLAG*	**77.86**	**24.72**	**95.49**

4.2 Main Result (Q1)

The experimental results for both the industrial and public datasets are presented in Table 3 and Table 4, respectively. Notably, for the public datasets, we apply FLAG to a variety of GNN backbones to demonstrate its model-agnostic nature and its potential as a fundamental data augmentation strategy across different GNNs. For clarity, we employ the following notations to convey their respective meanings: "baseline" indicates the vanilla GNN model utilizing shallow embeddings. The notation "+text" refers to the model trained with raw text embeddings. "+FLAG" signifies the performance of FLAG in a zero-shot context, whereas "+FLAG*" represents the performance of FLAG after fine-tuning. In contrast, for the industrial dataset, we fix the backbone to GeniePath [28] due to its resource efficiency and the stability it provides for month-to-month predictions. We have the following observations:

- Our proposed FLAG consistently outperforms all the baselines by a considerable margin across most cases of all the datasets. For instance, on the industrial dataset, FLAG achieves improvement of 0.9% in KS, 1.4% in F1-macro and 0.47% in AUC on industry datasets compared with its backbone Geniepath. Additionally, on the public dataset, FLAG obtain the average gains of 3.14% in F1-macro and 6.97% in AUC. These results collectively highlight the effectiveness of FLAG through its efficient sampling approach and the integration of LLM for extracting discriminative text to enhance its robustness against camouflaged neighbors.

Table 4: The performance of FLAG with different backbones on Reddit and Instagram (AUC and F1-macro in percentage).

Models	Variant	Reddit		Instagram	
		F1-macro	AUC	F1-macro	AUC
GCN	baseline	$45.46_{0.01}$	$50.32_{0.26}$	$47.88_{0.96}$	$52.61_{1.80}$
	+text	$45.84_{0.36}$	$57.82_{1.94}$	$47.29_{0.01}$	$55.74_{0.73}$
	+FLAG	$48.19_{1.02}$	$60.18_{0.79}$	$48.05_{0.69}$	$56.31_{0.83}$
	+FLAG*	$48.72_{1.59}$	$60.88_{0.68}$	$49.79_{1.08}$	$55.45_{1.21}$
GAT	baseline	$46.66_{0.96}$	$52.66_{2.25}$	$49.21_{1.38}$	$51.53_{1.12}$
	+text	$48.26_{2.23}$	$59.32_{0.29}$	$48.31_{0.98}$	$54.69_{0.76}$
	+FLAG	$49.70_{1.76}$	$60.61_{1.20}$	$49.13_{1.23}$	$54.97_{0.75}$
	+FLAG*	$50.20_{2.33}$	$60.57_{1.03}$	$\mathbf{50.65}_{1.61}$	$55.98_{1.67}$
GeniePath	baseline	$45.46_{0.01}$	$52.18_{1.48}$	$47.31_{0.05}$	$51.22_{3.16}$
	+text	$46.84_{1.89}$	$56.91_{1.85}$	$47.29_{0.01}$	$52.45_{2.31}$
	+FLAG	$48.30_{2.24}$	$59.43_{0.55}$	$48.41_{1.51}$	$55.59_{0.85}$
	+FLAG*	$48.69_{2.97}$	$59.74_{1.68}$	$48.28_{1.25}$	$56.24_{2.10}$
CARE-GNN	baseline	$45.46_{0.01}$	$51.35_{1.62}$	$47.29_{0.01}$	$52.07_{1.95}$
	+text	$47.66_{1.59}$	$56.72_{1.38}$	$48.05_{1.07}$	$54.92_{0.38}$
	+FLAG	$50.78_{0.95}$	$58.43_{0.65}$	$49.64_{1.74}$	$55.79_{0.58}$
	+FLAG*	$\mathbf{51.95}_{2.21}$	$58.74_{1.40}$	$50.24_{0.46}$	$56.40_{1.29}$
BWGNN	baseline	$45.47_{0.01}$	$53.82_{2.49}$	$47.28_{0.01}$	$51.52_{2.43}$
	+text	$48.76_{1.54}$	$57.56_{1.53}$	$47.61_{0.72}$	$54.10_{0.71}$
	+FLAG	$50.93_{2.16}$	$58.89_{2.50}$	$48.55_{0.32}$	$56.33_{0.72}$
	+FLAG*	$51.91_{2.38}$	$59.20_{1.09}$	$49.35_{1.45}$	$57.19_{0.28}$
DGA-GNN	baseline	$45.46_{0.01}$	$50.10_{0.53}$	$47.29_{0.01}$	$50.86_{0.48}$
	+text	$45.49_{0.11}$	$59.59_{1.60}$	$47.29_{0.01}$	$56.28_{1.07}$
	+FLAG	$48.77_{2.18}$	$61.05_{0.71}$	$48.53_{0.42}$	$56.73_{0.66}$
	+FLAG*	$49.50_{0.83}$	$61.61_{0.78}$	$48.95_{0.68}$	$57.20_{1.03}$
PMP	baseline	$46.31_{1.04}$	$50.16_{0.12}$	$47.50_{1.56}$	$50.63_{0.52}$
	+text	$47.21_{1.18}$	$59.79_{0.43}$	$48.29_{0.01}$	$56.05_{1.27}$
	+FLAG	$48.91_{1.30}$	$61.32_{0.66}$	$48.48_{0.82}$	$57.10_{0.62}$
	+FLAG*	$50.14_{1.48}$	$\mathbf{61.80}_{0.99}$	$49.76_{1.64}$	$\mathbf{57.67}_{1.09}$

- In the zero-shot setting, FLAG continues to show substantial performance improvements over GNNs that rely solely on raw text embeddings. Specifically, on the industrial dataset, FLAG enhances its backbone with improvements of 0.95% in KS, 1.44% in F1-macro, and 0.52% in AUC. On the public dataset, FLAG achieves average gains of 1.58% in F1-macro and 1.48% in AUC. This improvement is attributed to the zero-shot generalization capabilities of large language models (LLMs) across tasks and

Table 5: Ablation studies of FLAG with different backbones on Reddit and Instagram (AUC and F1-macro in percentage).

Backbone	Variant			Reddit		Instagram	
	SS	LLM	SG	F1-macro	AUC	F1-macro	AUC
GCN			✓	$47.92_{2.77}$	$59.60_{1.56}$	$47.84_{0.79}$	$55.52_{0.46}$
	✓		✓	$48.11_{0.66}$	$60.07_{0.64}$	$47.97_{0.53}$	$55.85_{0.38}$
	✓	✓		$48.19_{0.51}$	$59.81_{2.11}$	$47.99_{0.31}$	$56.03_{0.65}$
GAT			✓	$49.03_{2.69}$	$59.96_{0.41}$	$48.76_{1.83}$	$54.20_{1.64}$
	✓		✓	$49.16_{0.63}$	$59.80_{1.64}$	$48.79_{0.98}$	$54.79_{0.25}$
	✓	✓		$49.73_{1.10}$	$\mathbf{60.27}_{1.02}$	$\mathbf{49.28}_{2.02}$	$54.27_{1.02}$
GeniePath			✓	$47.47_{2.14}$	$58.35_{0.26}$	$47.34_{0.79}$	$54.38_{0.61}$
	✓		✓	$48.38_{2.52}$	$58.31_{0.36}$	$47.54_{0.48}$	$55.24_{1.94}$
	✓	✓		$48.13_{1.74}$	$59.24_{2.99}$	$47.88_{0.96}$	$55.61_{1.80}$
CAREGNN			✓	$50.19_{1.10}$	$57.39_{0.64}$	$48.47_{1.04}$	$55.45_{0.64}$
	✓		✓	$50.48_{2.29}$	$57.94_{0.60}$	$48.70_{0.42}$	$55.47_{0.65}$
	✓	✓		$\mathbf{50.50}_{1.30}$	$57.90_{0.54}$	$48.50_{1.48}$	$55.83_{0.48}$
BWGNN			✓	$49.23_{2.05}$	$58.46_{2.36}$	$48.28_{0.24}$	$55.54_{0.25}$
	✓		✓	$50.19_{2.00}$	$58.76_{1.82}$	$47.64_{0.79}$	$\mathbf{56.86}_{0.41}$
	✓	✓		$50.19_{2.89}$	$58.63_{1.76}$	$48.29_{0.81}$	$56.27_{2.15}$

contexts, making FLAG particularly valuable in scenarios where labeled data is limited or unavailable.

- Compared with FLAG, the fine-tuned FLAG* yields an average increase of 0.84% in F1-macro and 0.42% in AUC. This suggests that the fine-tuning process effectively strengthens the relationship between the extracted text and the node labels, thereby enhancing the robustness and distinguishability of nodes.

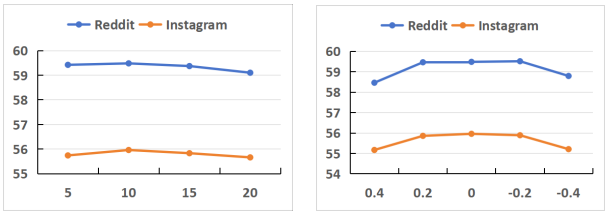
4.3 Ablation Studies (Q2)

In this section, we conduct ablation experiments to assess the effectiveness of each component in FLAG. The variants tested include SS (semantic similarity sampling), LLM (using LLM for extracting discriminative text), and SG (using skip-GNN). For the sake of fairness, experiments without LLM use the raw text encoding as node features, and the LLM used in these experiments is not fine-tuned.

The results, summarized in Table 5, demonstrate that each component in FLAG contributes effectively to the overall performance. First, when comparing the performance of the variant using both SS and SG (without LLM) with the zero-shot results, we observe that the LLM module improves the F1-macro and AUC by 1.01% and 0.31%, respectively. Second, when comparing the variant using SS and LLM (without SG) to the zero-shot performance, we find that the SG module leads to improvements in F1-macro and AUC by 0.33% and 0.74%, respectively. Finally, when comparing the performance of the variant using only SG with the combination of SS and SG, we observe that SS further enhances the performance, increasing the F1-macro and AUC by 0.15% and 0.66%, respectively. In summary, the ablation studies confirm that each component of FLAG plays a crucial role in enhancing performance.

4.4 Visualization (Q3)

In this section, we visualize how FLAG enhances the discriminability of node embeddings compared to baseline GNN models on the Reddit dataset. Specifically, we obtain 32-dimensional node embeddings from each method and use t-SNE to project these embeddings into 2D space. Given the 1:10 class ratio in the test set, for better visualization, we select all minority class nodes and randomly sample 1/3 of the majority class nodes. Notably, the LLM used in this experiment was not fine-tuned.



(a) N (b) δ

Figure 7: Sensitive analysis of hyper-parameters.

As shown in Figure 6, the node embeddings generated by the basseline GNN models are highly mixed. However, when FLAG is applied, the distinction between the two classes becomes much clearer, especially for GCN, GAT, CARE-GNN, BWGNN, and PMP. This demonstrates that FLAG successfully enhances the discriminability of node embeddings, validating the effectiveness of FLAG.

4.5 Hyper-parameter Sensitivity (Q4)

In this section, we detail the rationality of the parameter settings (i.e., selecting the top 10 nodes with the threshold to be 0) illustrated above. We use GCN as the backbone and assess the impact on GNN†'s AUC performance by varying one hyper-parameter while keeping others fixed.

Figure 7(a) shows the performance variation of GCN† with different values of top-N ({5, 10, 15, 20}). It is observed that the AUC of GCN† is slightly lower when $N = 5$ compared to $N = 10$, and it decreases when N is set to 15 and 20. This is because a larger N leads to the sampled subgraph including more normal nodes, which diminishes the focus on fraudulent nodes. When N is reduced, this issue is mitigated, but setting N too small may result in insufficient neighbor information, causing performance degradation.

Figure 7(b) shows the impact of varying the similarity threshold δ across values of {0.4, 0.2, 0, -0.2, -0.4}. We observe that when $\delta = 0.2$ or δ=-0.2, the performance remains similar to that of $\delta = 0$. However, at $\delta = 0.4$ and $\delta = -0.4$, performance significantly declines. This is consistent with the previous conclusion: a threshold that is too high reduces the central node's neighbor information, while a threshold that is too low increases heterophily in the fraudulent subgraph, leading to performance loss.

5 Conclusion

In this paper, we propose a novel framework FLAG that integrates LLMs with graph-based methods for fraud detection. FLAG addresses two major challenges: the neighborhood camouflage problem inherent in fraud detection graphs and the input size constraints of LLMs. By leveraging semantic similarity neighbor sampling strategy and LLM-based node enhancement module, we significantly improve the performance of various GNNs in fraud detection tasks. While this work focuses on static graphs to tackle financial fraud detection, there is significant potential in exploring dynamic graphs for more adaptive and real-time detection. In future work, we will investigate the use of dynamic graphs to enable real-time fraud detection, capturing evolving relationships over time.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Khaled Gubran Al-Hashedi and Pritheega Magalingam. 2021. Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019. *Computer Science Review* 40 (2021), 100402.

[3] Petros Boulieris, John Pavlopoulos, Alexandros Xenos, and Vasilis Vassalos. 2024. Fraud detection with natural language processing. *Machine Learning* 113, 8 (2024), 5087–5108.

[4] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* 15, 3 (2024), 1–45.

[5] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter* 25, 2 (2024), 42–61.

[6] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2023. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668* (2023).

[7] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*. 233–240.

[8] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 315–324.

[9] Mingjiang Duan, Tongya Zheng, Yang Gao, Gang Wang, Zunlei Feng, and Xinyu Wang. 2024. DGA-GNN: Dynamic Grouping Aggregation GNN for Fraud Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 11820–11828.

[10] Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. 2024. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 747–758.

[11] Yuan Gao, Xiang Wang, Xiangnan He, Zhenguang Liu, Huamin Feng, and Yongdong Zhang. 2023. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference 2023*. 1528–1538.

[12] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[13] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066* (2023).

[14] Zirui Guo, Lianghao Xia, Yanhua Yu, Yuling Wang, Zixuan Yang, Wei Wei, Liang Pang, Tat-Seng Chua, and Chao Huang. 2024. Graphedit: Large language models for graph structure learning. *arXiv preprint arXiv:2402.15183* (2024).

[15] Rajan Gupta and Nasib Singh Gill. 2012. Financial statement fraud detection using text mining. *International Journal of Advanced Computer Science and Applications* 3, 12 (2012).

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[17] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523* (2023).

[18] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[19] Shaio Yan Huang, Chi-Chen Lin, An-An Chiu, and David C Yen. 2017. Fraud detection using fraud triangle risk factors. *Information Systems Frontiers* 19 (2017), 1343–1356.

[20] Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024. Can GNN be Good Adapter for LLMs?. In *Proceedings of the ACM on Web Conference 2024*. 893–904.

[21] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[22] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[23] Sundar Krishnan, Narasimha Shashidhar, Cihan Varol, and AR Islam. 2022. A novel text mining approach to securities and financial fraud detection of case suspects. *International Journal of Artificial Intelligence and Expert Systems* 10, 3 (2022).

[24] Ranran Li, Zhaowei Liu, Yuanqing Ma, Dong Yang, and Shuaijie Sun. 2022. Internet financial fraud detection based on graph learning. *IEEE Transactions on Computational Social Systems* 10, 3 (2022), 1394–1401.

[25] Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Victor Wai Kin Chan, and Jia Li. 2024. Glbench: A comprehensive benchmark for graph with large language models. *arXiv preprint arXiv:2407.07457* (2024).

[26] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2023. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149* (2023).

[27] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*. 3168–3177.

[28] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4424–4431.

[29] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1569–1572.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013).

[31] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024).

[32] Soroor Motie and Bijan Raahemi. 2024. Financial fraud detection using graph neural networks: A systematic review. *Expert Systems with Applications* 240 (2024), 122156.

[33] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435* (2023).

[34] N Reimers. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084* (2019).

[35] Fengzhao Shi, Yanan Cao, Yanmin Shang, Yuchen Zhou, Chuan Zhou, and Jia Wu. 2022. H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. In *Proceedings of the ACM web conference 2022*. 1486–1494.

[36] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. 2022. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*. PMLR, 21076–21089.

[37] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295* (2024).

[38] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine* 29, 8 (2023), 1930–1940.

[39] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[40] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[41] 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).

[42] Yuchen Wang, Jinghui Zhang, Zhengjie Huang, Weibin Li, Shikun Feng, Ziheng Ma, Yu Sun, Dianhai Yu, Fang Dong, Jiahui Jin, et al. 2023. Label information enhanced fraud detection against low homophily in graphs. In *Proceedings of the ACM Web Conference 2023*. 406–416.

[43] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 806–815.

[44] Han Xie, Da Zheng, Jun Ma, Houyu Zhang, Vassilis N Ioannidis, Xiang Song, Qing Ping, Sheng Wang, Carl Yang, Yi Xu, et al. 2023. Graph-aware language model pre-training on a large graph corpus can help multiple graph applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5270–5281.

[45] Ajit Kr Singh Yadav and Marpe Sora. 2021. Fraud detection in financial statements using text mining methods: A review. In *IOP conference series: Materials science and engineering*, Vol. 1020. IOP Publishing, 012012.

[46] Bo Yan, Cheng Yang, Chuan Shi, Jiawei Liu, and Xiaochen Wang. 2023. Abnormal event detection via hypergraph contrastive learning. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 712–720.

[47] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*. 1955–1973.
[48] Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, Jian Yang, and Jianliang Gao. 2022. efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Transactions on Information Systems (TOIS)* 40, 3 (2022), 1–29.
[49] Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. 2024. GraphTranslator: Aligning Graph Model to Large Language Model for Open-ended Tasks. In *Proceedings of the ACM on Web Conference 2024*. 1003–1014.
[50] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709* (2022).
[51] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems* 33 (2020), 7793–7804.
[52] Wei Zhuo, Zemin Liu, Bryan Hooi, Bingsheng He, Guang Tan, Rizal Fathony, and Jia Chen. 2024. Partitioning message passing for graph fraud detection. In *The Twelfth International Conference on Learning Representations*.

A Pseudo Code

Algorithm 1 Fraud Detection with LLM-enhanced Graph Neural Networks (FLAG).

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, Node Text $\mathcal{T}$, Pre-trained LLM, Frozen LM

Sampling Phase:
for each node $v \in \mathcal{V}$ **do**
  for each neighbor $u \in \mathcal{N}(v)$ **do**
    Calculate semantic similarity $sim(v, u)$ using Eq. 3
  end for
  Select top-k neighbors $\mathcal{N}_{selected}(v)$ using Eq. 4
  Construct subgraph $\mathcal{G}_v$ based on selected neighbors
end for
Training Phase:
for each node $v \in \mathcal{V}$ **do**
  Extract discriminative text t_v^D and residual text t_v^R using fine-tuned LLM
  Encode both discriminative and residual texts using frozen LM
  Feed the encoded features into shared skip-GNN
  Compute the total loss $\mathcal{L} = \mathcal{L}_{Disc.} + \mathcal{L}_{Res.} + \mathcal{L}_{Orthog.}$
  Update LLM and skip-GNN parameters alternately
end for
Inference Phase:
Reset skip-GNN parameters
for each test node $v \in \mathcal{V}$ **do**
  Extract discriminative text t_v^D
  Encode both raw text t_v and discriminative text t_v^D using pre-trained LM
  Process the encoded features through skip-GNN
  Apply attention mechanism to fuse outputs: $z_v = \text{Attention}(z_v^D, z_v)$
  Update skip-GNN using BCE loss and backpropagation
end for
return Node predictions z_v

B Related Work

B.1 Large Language Model for Graph

LLMs have demonstrated exceptional language understanding and summarization capabilities in natural language processing, showcasing their strong generalization abilities across unseen tasks [31, 33]. Their emergence has sparked significant interest in the graph learning community, prompting research into how LLMs can enhance performance on graph-related tasks [6, 20]. Most existing approaches that combine LLMs with GNNs can be broadly classified into three categories: LLMs-as-Predictors, LLMs-as-Enhancers, and LLM-GNN Alignment. LLMs-as-Predictors leverage LLMs to directly predict node labels or graph properties. Methods like NLGraph [41] and GPT4Graph [13] tackle graph tasks by describing graph structures through natural language, enabling the LLMs to infer patterns and make predictions. InstructGLM [47], on the other hand, employs instruction tuning to adapt the LLM to graph downstream tasks, using natural language instructions and graph-text alignment to express graph structural information effectively. LLMs-as-Enhancers utilize LLMs to enhance node representations by generating supplementary textual features. OFA [26] unifies graph data from diverse domains into a common embedding space for cross-domain learning. TAPE [17] generates custom prompts to query LLMs, producing both predictions and textual explanations for nodes, which are then converted into GNN node embeddings. Similarly, LLMRec [43] uses LLM-generated profiles to filter out noisy interactions and enrich node embeddings, thereby significantly improving recommendation performance. LLM-GNN Alignment focuses on aligning the capabilities of LLMs and GNNs to improve graph-based task performance. GLEM [50] iteratively utilizes the LLM and GNN to provide mutual labels for node classification, thereby aligning their respective strengths. GraphTranslator [49] uses a translator module to align graph models with LLMs, facilitating seamless integration for improved performance on graph tasks. However, these methods generally faces a significant challenge: the neighbor camouflage problem, which hinders the effectiveness of these methods in fraud detection tasks.

B.2 Graph-Based Fraud Detection

Graph-based fraud detection is essential for identifying fraudulent activities in financial and online networks [29, 46, 48], where increasing complexity makes it both more critical and challenging. The primary challenges in this domain is neighborhood camouflage. Fraudulent nodes often connect to numerous normal nodes, and when GNNs aggregate information from these heterophilous neighborhoods, the distinctiveness of fraudulent nodes becomes diluted. This makes it increasingly difficult to distinguish anomalous nodes from normal ones, thus complicating fraud detection. To address neighbor camouflage, existing approaches are primarily categorized into spatial and spectral methods. Spatial methods focus on mitigating the influence of noisy neighbors through strategies like re-sampling, re-weighting, or neighbor selection. CARE-GNN [8] is an early one to address neighbor camouflage explicitly, proposing similarity-aware neighbor selection. PC-GNN [27] extents this by balancing subgraph labels through sampling and refining neighborhood aggregation. DGA-GNN [9] introduces a feedback dynamic grouping strategy that classifies nodes into two distinct groups

and employs hierarchical aggregation to extract more discriminative features for fraud detection tasks. PMP [52] mitigates the effects of class imbalance by assigning different weight matrices to neighbors of different labels and unlabeled neighbors. In contrast, spectral methods tackle the problem using graph spectral properties. For instance, BWGNN [36] identifies that anomalous nodes often concentrate their features in the high-frequency domain and designs mechanisms accordingly. Similarly, GHRN [11] establishes a connection between heterophily and spectral properties, pruning heterophilic edges to improve classification. While effective, these methods fail to fully utilize the contextual and semantic richness inherent in textual features, which are critical in real-world financial datasets.

C Examples of Discriminative and Residual Text

Below are examples of discriminative and residual texts extracted from the original text of a popular user in the Reddit dataset. Clearly, the discriminative text captures the unique identity of the nodes.

Ori. Text: I've been experimenting with a new strategy in Civilization VI, focusing on culture early to gain more diplomatic influence. It's a bit of a gamble, but so far it's paying off. Anyone else tried this approach?; I'm currently streaming my gameplay of Age of Empires on Twitch. I've been getting a lot of viewers lately—thanks to everyone who's been tuning in! It's awesome to discuss strategies with so many people in real time.; Just came back from a hiking trip to the Rocky Mountains. The weather was perfect for a hike, and I got some amazing shots of the landscape. Here's one of my favorite photos from the trip!

Disc. Text: This user frequently shares unique game strategies for Civilization VI and streams Age of Empires on Twitch, where they engage with a large community of viewers.

Res. Text: This user shares personal experiences, such as their hiking trip to the Rocky Mountains and the landscape photos they took.