國立臺灣大學電機資訊學院暨中央研究院

資料科學學位學程

碩士論文

Data Science Degree Program

College of Electrical Engineering and Computer Science

National Taiwan University and Academia Sinica

Master Thesis

以帶潛在標籤的關係圖神經網絡改進垃圾評論之檢測

Improving Detection of Spam Reviews via Relational Graph Neural Networks with Potential Labels

洪贊濱

Tsan-Pin Hung

指導教授：謝宏昀 博士、王志宇 博士

Advisor: Hung-Yun Hsieh, Ph.D., Chih-Yu Wang, Ph.D.

中華民國 112 年 8 月

August, 2023

doi:10.6342/NTU202304108

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

以帶潛在標籤的關係圖神經網絡改進垃圾評論之檢測

Improving Detection of Spam Reviews via Relational Graph Neural Networks with Potential Labels

本論文係 洪贊濱（R09946019）在國立臺灣大學資料科學學位學程完成之碩士學位論文，於民國 112 年 7 月 27 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Data Science Degree Program on 27 July 2023 have examined a Master's thesis entitled above presented by TSAN-PIN HUNG (R09946019) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

謝宏昀	王志宇	
（指導教授 Advisor）	（指導教授 Advisor）	
黃瀚萱	蔡銘峰	

學程主任 Director: 謝宏昀

致謝

日月如梭，轉眼間就過了三年，從剛進實驗室不懂的如何做研究的我到如今也慢慢知道如何用自己的所去看待研究並在巨人的肩膀上改進現有的研究來完成自己得論文，一路走來雖然跌跌撞撞，多虧有大家的幫助最後終於能運用自己所學順利的完成論文。

首先我要特別感謝謝宏昀教授的指導指導和包容，感謝教授在一次次的討論中指點我該如何看待以及深入研究課題，教授總能在我有盲點的時候點出來讓我知道該如何用不同的角度去思考，讓我能夠從剛入學完全不懂如何閱讀論文和推進研究到現在能夠產生自己的觀點並在閱讀一篇篇論文後深入研究主題，非常感謝教授不厭其煩的指導，成就如今的我。

接下來我也要特別感謝我的共同指導老師王志宇研究員，感謝王老師在每個月的討論中點出我沒注意到的地方，並提供了他的觀點來幫助我把研究做得更好，也感謝王老師提供了強大的運算資源讓我能夠順利的跑出原本以為無法跑出來的結果。

我要感謝大寬在我們三上和一起跟老師討論的過程中互相打氣，也感謝健達不管是修課和研究上都幫助我許多，感謝這兩位同學讓我對論文的產出和口試沒那麼徬徨，也感謝政燁學長在計中工讀時的協助，讓我增進了報告的能力，也感謝學長在口試前最後的提點，讓我最終的實驗能更豐富。

接下來感謝碩三時一起待在實驗的學弟妹，在我面臨研究和畢業壓力時不會感到孤獨，在面對低潮時能夠更快的振作，也感謝宇翔在我壓力大的一直聽我抱怨幫我加油打氣，也感謝昶凱在最後半年對我的幫助，你對我的提點讓我獲益良多。

最後我也要感謝在口試前幫我順口試投影片和內容的昶凱、定為、奕寶，在口試的前一晚協助我修改口試投影片，也給了我很多建議，讓我能順利完成口試。

時光飛逝，三年一下就過了，如今也要畢業了，感謝在研究所遇到的各位給我的種種協助，在我遇到瓶頸時拉我一把幫助我度過難關，讓我能夠堅持下去，也感謝各位對我的提點，幫助我看到自己的不足之處，即使我也很多不足的部分仍在旁協助我，幫助我面對種種的難關。

2023/8/10 洪贊濱筆

doi:10.6342/NTU202304108

摘要

在垃圾評論檢測領域，基於圖的檢測法由於能捕捉評論間的互動關係而受到廣泛矚目。然而圖神經網路（GNN）反覆聚合鄰點訊息的特導致過平滑的問題，使得良性與惡性評論的節點表示有可能趨同。雖然早前有研究試圖透過同時考慮同質和異質連接來降低影響，嘗試反向聚合異質連接，但由於依然使用相同的聚合函數同時聚合不同標籤的鄰點，且假設所有良惡評論節點表示應各自相近，導致未能有效避免過度平滑。此外，一次性更新所有節點的表示在資料量增長時將導致記憶體需求過大，因此使用子圖聚合在實際應用中變得必不可少。然而過去的方法在建構子圖時，並未考慮到圖的拓撲結構來進行鄰點採樣，因此無法有效補捉緊密交互的鄰點之訊息。為了解決上述的問題我們提出了一種基於潛在關係的圖神經網路垃圾評論模型，該模型根據圖的拓撲結構相似性對進行採樣產生子圖進行隨機訓練，在聚合鄰點訊息前先使用分類器分類出潛在良性與惡性評論鄰點，接著使用分層的聚合策略，將潛在良性與惡性評論視為兩種不同的關係分開進行聚合後，再組合這兩類評論鄰點的訊息進行下一層的聚合。同時，我們設計了一種新的三元損失函數，使良性評論的表示與評論對象的表示之間的相似度高於與惡性評論節點的相似度，來降低過度平滑的影響，更符合現實中的觀察。我們的實驗結果證明了我們方法的有效性，在 yelpNYC 資料集中使用隨機切分的情況我們的方法在 AUC 分數的表現上平均高於主要參考模型 6%和次要參考模型 1.5%，達到了 0.84，而在按時間序切分的情況下，我們的 AUC 分數上平均分別高於主要以及次要參考模型 5.5%以及次要參考模型 6.5%，在其他資料及上也都得到優於參考模型的節結果，並且在每一次的實驗結果中的 AUC 的分數都優於其他兩者。

doi:10.6342/NTU202304108

ABSTRACT

Graph-based spam review detection has been appealing due to its ability to capture review interactions. However, it has problems with over-smoothing because the recurrent aggregation of neighborhood data makes it difficult to distinguish between benign and spam reviews. Although existing studies consider homogeneous and heterogeneous connections, but employ the same aggregation function and presume that benign and spam review representations should be similar, which results in inefficiencies. Additionally, updating all node representations at once becomes unfeasible as data quantities increase due to memory constraints, necessitating subgraph aggregation. However, prior approaches did not consider the topological structure of the graph in subgraph construction, making it difficult to capture information from closely interacting neighbors effectively. To address these issues, we present a GNN model for spam review detection based on potential labels to overcome these problems. According to the topology of the graph, our model sample subgraphs use a hierarchical aggregation strategy and treat potential labels of benign and spam reviews as two different relationships. We also designed a novel triplet loss function that ensures the similarity between the representation of benign review and the target of review is higher than that with spam review nodes, mitigating over-smoothing. Our experimental results demonstrate the effectiveness of our method. In the YelpNYC dataset, under random splitting, our approach outperformed the primary and secondary baseline models by 6% and 1.5% respectively on average AUC scores, achieving a score of 0.84; in the case of chronological splitting, our AUC scores were on average 5.5% and 6.5% higher than the primary and secondary baseline models respectively, achieving a score of 0.68. Our method also achieved superior results on other datasets and consistently exceeded the AUC scores.

doi:10.6342/NTU202304108

TABLE OF CONTENTS

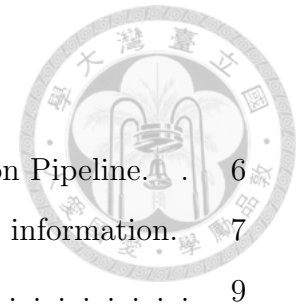
doi:10.6342/NTU202304108

doi:10.6342/NTU202304108

LIST OF TABLES

doi:10.6342/NTU202304108

LIST OF FIGURES



doi:10.6342/NTU202304108

doi:10.6342/NTU202304108

CHAPTER 1

INTRODUCTION

In modern society, online reviews play an important role in daily life; most people would read online reviews before making a purchase decision. In 2022, 98% of consumers read online reviews for local businesses, and only 21% of consumers don't trust online reviews as much as experts. [2] In other words, these user-generated reviews are crucial for preserving confidence in the online ecosystem as they significantly influence how consumers make purchasing decisions.

Due to the lack of rigorous controls, spam reviews can be generated and uploaded on e-commerce websites with relative simplicity, which is specifically the reason for an increase in spam activities. Companies may hire people, referred to as spammers, to create fraudulent evaluations of their goods or services. [3] These spam reviews are frequently written to increase sales or the visibility and appeal of the company's products. Review spamming is the word used to refer to this behavior.

Unfortunately, the growth of online e-commerce platforms has been followed by a rise in fraudulent strategies such as spam activities. This increase is mainly due to loose rules and controls, which make it relatively easy for spam reviews to be created and posted on e-commerce platforms. Companies may hire people—commonly known as spammers—to manufacture evaluations for their goods or services, [3]. Increasing the visibility of products, boosting sales, or otherwise changing the public opinion of a business's products or services are frequently the targets of these spam reviews. '*Review spamming*' is the typical term for this fraudulent behavior.

Researchers are actively working to stop the growing threat of fraud in the modern digital environment, particularly in the form of spam reviews [4]. Due to the rapid increase in this fraudulent activity, quick and efficient prevention efforts are required. As a result, both the academic and business communities have given the problem of picking up and removing spam reviews a lot of attention. This change has led to increasing attention on research aimed at understanding the structure of spam reviews, creating powerful detection systems, and ultimately reducing their incidence. Such carefully targeted efforts show how important and well-known this topic is becoming.

For instance, feature-centric approaches try to extract or create informative traits that help distinguish between benign and spam reviews [5]. In general,

doi:10.6342/NTU202304108

statistical learning or machine learning approaches are implemented to achieve this. Natural language processing characteristics like Linguistic Inquiry and Word Count (LIWC) and Part of Speech (POS) tagging are used in certain feature-centric techniques [6]. However, experienced spammers can frequently readily camouflage this rule-based features [7], prompting academics to examine alternative options, including graph-based techniques.

By their very nature, spammers collaborate and communicate often with other spammers in order to spread their impact. network-based approaches can make use of these behavioral patterns to distinguish between genuine and spam reviews inside a graph structure. In recent research, GNNs have been used to address this issue, often by constructing homogenous graphs using manually created rules. However, this procedure may cause information to be lost between reviewers and the items being evaluated.

Many academics have experimented with constructing homogenous graphs to detect spam reviews. CARE-GNN [7] is a notable example. It employs rule-based techniques to identify relationships between reviews and then maps the structured data into a homogenous graph. They need domain-knowledge experts to define rules for every dataset in order to create the graph. however, performing such an operation would drop the information between reviews, which would have lost the information between reviews.

Additionally, although graph-based techniques are capable of extracting spam patterns from a graph structure, they mainly work on the homophilic assumption. The graph-based model iteratively aggregates the information from neighbors to generate the final embedding. This frequently results in an over-smoothing problem and a high false negative rate when combined with the excess imbalance between spam and spam reviews. Recent research has sought to solve this problem by taking into account both homophilic and heterophilic connections inside a homogeneous network, such as H^2-FDetector [1]. However, they use a naive approach that simply adds a minus operation to aggregate the heterophilic connections, which can not efficiently aggregate the category message between spam and benign reviews.

To address the above disadvantage, we propose a novel graph-learning framework intended to reduce over-smoothing. Our system is constructed with the intention of separately aggregating spam and benign neighbors. Our approach guarantees a thorough comprehension of the unique features and attitudes of each group. As a result, it produces a representation of a graph that is more precise and balanced, improving model performance. This novel approach helps to improve the validity and reliability.

doi:10.6342/NTU202304108

We also adapt the triplet loss to add to our model to reduce the over-smoothing issue by distinguishing the embedding between different categories. The model's capacity to distinguish between several categories is improved by this inclusion, which also strengthens the reliability of the results. In our system, we also include a topology-aware neighbors sampling module. The robustness of our results is further strengthened by this module's assistance in sampling stronger tie neighbors. Combining these complex methods yields a more comprehensive and robust model for spam review detection.

The main contributions of this thesis are summarized as follows:

1. We propose a relational graph neural network with potential labels to aggregate different potential labels separately.

2. We adapt the triplet loss to distinguish the benign and spam reviews of each product.

3. We introduce the topology-aware neighbors sampling approach to sample neighbors with stronger ties.

The remaining chapters of this thesis are organized as below:

- In Chapter 2, we discuss the background knowledge of spam review and spam review detection as well as the related work that inspired us.

- In Chapter 3, we introduced the detection pipeline of our system and described each module in our system.

- In Chapter 4, we describe our proposed methods, including the neighbors sampling approach, our proposed detection model, and the optimization approach.

- In Chapter 5, we reported the performance of our proposed method and other baselines.

- In Chapter 6, we summarize our results and future work.

doi:10.6342/NTU202304108

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we briefly introduce the background of spam review and spam review detection methods and organize the related work of these methods.

2.1 Spam Review

Nowadays, people may readily express their opinions on forums, blogs, and e-commerce websites. In addition, it is widespread for people to read opinions about services and products before purchasing. Opinion spam, which occurs when biased reviewers post false feedback to either promote or disparage a product (or service) in order to mislead customers for profit or reputation, has sadly become a serious problem in online reviews. This conduct is referred described as "*review spamming*" [8]. In general, spam reviews usually can be categorized into three different forms [9]:

- **untruthful opinions:** Reviews that intentionally fraudulent readers or opinion mining systems by giving some specific objects undeserving positive feedback to promote the specific products (termed push attack [10]) or by giving some other specific products unfair or intentionally negative reviews to damage their reputations (termed nuke attack [10]).

  Collectively, these untruthful reviews are commonly referred to as fake reviews. These fake reviews have caused great damage to personalized recommendation systems and undermined consumer and company confidence in the online market.

- **reviews on brands only:** Reviews that focus on the manufacturers, sellers, or brands of the products rather than the things themselves. Considering the fact that they may be beneficial, Researchers consider them to be spam [8] because they frequently have prejudice and are not directed at any specific products.

- **non-reviews:** Reviews can be categorized into two groups: 1) advertisements and 2) pointless reviews with unrelated text (e.g., questions, answers, and random texts).

The topic of spam review detection has been the focus of numerous investigations. Jindal and Liu [9] presented the first categorization of this topic into three

doi:10.6342/NTU202304108

categories: review-centric, reviewer-centric, and product-centric. They extracted distinctive features from these categorizations and developed a logistic regression model. Other traditional techniques use statistical learning to approach the issue. These techniques carry out feature-centric spam review identification by employing supervised classifiers to extract anomaly patterns based on review-associated semantic messages [11]. Furthermore, other studies concentrate on the use of language models or utilize linguistic inquiry and word count (LIWC) or Part of speech (POS) to extract different aspects from textual reviews.

The reality of the problem is further complex, even while the method of using feature-centric or statistics-based spam review detection techniques can show to be a significant benefit in the identification and labeling of spam reviews. Although offering a complex design, such strategies are based on the recognition of patterns and indications frequently associated with spam reviews. For example, they can spot suspiciously high keyword frequency, unusual linguistic patterns, or inconsistent rating behavior. These methods essentially examine a putative spam review via the prism of statistical differences or gaps from the fraudulent set by benign reviewers.

However, spammers have created adaptive methods that regularly prevent these detection systems. They have developed the skill of controlling the specific pattern these systems depend on, gradually reducing their effectiveness. They frequently substitute alternatives or sentences with semantically similar meanings for terms that have been detected. By employing this strategy, they avoid malicious comments recognized by the spam review detection system.

Additionally, in order to hide their behavior, these spammers worked on their ability to duplicate the habits of genuine reviewers. They adopt their review submission routines, use several rating scales, and adjust their commenting routines. It becomes harder and harder for detection systems to discriminate between genuine and spam reviews as a result of such camouflage behavior. The problem comes from the fact that these spammers are essentially copying human behavior, which is always changeable and unexpected.

2.2 Graph-based Spam Review Detection

Since spammers can easily disguise themselves as genuine reviewers with some simple tricks, the focus on research in this area has shifted to more complex detection techniques like graph-based methods [12]. A rising number of academics are focusing their research on the investigation and use of graph-based approaches. These approaches are made to make use of the intricate relationships that exist between different entities, such as users, products, and reviews, in the context of

doi:10.6342/NTU202304108

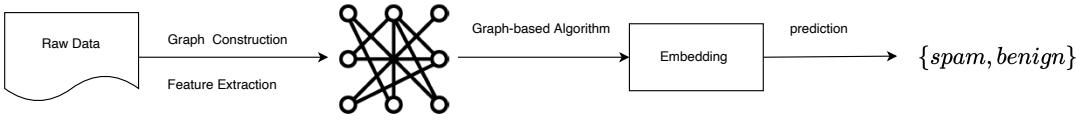


Figure 1: An illustration of Graph-Based Spam Review Detection Pipeline.

the review system [5].

The graph-based method, which has received a commendation for its use in representation learning on graphs such as social networks and knowledge graphs, has developed into a crucial tool for studying and understanding complicated relational data. Recently, researchers have been more aware of how, despite their relative effectiveness, traditional feature-based methods sometimes ignore the complex connections between reviews, reviewers, and products. Given that these relations can be essential in detecting spam reviews in some situations, in particular, this makes up a huge research interest.

The ability of the graph-based methods to represent data is well established. The key idea behind this system is to illustrate the complex relationships between reviewers, products, and reviews as a graph. Every node and edge in a homogeneous graph belongs to the same type. When different kinds of nodes (such as users, products, and reviews) and edges coexist, the network is said to be heterogeneous.

2.2.1 Graph Neural Network

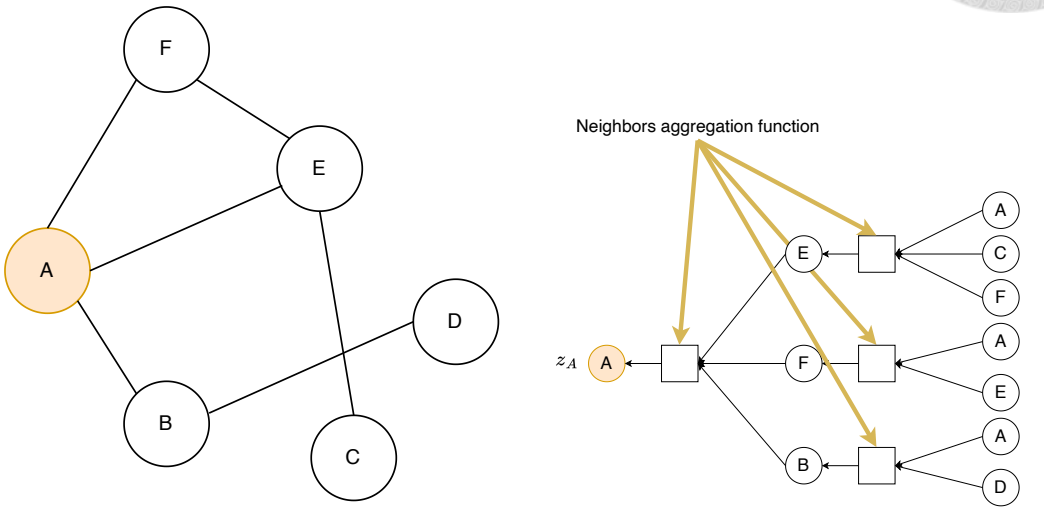
The primary advantage of Graph Neural Networks (GNN) is their power to learn and represent features based on network structures and attributes, providing a solid foundation for comprehending the entire graph in contrast to traditional machine learning, which only models the relationships based on individual data points. The interactions and larger relational contexts that are associated with graph structures are sometimes overlooked by traditional approaches. While capturing hidden patterns within the topological graph structure to provide a more comprehensive and interconnected representation of the data, GNNs succeed in scenarios where data relations are important.

Gilmer et al. [13] proposed a general Message Passing Neural Networks (MPNNs) on a graph that can take an input graph $G = (V, E)$, with a set of node features $X \in \mathbb{R}^{d \times |V|}$, where d is the dimension of node features, and use these attributes to generate node embeddings $z_u, \forall u \in V$.

In each message-passing iteration of a GNN, the hidden state of embedding h_u^k associated with each node $u \in V$ is updated based on information aggregated from $u's$ neighbors $\mathcal{N}(v)$ on the graph. Figure 2 illustrates how an MPNN generates

doi:10.6342/NTU202304108

embedding of a single node A. The model aggregates messages from A's neighbors (i.e., B, E, and F) in the Figure 2(a), the messages of these neighbors are based on messages aggregated from their own neighbors, and so on. Figure 2(b) shows the computation graph of how a two layers message passing model generates the embedding of a single node A.



(a) Input graph

(b) Computation graph for updating the embedding of node A

Figure 2: Illustration of how a single node aggregates neighbor's information.

The hidden state of a node u's embedding at l-th in this message-passing update pipeline is formulated as follows:

$$h_u^{(l)} = \text{UPDATE}^{(l)}\left(h_u^{(l-1)},\ \text{AGGREGATE}^{(k-1)}\left(\{h_v^{(l-1)},\ \forall v \in \mathcal{N}(u)\}\right)\right) \tag{2.1}$$

$$= \text{UPDATE}^{(l)}\left(h_u^{(l-1)}, m_{\mathcal{N}(u)}^{(l-1)}\right), \tag{2.2}$$

in this context, UPDATE and AGGREGATE stand as any distinct differentiable functions, with $m_{\mathcal{N}(u)}$ being the "message" accumulated from $\mathcal{N}(u)$, the set of neighbors of node u.

During each iteration l within the GNN, the model leverages the AGGREGATE function to process the set of node embeddings in the neighboring nodes $\mathcal{N}(u)$, creating a message $m_{\mathcal{N}(u)}^{(l)}$ based on this collected neighbor information. Following this, the model employs the UPDATE function to merge the message $m_{\mathcal{N}(u)}^{(l)}$ with the embedding $h_u^{(k-1)}$ of node u obtained from the preceding iteration. The node embeddings at the starting point of iteration, $l = 0$, are set as node features, which is to say, $h_u^{(0)} = x_u,\ \forall u \in V$. After undergoing L times of GNN message-passing, the output from the final iteration is used to establish the embeddings for each individual node. Consequently, the embeddings for each node

doi:10.6342/NTU202304108

are defined as:

$$z_u = h_u^{(L)},\ \forall u \in V. \tag{2.3}$$

The most basic GNN message-passing that described in the Equation 2.1 is defined as [14]:

$$h_u^{(l)} = \sigma\left(W_{\text{self}}^{(l)} h_u^{(l-1)} + W_{\text{neighbors}}^{(l-1)} \sum_{v \in \mathcal{N}(u)} h_v^{(l-1)} + b^{(l)}\right), \tag{2.4}$$

where $W_{\text{self}}^{(l)},\ W_{\text{neighbors}}^{(l-1)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ are trainable weight matrices, $d^{(l)}$ denotes the degree of hidden state nodes embedding in the l-th iteration, and σ denotes a non-linear activation function (e.g., ReLU, Sigmoid, ...).

Define the x_v as the feature for node v, where $x_v \in \mathbb{R}^{d_1}$, and the feature w_e as the feature for edge (u, v), where $w_e \in \mathbb{R}^{d_2}$.

The paradigm of message-passing operations of the node-wise and edge-wise at layer $l+1$ can define as follows: [15]:

$$\text{Edge-wise: } m_e^{(l+1)} = \phi(x_e^{(l)}, x_v^{(l)}, x_u^{(l)}),\ (e, u, v) \in \mathcal{E}, \tag{2.5}$$

$$\text{Node-wise: } x_u^{(l+1)} = \psi\left(x_u^{(l)}, \rho\left(\{m_e^{(l+1)} : (e, u, v) \in \mathcal{E}\}\right)\right), \tag{2.6}$$

where ϕ is an edge-wise message function assigned to each edge that combines the features of the edge with those of its incident nodes to generate a message. Additionally, ψ is an update function for each node responsible for updating the feature of nodes. And ρ represented the reduction function used to aggregate incoming messages during this updating process. In the above equations, ϕ is a message function defined on each edge to generate a message by

2.2.2 Stochastic Training on Graphs

Due to computational and memory limitations, training GNNs on large graphs, especially ones with millions or even billions of nodes or edges, offers tough challenges. For such big graphs, the full-graph training method that updates all of $m_e^{(l+1)}$ and $x_v^{(l+1)}$ simultaneously is frequently impossible. Consider performing a GNNs convolution on a Graph with N nodes that have a hidden state size of H with L layers as an example; for large values of N, maintaining all of the hidden states requires $O(NLH)$ memory, which would rapidly overtake a single GPU's capabilities.

However, performing a mini-batch of stochastic training on a graph is not naive. The linkage of graphs is one of the major roadblocks. Nodes in a graph

doi:10.6342/NTU202304108

have relations, in contrast to traditional machine learning tasks where data points are typically independent of one another and can be sampled independently. It suggests that a node's representation is based on its own attributes as well as those of its neighbors and perhaps even nodes further away in the graph. When processing a small subset of nodes or edges at once in a minibatch, this dependence on nearby nodes becomes an issue. The model will be unable to produce an effective representation for a node if it depends on the features of plenty of other nodes that are not present in the current minibatch.

Furthermore, simply sampling N edges from the edges set E, where $N << E$, can result in the sampling of isolated edges or disconnected subgraphs. This can be troublesome since it does not give an exhaustive view of the structure of the entire graph, which is necessary for efficiently training Graph Neural Networks (GNNs). This setting produces isolated edges and disconnected subgraphs, making them less useful to the learning algorithm. Additionally, the connectedness of the manufactured graph, which is necessary for the propagation of messages in GNNs, might be harmed if the sampled edges are isolated.

In order to accomplish such stochastic training on graphs, GraphSAGE [13] samples a batch of nodes as well as a fixed size of neighbors for each node. Unlike passing messages on the full graph that need to be loaded the entire graph into memory, GraphSAGE [13] randomly samples a small, fixed number of neighbors for each node, which helps to keep the size of the computation graph small and manageable. Importantly, these neighborhoods are sampled individually for each node in the minibatch, meaning the computation graph is different for each mini-batch. By iteratively aggregating and transforming information from a node's local neighborhood, we are able to generate node embeddings that capture both the local graph structure and features of nodes.

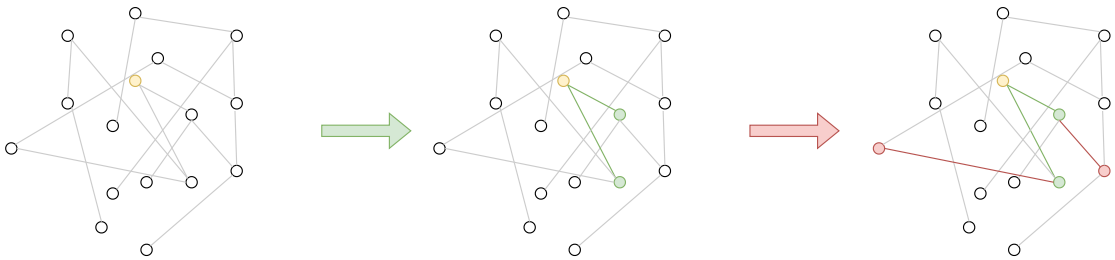


Figure 3: Overview of neighbors Sampling methods.

Figure 3 illustrates the neighbor sampling method. Assume we are employing a 2-layer GNN model. We first sample the 1-hop neighbors (shown by green nodes) and then the 2-hop neighbors (represented by pink nodes) of each center node in the batch (represented by an orange node). We first use the message from the pink nodes to update the embedding of green nodes before determining the embedding

doi:10.6342/NTU202304108

of orange nodes. The final embedding for the orange nodes can be calculated using the green nodes.

2.2.3 Neighbor Sampler

For the stochastic training approach, nodes aggregate messages selectively, taking into account only a chosen subset of neighbors as opposed to all possible neighbors. Thus the neighbor selection approach is a pivotal factor in graph-based spam review detection and make a significant effect on the quality of the final node and edge embeddings.

In CARE-GNN [7], they employ reinforcement learning to find the optimal threshold to perform the top-p sampling and compute the similarity with neighbors to measure the association with neighbors in terms of feature similarity to sample top-p similar neighbors. With this approach, we can only aggregate the message of nodes in which the feature is the most similar to the center nodes, and we also can not aggregate the message from neighbors, which frequently interact with the center node.

In PC-GNN [16], they separate their neighbor sampler into two parts: pick and choose. In the pick stage, they sum up the adjacency matrices of all relations and then use the two norms of the column vector of neighbors in the normalized adjacency matrix and the label frequency of neighbors to calculate the sample probability to pick the neighbors nodes. Then, they define a distance function to calculate the distance of selected neighbor nodes and operate over-sampling of neighbors in the minority class and under-sampling the neighbors in the majority class. Though their approach can capture the local importance by the adjacency matrix and the utilization of the distance function can help them sample the more related nodes in terms of features, this approach can not efficiently capture the neighbors with frequent interactions.

2.3 Related Work

2.3.1 GAS

Li et al. [17] proposed a novel anti-spam model that makes use of a Graph Convolutional Network (GCN) in the setting of a bipartite graph with node and edge attributes. This novel method was implemented on Xianyu, a well-known Chinese online marketplace, and makes use of the task of spam review detection as an edge classification task that utilizes the edge embedding together with the source and target nodes that this edge links to identify the spam review.

To perform graph convolutional networks on bipartite graphs, Li et al. [17]

doi:10.6342/NTU202304108

design three aggregating functions for the user, product, and review entities, respectively, to execute graph convolutional networks (GCN) on bipartite graphs. The aggregating function for review entities aggregates the information of the review product and review user that this edge links; aggregating function for user and product utilize the edge embedding, and the product/user embedding of the edges that linked to it with attention mechanism [18] to aggregate the neighbors' information.

After aggregating the neighbors' information, Li et al. [17] combine the node attribute of the neighbors' node with trainable weight matrices for both the user and product node to generate the final node embedding. Ultimately, Li et al. [17] use the user, product, and review embedding of the given review to identify spam reviews.

In conclusion, GAS [17], employs a heterogeneous graph representation widely applicable in real-world scenarios where data can be naturally depicted. This method can update the review embedding by aggregating user and product messages. The Graph Convolutional Network (GCN)-based solution, on the other hand, is constrained by severely unbalanced labeling and only depends on the homophily assumption. As a result, there may be a noticeably higher false negative rate.

2.3.2 H^2-FDetector [1]

H^2-FDetector [1] models both homophilic and heterophilic connections (H^2-connection) between nodes simultaneously in the fraud graph, assimilation of homophilic connection nodes and discrimination of heterophilic connections. To accomplish this goal, they first recognized the H^2-connection in a fraud graph, then aggregated the message under the influence of the H^2-connection. Ultimately, they use the category features of all known fraudsters to identify new fraudsters.

Considering the premise that nodes with the same label are similar and those with different labels are distinct, H^2-FDetector [1] designs a H^2-connection identification sub-layer to measure the difference or similarity between nodes and uses it to predict whether an edge is a homophilic connection or heterophilic connection from the nodes that have been labeled.

After extracting the edge connection types, H^2-FDetector [1] combines neighbor nodes from homophilic and heterophilic neighbors, making the representations between homophilic connections similar. In contrast, the representations between heterophilic connections become discriminative.

Then H^2-FDetector [1] finds each class's approximate category center to extract each class's category information. The model derives the node representation

doi:10.6342/NTU202304108

based on the H^2-connection aggregate strategy in the previous submodule. However, some fraudsters are stranded in neighborhoods with an excessive number of benign entities, which make them only able to aggregate message from their inter-class neighbors but unable to learn inter-class similarities from other fraudsters. To solve this issue, H^2-FDetector [1] employs a prototype extraction strategy to reduce the distance between each sample and the prototype, resulting in a closer similarity between samples within the class.

In conclusion, the H^2-FDetector model presented by Shi et al. [1] considers homophilic and heterophilic connections within a homogeneous graph. Consequently, it necessitates the construction of a homogeneous graph from raw data, which may inadvertently lead to the loss of some information during the graph-building process. Furthermore, this particular model cannot harness the information about the review target to predict the presence of spam reviews accurately.

2.4 Summary

In this chapter, we explore the topic of spam reviews, which are a big issue on online platforms in Section 2.1 and give a precise definition of spam review. Along with this definition, we also introduced some traditional detection methods. And then, we explore the graph-based spam review detection techniques in Section 2.2, which leverage the power of graph theory in order to extract the social interaction between reviews. Finally, in Section 2.3, we introduced the related study that inspired us and discussed the pros and cons of these methods.

doi:10.6342/NTU202304108

CHAPTER 3

SYSTEM MODEL

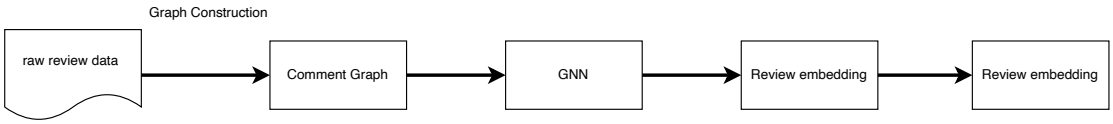


Figure 4: System model

In this chapter, we explain our spam review detection pipeline. The system constructs the comment graph from the raw review datasets. And then using the GNN model to learn the social interaction between reviews to update its embedding and utilize it to detect the spam review. The illustration of the spam review detection pipeline is shown in Figure 4.

The sections are arranged as follows: Section 3.1 introduces the review dataset we used in this work. In Section 3.2, we introduce how to construct the comment graph from the dataset, and then in Section 3.3, we introduce how the system sample subgraph further feeds into the GNN model. In Section 3.4, we introduce how to use heterogeneous graph convolutional Networks to learn the embedding of nodes and edges to identify spam reviews.

3.1 Dataset Description

To evaluate system performance, We used two open datasets from Yelp.com that included restaurant reviews in different parts of America collected by Rayana et al. [5]. Yelp.com is a well-known website for local business reviews. Users of this website can submit a review of any of the products or services listed.

Yelp has a filter system that can separate reviews into recommended and unrecommended (filtered) lists to identify fake/suspicious content. We classify them as benign and spam, respectively. We also categorize users as spammers (authors of filtered reviews) or benign (authors without filtered reviews).

Table 1 shows the summary statistics of reviews in these datasets. The two datasets we use are YelpNYC and YelpZip. YelpNYC and YelpZip are different because the YelpNYC dataset only contains restaurant reviews from New York City, but YelpZip contains reviews from NJ, VT, CT, and PA.

These two datasets include reviews for several restaurants from 2004 to 2015. Every review in Yelp datasets consists of a reviewer_id, prod_id, rating, review

doi:10.6342/NTU202304108

date, review context, and label. Rows in Table 2 are snippets of review data. Review in row 1 means a reviewer whose reviewer_id is 933 submitted a review to a product/service whose prod_id is 0 on 2014-01-21 and rated it five on a scale of 1-5, which is labeled as 0, which means "spam reviews".

Table 1: Review datasets used in this work.

Dataset	#Reviews (filtered %)	#Reviewers (spammer %)	#Products (restaurant)	Time
YelpNYC	359,052 (10.27%)	160,225 (17.79%)	923	2004.10-2015.1
YelpZip	608,598 (13.22%)	260,277 (23.91%)	5044	2004.10-2015.1
YelpChi	67,395 (13.23%)	38,063 (20.33%)	201	2004.10-2012.12

Table 2: Examples of reviews

reviewer_id	prod_id	rating	date	review context	label
933	0	5	2014-01-21	pretty cool place...good food...good people	1 (spam)
952	0	5	2014-01-16	Delicious lamb sandwich	0 (benign)

3.2 Comment Graph Construction

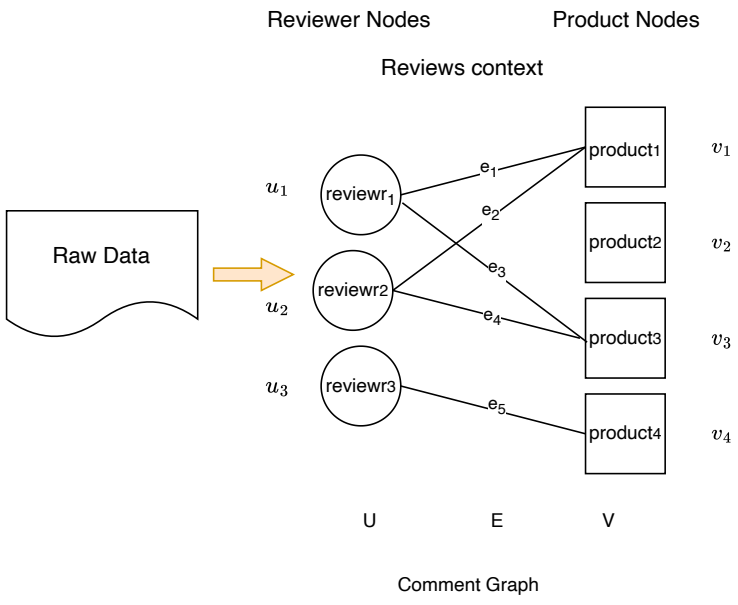


Figure 5: Graph Construction

The system creates embeddings or representations of reviewers, products, and review context to detect spam reviews. The system constructs the comment graph $G(U, V, E)$ in this module, which is a heterogeneous information bipartite graph

doi:10.6342/NTU202304108

made up of nodes in two distinct sets, U (including reviewers) and V. (containing products). An edge $e \in E$ exists means that a reviewer $u \in U$ has submitted a review to the item $v \in V$.

Along with the graph structure, we also assume that each reviewer and product have a real-valued attribute, $x^u, x^i \in \mathbb{R}^d$, associated with the review context submitted by/to, respectively. The edge with the attribution $x_e \in \mathbb{R}^d$ represents the review context, where d is the dimension of the review representation vector.

We present how we embed the review context into sentence vectors in Section 3.2.1, how we use these sentence vectors to generate the edges attribute in Section 3.2.2, and how we use these edges attribute to generate initial nodes attribute in Section 3.2.3 in the following sections.

3.2.1 Review Context Representation

Since word embedding (WE) techniques have demonstrated great performance in spam review detection [19], we employ them to vectorize the review sentence to extract the information of the review context. This is a result of the ability to identify word semantic similarities. In other words, using the WE approach, terms with similar meanings would be considered similar.

Because words can have multiple meanings depending on the context, we employ sentence embedding, an extension of word embedding, to attempt to capture the meaning of the review context. Each review context r is embedded into $\mathbb{R}^d$, where d is the dimension of the sentence vector.

3.2.2 Edge Representation

The system uses a pre-trained Transformer-based [18] language model: SentenceTransformers [20] to encode the review context into a sentence vector $x_e \in \mathbb{R}^d$, d is the dimension of the sentence vector, as the initial edge attribute, to generate the embedding of nodes and edges to identify the spam review further.

3.2.3 Node Representation

We employ mean pooling to combine sentence vectors of review context to generate initial node embedding, inspired by Shehnepoor et al. [21]. Let $\mathcal{N}(v)$ be the neighbors set of node v, $U(e)$, and $V(e)$ denote the reviewer and product nodes that are connected by e. Since we use a bipartite graph to model the reviews data, $\mathcal{N}(u)$ refers to the sets of restaurants $v \in V$ that have been reviewed by a reviewer u. In contrast, $\mathcal{N}(v)$ refers to the sets of reviewers that have submitted reviews on restaurants $v \in V$ that have been reviewed by a reviewer u. The initial node

doi:10.6342/NTU202304108

attribute of node v is defined as:

$$x_v = \begin{cases} \frac{1}{|\mathcal{N}(v)|} \sum \{x_e | U(e) = v\}, & \text{if } v \in U \\ \frac{1}{|\mathcal{N}(v)|} \sum \{x_e | V(e) = v\}, & \text{if } v \in V. \end{cases} \tag{3.1}$$

3.3 Graph Sampling

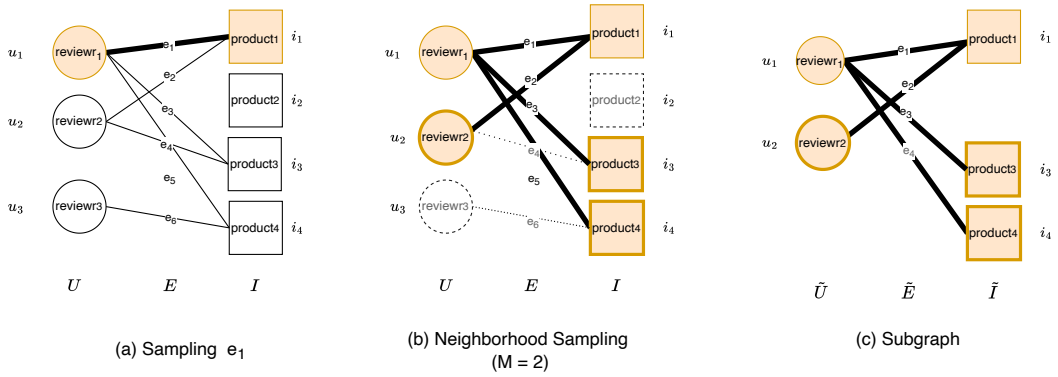


Figure 6: An illustration of edge sampling.

In this work, we employed a mini-batch strategy similar to Ying et al. [22] to update the representation of nodes and edges further to identify the spam review rather than feeding the complete comment network to the HGCN model. As a result of the mini-batch approach's ability to render model size regardless of data size, this method is scalable for use in the criteria of large-scale graphs. To generate a sub-graph for this work, we sample a specific number of neighbors for each node.

The graph sampling approach is shown in Algorithm 1. The system samples a set of edges $E' \subset E$. $\tilde{U}$ and $\tilde{V}$ are stand for the reviewer and product nodes of edge $e \in E'$, respectively. For every edge $e \in E'$, the system samples a specified number M of the reviewer and product nodes' neighbors that are linked to edge e. The edges are then collected by the system and union to E', which is denoted as $\tilde{E}$. Then the sampled reviewer and product nodes set of product and reviewer nodes are unions with $\tilde{U}$ and $\tilde{V}$, respectively.

The illustration of the Algorithm 1 is shown in Figure 6. Let the size of neighbors sample subset $M = 2$ without loss of generality. M is the maximum amount of samples if the size of the neighbors set is greater than M. In the case of Figure 6(a), the initial subgraph $G(\tilde{U}, \tilde{V}, \tilde{E})$ is form by edge e_1, reviewer node u_1 and product node i_1 that linked to e_1, that is $\tilde{U} = U(e_1) = \{u_1\}$, $\tilde{V} = V(e_1) = \{v_1\}$, and $\tilde{E} = \{e_1\}$. Suppose that we sample u_1's neighbors $\{v_3, v_4\}$ and edges set $\{e_3, e_4\}$, v_1's neighbors $\{u_2\}$ and edges set $\{e_2\}$ to add into the subgraph which

doi:10.6342/NTU202304108

Algorithm 1 Graph sampling

Input: comment graph G(U, V, E), edge set E′ ⊂ E, neighbors size M

Output: comment graph subgraph G(Ũ, Ṽ, Ẽ)

$\tilde{E} = \{e \mid \forall e \in E'\}$
$\tilde{U} = \{u \mid u = U(e),\ \forall e \in E'\}$
$\tilde{V} = \{v \mid u = V(e),\ \forall e \in E'\}$
for each $e \in E'$ **do**
  $u = U(e),\ v = V(e)$
  sampling neighbors subset $\tilde{\mathcal{N}}(u) \subset \mathcal{N}(u),\ |\tilde{\mathcal{N}}(u)| = M$
  sampling neighbors subset $\tilde{\mathcal{N}}(v) \subset \mathcal{N}(v),\ |\tilde{\mathcal{N}}(i)| = M$
  $\tilde{E} = \tilde{E} \cup \{e' \mid v = V(e'),\ \forall\, V(e') \in \tilde{\mathcal{N}}(i)\} \cup \{e' \mid u = U(e'),\ \forall\, U(e') \in \tilde{\mathcal{N}}(u)\}$
  $\tilde{V} = \tilde{V} \cup \tilde{\mathcal{N}}(u)$
  $\tilde{U} = \tilde{U} \cup \tilde{\mathcal{N}}(v)$
end for

shows in Figure 6(b). In other words, $\tilde{U} = u_1 \cup \{u_2\}$, $\tilde{V} = v_1 \cup \{v_3, v_4\}$, and $\tilde{E} = e_1 \cup \{e_2, e_3, e_4\}$. At the end, the system return the subgraph $G(\tilde{U}, \tilde{V}, \tilde{E})$ which shown in Figure 6(c).

3.4 Heterogeneous Graph Convolutional Network

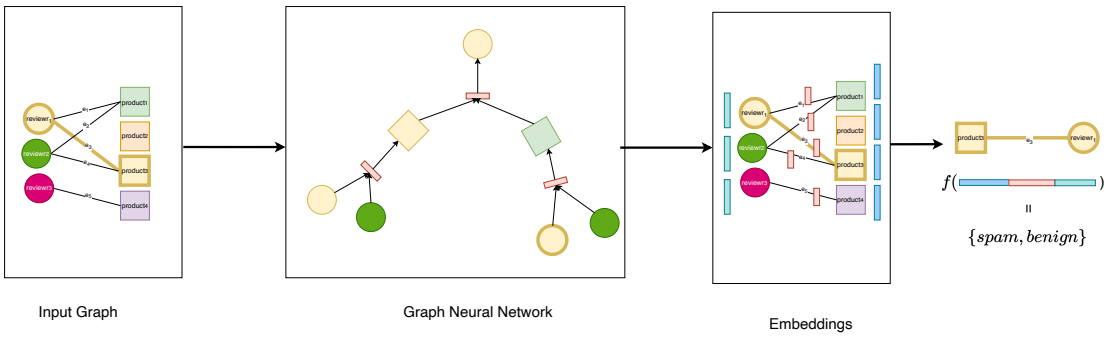


Figure 7: An illustration of the HGCN model training pipeline.

After sampling the subgraph, the system aggregates neighbors' information of a given pair of nodes in the comment sub-graph by implementing the heterogeneous graph convolutional network (HGCN) model to update their node's representation. Then the system uses the representation learned by the HGCN model to identify the spam review. HGCN, being a model based on GCN, harnesses a heterogeneous bipartite graph for its operations. Our system takes shape as a result of layer-wise HGCN propagation on a bipartite graph. In each layer, a simultaneous update for all nodes and edges aggregates the embeddings of surrounding nodes.

doi:10.6342/NTU202304108

A propagation layer can be bifurcated into two distinct phases: aggregation and combination. The aggregation stage and the combination stage are the two main steps that make up a propagation layer. The aggregation and combination stages at any $l-th$ layer where $l \in [1, L]$in an HGCN with L can defined as follows:

$$h^l_{\mathcal{N}(v)} = \sigma\left(W^l \cdot AGG\left(\{h^{l-1}_u, \forall u \in \mathcal{N}(v)\}\right)\right). \tag{3.2}$$

$$h^l_v = COMBIME\left(h^{l-1}_v, h^l_{\mathcal{N}(v)}\right) \tag{3.3}$$

To update h^l_v, the representation of node v in layer l, the model aggregates the representation h^{l-1}_u, where $u \in \mathcal{N}(v)$ and then combines it with the representation of itself. For every node v in the layer l, the model aggregates $h^{l-1}_{v'}$ by using the aggregating function AGG and a learnable weight matrix W^l shared among all nodes at layer l to aggregate h^{l-1}_u, the representation from the previous layer of its neighbors $\forall u \in \mathcal{N}(v)$. After aggregating the representation of $v's$ neighbors, the model using the function $COMBIME$ to combine the information of $v's$ neighbors $h^l_{N(v)}$ and self-representation from the previous layer h^{l-1}_v to generate new representation. The system takes $h^0_v = x_v$ for $v \in U \cup V$ as the initial node representation.

Figure 8 illustrates how $n-$layers HGCN model updates a given reviewer node of the input subgraph with the information of its neighbors. The model iteratively uses aggregate the representation of k-hop neighbors to update the representation of $(k-1)$-hop neighbors for l times.

In the following sections, we explain how the model aggregates information from neighbors in Section 3.4.1 and how it combines the knowledge of individual nodes or edges with that of their neighbors in Section 3.4.2.

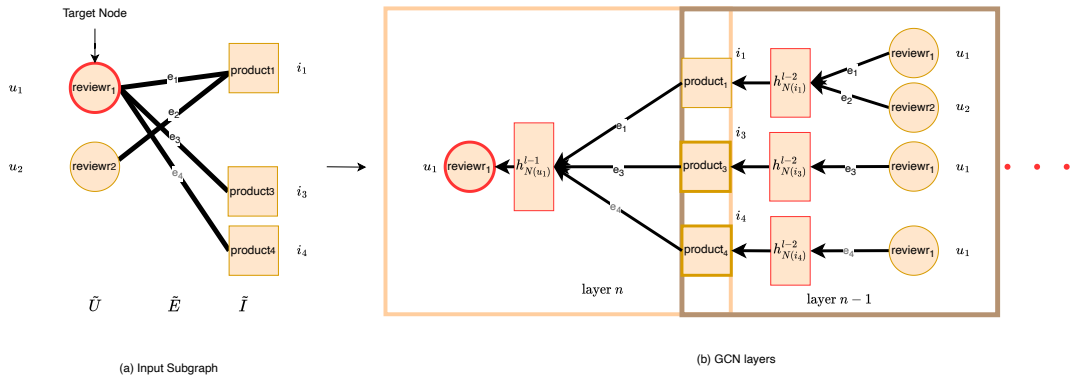


Figure 8: An illustration of how the HGCN model updates the attribute of a given node with input subgraph.

doi:10.6342/NTU202304108

3.4.1 Aggregation Stage

At this stage, the model aggregates neighbors' information to update the embedding of the given node or edge. For example, to output the embedding of u_0 in Figure 8(a), the model needs to update embeddings of i_1, i_3 and i_4 first, then use them to update u_1 to output the final representation. In addition, the model also updates edge embedding by aggregating the representation of its incident nodes.

For the edge in the aggregation stage at layer l, the representation of an edge e is defined as

$$h_e^l = \sigma\left(W_E^l \cdot AGG_E^l\left(h_e^{l-1}, h_{U(e)}^{l-1}, h_{V(e)}^{l-1}\right)\right), \tag{3.4}$$

where W_E^l is a trainable weight matrix shared among all nodes at layer l, σ is a non-linear activation function, and

$$AGG_E^l\left(h_e^{l-1}, h_{U(e)}^{l-1}, h_{V(e)}^{l-1}\right) = \left(\left[h_e^{l-1} \| h_{U(e)}^{l-1} \| h_{V(e)}^{l-1}\right]\right). \tag{3.5}$$

To update h_e^l, the representation of the given edge e at layer l, the model multiplies a layer-wise weight matrix W_E^l with the representation of the nodes e connected to and by itself from the previous layer.

For a review node $u \in U$ and product node $i \in I$, the representation of the neighbors' nodes and edges connected to it are gathered to summarize the representation of neighbors of u and i at layer l which is denoted as $h_{N(u)}^l$ and $h_{N(v)}^l$, respectively. They are defined as follows:

$$\begin{aligned} h_{N(u)}^l &= \sigma\left(W_U^l \cdot AGG_U^l\left(\mathcal{H}_{VE}^{l-1}\right)\right) \\ h_{N(i)}^l &= \sigma\left(W_V^l \cdot AGG_I^l\left(\mathcal{H}_{UE}^{l-1}\right)\right) \end{aligned} \tag{3.6}$$

To update the reviewer node u and the product node i, the model multiplies layer-wise weight matrices, with the gathering of information from their neighbors and edges they connected to, which denote as $\mathcal{H}_{IE}^{l-1}$ and $\mathcal{H}_{UE}^{l-1}$, where

$$\begin{aligned} \mathcal{H}_{VE}^{l-1} &= \left\{\left[h_v^{l-1} \| h_e^{l-1}\right], \forall e = (u, v) \in E(u)\right\} \\ \mathcal{H}_{UE}^{l-1} &= \left\{\left[h_u^{l-1} \| h_e^{l-1}\right], \forall e = (u, v) \in E(v)\right\} \end{aligned} \tag{3.7}$$

For a user node u, the model concatenate $h_{N(u)}^{l-1}$, attribute of product nodes from neighbors set $\mathcal{N}(u)$ and h_e^{l-1}, attribute of edges e connected to u to gathering the information of u's neighbors from the previous layer, said $\mathcal{H}_{IE}^{l-1}$. For a product node i, the model concatenate $h_{N(i)}^{l-1}$, reviewer nodes from neighbors set $\mathcal{N}(i)$ from the previous layer, said $\mathcal{H}_{UE}^{l-1}$.

The model summarizes the information from neighbors of reviewer nodes and product nodes with two kinds of aggregation functions: AGG_U^l and AGG_I^l to

doi:10.6342/NTU202304108

aggregateh_u^{l-1} with $\mathcal{H}_{IE}^{l-1}$ and h_i^{l-1} with $\mathcal{H}_{UE}^{l-1}$ to aggregate the information from neighbors of reviewer nodes and product nodes, respectively. The aggregating functions AGG_U^l and AGG_I^l are defined as follows:

$$\begin{aligned} AGG_U^l\left(\mathcal{H}_{IE}^{l-1}\right) &= ATTN_U\left(h_u^{l-1}, \mathcal{H}_{IE}^{l-1}\right) \\ AGG_I^l\left(\mathcal{H}_{UE}^{l-1}\right) &= ATTN_I\left(h_i^{l-1}, \mathcal{H}_{UE}^{l-1}\right) \end{aligned} \tag{3.8}$$

The attention function, denoted as $ATTN$, operates as a mapping function that can describe as $f : h_{key} \times \mathcal{H}_{val} \rightarrow h_{val}$ [18]. This function enables a transformation that maps a single feature vector, h_{key}, and a collection of candidate feature vectors, $\mathcal{H}val$, to a weighted sum of the elements contained in the collection of candidate feature vectors $\mathcal{H}val$. The importance of the qualities belonging to nodes u or v in relation to the attributes of those nodes' respective neighbor nodes is highlighted by the weighting process.

The model uses the attention function to learn the weight matrix to compute attention coefficients across pairs of attribute $(h_u^{l-1}, h_v^{l-1}),\ v \in \mathcal{H}_{VE}^{l-1}$ or $(h_v^{l-1}, h_v^{l-1}),\ v \in \mathcal{H}_{UE}^{l-1}$.

3.4.2 Combination Stage

After aggregating the information of neighbors, the model combines it with the nodes representation of itself from the previous layer to update the node representation by an equation defined as follows:

$$\begin{aligned} h_u^l &= concat\left(V_U^l \cdot h_u^{l-1}, h_{N_u}^l\right), \\ h_v^l &= concat\left(V_V^l \cdot h_v^{l-1}, h_{N_v}^l\right), \end{aligned} \tag{3.9}$$

where V_U^l, V_V^l are trainable weight matrices for the user and product nodes at the layer l, respectively.

Through iterative updates, We can learn embeddings that contain information gathered regarding the local structure of the graph, in which we combine and aggregate messages from nearby nodes.

3.4.3 Summary of the HGNN Model

The entire Hierarchical Graph Convolutional Network (HGCN) procedure is described by the algorithm 2. The model samples the adjacent nodes of the chosen edges first (lines 1-4) before constructing a computational subgraph. then updates the hidden state of edges using the edge-wise aggregation function *phi* (as shown in line 5). as seen in lines 6 through 14, and performs node-wise message

doi:10.6342/NTU202304108

Table 3: Notation Table

Symbol	Description
U	Set of reviewer nodes
V	Set of product nodes
E	Set of edges
$G(U, V, E)$	Heterogeneous bipartite graph formed by U, V, and E
e	An edge in E
u	A reviewer node in U
v	A product node in V
$\tilde{U}$	Subset of reviewer nodes selected for the subgraph
$\tilde{V}$	Subset of product nodes selected for the subgraph
$\tilde{E}$	Subset of edges selected for the subgraph
$G(\tilde{U}, \tilde{V}, \tilde{E})$	Subgraph formed by $\tilde{U}$, $\tilde{V}$, and $\tilde{E}$
M	Maximum size of the neighbor sample subset
$\mathcal{N}(v)$	Neighbors of node v
h_v^l	Representation of node v at layer l
$h_{\mathcal{N}(v)}^l$	Representation of the neighbors of node v at layer l
W^l	Weight matrix shared among all nodes at layer l
AGG	Aggregating function used by HGCN model
$COMBIME$	Function used by the HGCN model to combine information
$h_{N(u)}$, $h_{N(i)}$	Aggregated representation of neighbors for reviewer and product nodes
$\mathcal{H}_{IE}$, $\mathcal{H}_{UE}$	Gathering of information from neighbors for reviewer and product nodes
AGG_U^l, AGG_I^l	Aggregating functions for reviewer and product nodes
$ATTN$	Attention function
V_U^l, V_I^l	Trainable weight matrices for reviewer and product nodes at layer l

doi:10.6342/NTU202304108

Algorithm 2 HGCN algorithm

Require: 1) Edges set $\tilde{E} \in E$.
2) Numbers of layers L.
3) Functions $V(\tilde{E})$ and $U(\tilde{E})$ that map $\tilde{E}$ to the product and user nodes that $\tilde{E}$ connected.
4) Comment Graph $G(U, V, E)$.

Ensure: The hidden states of product and user nodes z_u,z_v and the review edges z_e in the L-th layer, $\forall e \in \tilde{E}$, $\forall u \in U(\tilde{E})$, $\forall v \in V(\tilde{E})$

Initialize $E^l \leftarrow \tilde{E}$, $U^l \leftarrow U(\tilde{E})$, $V^l \leftarrow V(\tilde{E})$
for $l = L, \ldots, 1$ **do**
  Update $U^{l-1} \leftarrow U^l$, $V^{l-1} \leftarrow V^l$
  Sampling $\mathcal{N}(u)$ and $\mathcal{N}(v)$, $\forall u \in U^{l-1}, v \in V^{l-1}$
  $h_e^l = \phi(h_e^{l-1}, h_{U(e)}^{l-1}, h_{V(e)}^{l-1})\ \forall e \in E^l$
  for $u \in U^l$ **do**
    $H_u^l = \{[h_e^{l-1} \| h_v^{l-1}] :\ \forall v \in \mathcal{N}(u)\}$
    $h_{\mathcal{N}(u)}^l = \phi(H_u^l)$
    $h_u^l = \psi\left(h_u^{l-1}, h_{\mathcal{N}(u)}^l\right)$
  end for
  for $v \in U^l$ **do**
    $H_v^l = \{[h_e^{l-1} \| h_u^{l-1}] :\ \forall u \in \mathcal{N}(v)\}$
    $h_{\mathcal{N}(v)}^l = \phi(H_v^l)$
    $h_v^l = \psi\left(h_v^{l-1}, h_{\mathcal{N}(v)}^l\right)$
  end for
end for
$\mathcal{Z}_e = h_e^l, \forall e \in \tilde{E}$
$\mathcal{Z}_u = h_u^l, \forall u \in \tilde{U}$
$\mathcal{Z}_v = h_v^l, \forall v \in \tilde{V}$

passing *psi* for the node types U and V for each layer. It leverages the hidden states of all nodes and edges after conducting convolutions of overall L layers to create the final embeddings.

3.5 Summary

We provide the dataset used for our study in Section 3.1. Then, Section 3.2 explains the process we used to construct the Comment Graph $G(U, V, E)$ and how the initial graph embedding was also derived, and We describe how to sample a computation graph for stochastic training in Section 3.3. Last but not least, we describe how the Heterogeneous Graph Convolutional Network is used to enhance the social interactions amongst reviewers in Section 3.4, which will be essential for detecting spam reviews.

doi:10.6342/NTU202304108

CHAPTER 4

METHODOLOGY

In the previous chapter, we introduce how we derive the comment graph $G(U, I, E)$ from the review dataset and describe the detection of how the system works. In the following Section, we address the disadvantages of existing work in Section 4.1. Then we describe our model architecture in Section 4.2. We also introduce how we sample the computation graph based on the graph topology structure to train the model in Section 4.3. We further proposed a heterogeneous potential relational graph attention neural network in Section 4.4. Ultimately, we elaborate on how we optimize the model in Section 4.5.

4.1 Motivation

Although Graph Neural Networks (GNNs) have been shown to be effective at leveraging the rich relational information present in graph-structured data for various tasks, their performance is sometimes limited when dealing with imbalanced label distribution because of the GNN message-passing framework.

In the fraud detection tasks, benign entities were significantly more common than fraudulent entities. Therefore, the GNN model would prefer to utilize the information of benign entities to update the embedding of both fraudsters and benign users. Though some existing works like H^2-FDetector [1] utilize both homophilic and heterophilic interactions, they perform such operations on the graph with a homophilic graph with one type of node. They used the reviews and set several rules to construct the graph: nodes represent the review, and different types of edges connect the nodes. However, using nodes to represent the reviews and using a rules-based approach to connect these nodes would construct a large graph that would result in high computation loading. Additionally, they used the full graph to train the model, which is not scaleable. Furthermore, this approach would lose the information between reviewers and reviews, which is important for fraud detection.

To conquer these disadvantages, we proposed a multi-relation graph convolutional method based on the bipartite graph so that the reviewing behavior can be naturally represented, inspired by GAS [17]. They construct the bipartite comment graph to perform the graph convolution and employ a mini-batch stochastic

doi:10.6342/NTU202304108

training strategy to train the graph representation; though the approach can reduce the computation loading, they only consider the homomophic of the graph and sample the nodes.

4.2 Model Architecture

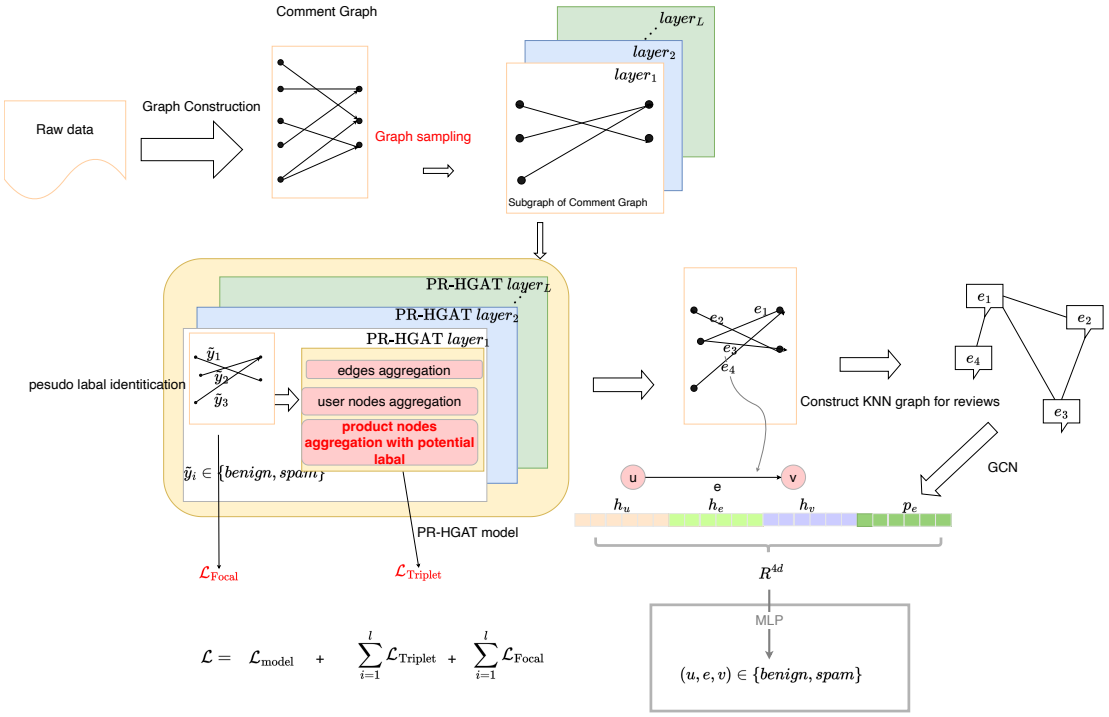


Figure 9: The overall architecture of PL-RGNN.

To take advantage of the low computational loading and ability to aggregate messages between reviews and review targets from GAS [17] and the advantage of learning both homophilic and heterophilic connections and overcome the over-smoothing and label imbalance issues from H^2-FDetector [1], we design a heterogeneous potential relational graph neural network with potential label (PL_RGNN) that inspired by the R-GCN [23]. R-GCN aggregates neighbors under different types of relations separately and combines then combine the categories' messages to update the nodes embedding.

The illustration of the overview of our PL-RGNN model is shown in Figure 9. We construct the comment graph $G(U, I, E)$ that has described Section 3.2. then we sample a batch of edges and their l-hop neighbors to construct the computation graph where l is the number of layers in our PL-RGNN model. to perform stochastic training. Then we utilize the different potential labels as different relations to perform the relational graph attention. Ultimately, we use the loss of the potential-label classifier, distance-based loss between review edges and products, and the loss of PL-RGNN prediction to update our model.

doi:10.6342/NTU202304108

4.3 Topology Aware Graph Sampling

Though GAS [17] already proposed a time-based sample strategy, they only sample the based on the closest neighbors in terms of time. However, an existing study [24, 25] shows that spammers tend to leave their comments in the review burst. Thus under this strategy, the model would remain locally optima easier, and only the top-M closest neighbors to the spam review would aggregate the information from the spam reviews.

Some studies also tried to sample the neighbors with high quality. Both CARE-GNN [26] and PC-GNN [16] measure the similarities between features. Additionally, PC-GNN [16] uses the information of degrees to perform neighbors sampling. They only consider the feature information and local structure from one-hop neighbors; however, the same types of node pairs are at least 2-hop neighborhoods in our tasks. Only considering the local structure of the 1-hop neighbors is not enough.

Inspired by the CARE-GNN [26] and PC-GNN [16], we employ node2vec [27] as node feature of the graph topology structure and measure their cosine similarity as weighted to sample the neighbors to construct the computation graph.

node2vec is a random walk embedding algorithm that is scalable to work on a large graph. It used flexible and biased random walks to trade off the local and global structures with depth-first search and breadth-first search random walks. And we also consider the data imbalance issue by dividing the label frequency. Therefore for a node v, we sample the neighbors with sampling probability:

$$p(v') \propto \frac{\cos(\text{emb}(v), \text{emb}(v'))}{\text{LF}(y_{v'})}, v' \in \mathcal{N}(v), \tag{4.1}$$

$$\cos(\text{emb}(v), \text{emb}(v')) = \frac{emb(v) \cdot emb(v')}{\|emb(v)\| \times \|emb(v')\|} \tag{4.2}$$

where $\text{emb}(v)$ denotes the node2vec [27] embedding of node v, and $\text{LF}(y_{v'})$ denotes the label frequency of class $y_{v'}$.

4.4 PL-RGNN Model

Though the review behavior of the spammer or benign user is under the same relation, their pattern and goal are distinguished; therefore, employing the same aggregation function may not work well. Additionally, due to the data imbalance, the model would update the embedding of nodes and edges mainly from the majority class: benign reviews. To address this problem, we treat benign reviews and spam reviews as different relations rather than the same relation. Motivated by the H^2-FDetector [1], we identify the potential label of reviews and apply a relational

doi:10.6342/NTU202304108

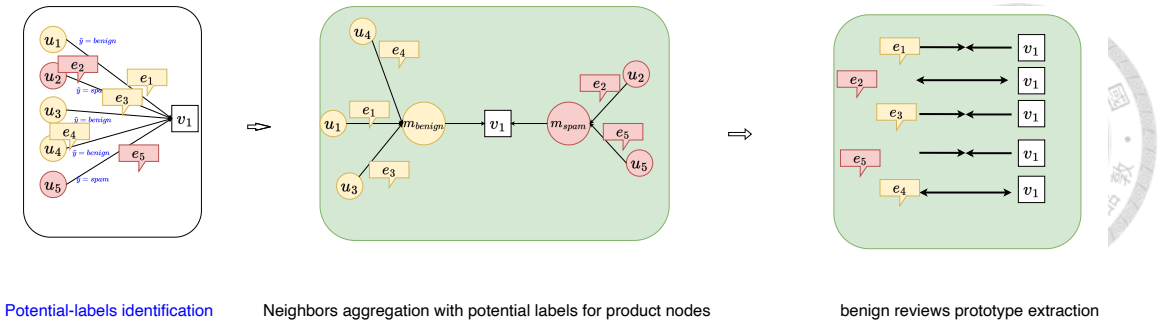


Figure 10: Pipeline of our PL-RGNN for aggregating product nodes.

graph convolution [23] to update the embedding that considers the social interaction of entities in the computation graph. Different from H^2-FDetector [1], we use the aggregation strategy like RGCN [23] to perform the two-stage aggregation: aggregate different potential labels separately and then aggregate the categories information, rather than just minus the features with a heteromorphic connection.

4.4.1 Potential Label Identification

Based on the holomorphic assumption [28]: nodes with the same label are similar, while nodes with different labels are dissimilar, and the observation is more closely related to the review target than spam reviews. so we need to prevent direct aggregation of the information from different labels to achieve better performance.

Inspired by the H^2-FDetector [1], originally designed for a homomorphic graph, we have adapted their homophily and heterophily connection (H^2-Connection) aggregation approach to function within the context of a heterogeneous graph to aggregate the messages from neighboring nodes in the GNN convolution layers to effectively utilizes both homophily and heterophily connections to enhance the performance of graph-based fraud detection. In order to extract the different connection types to treat them differently, we employ a potential-label identification module similar to H^2-FDetector [1] by measuring the similarity or difference between edge features that represent the review content and node features that represent the review target.

Since we utilize a graph neural network to learn its embedding to identify a spam review, U and I represent different types of nodes with attributes that may lie in distinct vector spaces, we first apply separate linear transformation for reviewer node V, product node I, and edge e, then a non-linear activation function. For a node $v \in V$ and $i \in I$, we perform the following operations:

Here, $x_u \in \mathbb{R}^{d_u}$, $x_i \in \mathbb{R}^{d_i}$, and $x_e \in \mathbb{R}^{d_e}$ represent the initial attributes of reviewer node u, product node i, and review embedding, respectively. h_u, h_i, and $h_e \in$

doi:10.6342/NTU202304108

$\mathbb{R}^{d_t}$ denote the transformed features of reviewers, products, and reviews. W denotes a trainable projection matrix to transform the initial nodes and edges attribute, b is the corresponding bias vector, and $\sigma(\cdot)$ is a nonlinear activation function.

Given a comment graph $G(U, I, E)$, we define $H_V^{(l-1)} = \{h_{v_1}^{(l-1)}, h_{v_2}^{(l-1)}, \ldots, h_{v_N}^{(l-1)}\}$ as the set of node embeddings and d_{l-1} as the dimension. We also define $H_E^{(l-1)} = \{h_{e_1}^{(l-1)}, h_{e_2}^{(l-1)}, \ldots, h_{e_M}^{(l-1)}\}$ as the set of review embeddings at layer $l-1$, where $V = U \cup I$.

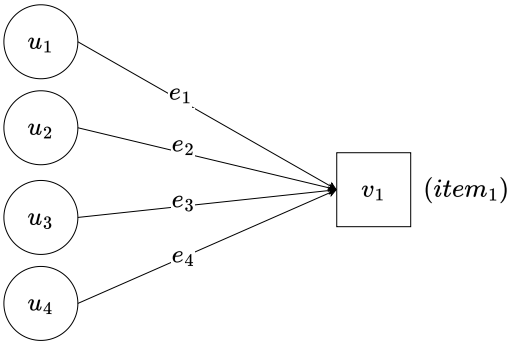


Figure 11: An illustration of the neighbors' aggregation.

To perform such an aggregate approach, we first use a classifier to predict whether a prior review label is spam/benign, then distinguish the spam review embedding from the embedding of review targets. And then, we employ a layer-wise loss the optimize this classifier that describes in Section 4.5

For every convolution layer l, the input to the H^2-connection identification sublayer is obtained from the transformation applied to the previous layer, with the initial input given by $H_U^{(0)} = X_U$, $H_I^{(0)} = X_I$, $H_E^{(0)} = X_E$. For each edge $e_{ui} \in E$, the head (reviewer) node and tail (product) node are denoted by u and i, respectively. For instance, for an edge $e \in E$, its input is $(\bar{h}_u^{(l)}, \bar{h}_v^{(l)})$:

$$\begin{aligned} \bar{h}_u^{(l)} &= \sigma(W_u^{(l)} h_u^{(l-1)}), \\ \bar{h}_v^{(i)} &= \sigma(W_i^{(l)} h_i^{(l-1)}), \\ \bar{h}_v^{(e)} &= \sigma(W_e^{(l)} h_e^{(l-1)}), \end{aligned} \tag{4.3}$$

Here, $h_u^{(l-1)}$ and $h_i^{(l-1)}$ are the embeddings of u and i at layer $l-1$, $W_u^{(l)}, W_i^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ are trainable transformation matrices, and $\sigma(\cdot)$ is a nonlinear activation function.

To capture more nuanced relationships between the review and its target, we utilize the concatenation and difference between transformed embeddings $barh_u^{(l)}$ and $\bar{h}_u^{(l)}$ as input of a one-layer Multi-layer Perceptron (MLP) classifier with a

doi:10.6342/NTU202304108

tanh activation. This classifier predicts a potential label for reviews.

$$m_{ve}^{(l)} = \tanh\left(W_c^{(l)}\left[\bar{h}_v^{(l)} \,||\, \bar{h}_e^{(l)} \,||\, (\bar{h}_v^{(l)} - \bar{h}_e^{(l)})\right]\right) \tag{4.4}$$

Here $W_c^{(l)} \in \mathbb{R}^{3d_l}$ is a learnable matrix that transforms the input features vector into hidden dimension d vector for the classifier, and $[.||.]$ denotes the vector concatenation operation. Then, we can extract the connection types by applying the sign function on $m_{uv}^{(l)}$:

$$\tilde{y}_{ue}^{(l)} = \text{sign}(m_{uv}^{(l)}) \tag{4.5}$$

In the process, if $\tilde{y}_{uv}^{(l)} = 1$, the edge $e_{ui} \in E$ is considered homophilic (potentially benign review). Conversely, if $\tilde{y}_{uv}^{(l)} = -1$, the edge $e_{ve} \in E$ is treat as heterophilic (potentially spam review).

Therefore, we can get the connection types of all edges in graph $G(U, I, E)$:

$$\tilde{Y}^{(l)} = \{\tilde{Y}_{ui}^{\{l\}}\}_{e_{ui} \in E}. \tag{4.6}$$

Then we treat the edges with different potential labels $\tilde{y}$ as different types of relations.

4.4.2 Relational Graph Attention Aggregation

Since spammers and benign users have different interaction behavior with products and other users, we decide to treat them as different types of relations and use different functions to aggregate their features and information rather than using the same function to aggregate but just adding a minus to the heteromorphic connections that H^2-FDetector [1] does.

In our work, we only apply a relational graph attention to aggregate the information from the neighbors of product nodes since benign/spam is only meaningful for products but worth nothing for users. Additionally, even the same review content would result in different labels that leave for different targets. Therefore we update the feature of the user nodes and reviews content (edges), same with the GAS [17].

For a review entity (user u, review target i, review content e), the hidden state of review content e (edge) is defined as:

$$h_{e_{ui}}^{\{l\}} = \sigma\left(\alpha_e^l\left[h_e^{l-1} \| h_u^{l-1} \| h_i^{l-1}\right]\right), \tag{4.7}$$

where h^{l-1} are hidden state from the previous layer, and α is the attention weight.

doi:10.6342/NTU202304108

For a user node u, the message of their neighbors $N(u)$ is calculated as:

$$\begin{aligned} h_{N(u)}^{l} &= \sigma\left(W_{U}^{l} \cdot AGG_{U}^{l}\left(\mathcal{H}_{IE}^{l-1}\right)\right), \\ AGG_{U}^{l}\left(\mathcal{H}_{IE}^{l-1}\right) &= ATTN_{U}\left(h_{u}^{l-1}, \mathcal{H}_{IE}^{l-1}\right), \\ \mathcal{H}_{IE}^{l-1} &= \{\left(h_{i}^{l-1} || h_{e}^{l-1}\right), \forall e = (u,i) \in E(u)\}, \end{aligned} \tag{4.8}$$

where $ATTN$ is the graph attention operation, $\mathcal{H}_{IE}^{l-1}$ are the features set of the neighbors nodes of u and the corresponding edges, and W is the trainable weight matrix. After we aggregate the message from neighbors, we use another weight matrix to combine the message of u from the previous layer and the message from neighbors to get h_u^l, the new hidden state of node u. The message combination approach is defined as:

$$h_{u}^{l} = \left(V_{U}^{l} \cdot [h_{u}^{l-1} || h_{N(u)}^{l}]\right) \tag{4.9}$$

As mentioned earlier, we treat benign reviews of the products and spam reviews of the products as different relations. Therefore we adjust our aggregation strategy for product i as below:

$$\begin{aligned} AGG_{I_{benign}}^{l}\left(\mathcal{H}_{UE}^{l-1}\right) &= \sum_{k=1}^{2} \alpha_{\text{benign},k} ATTN_{Ik}\left(h_{i}^{l-1}, \mathcal{H}_{UE}^{l-1}\right), \\ \mathcal{H}_{UE}^{l-1} &= \{\left(h_{u}^{l-1} || h_{e}^{l-1}\right), \forall e = (u,i) \in E(I) \wedge \tilde{y}_{e} = \text{benign}, \} \end{aligned} \tag{4.10}$$

$$\begin{aligned} AGG_{I_{spam}}^{l}\left(\mathcal{H}_{UE}^{l-1}\right) &= \sum_{k=1}^{2} \alpha_{\text{spam},k} ATTN_{Ik}\left(h_{i}^{l-1}, \mathcal{H}_{UE}^{l-1}\right), \\ \mathcal{H}_{UE}^{l-1} &= \{\left(h_{u}^{l-1} || h_{e}^{l-1}\right), \forall e = (u,i) \in E(I) \wedge \tilde{y}_{e} = \text{spam}\}, \end{aligned} \tag{4.11}$$

we treat the reviews with different kinds of potential labels as different kinds of relations and use the linear combination of basis transformations to compute the attention, thus can learn different representations of different kinds of reviews.

Then we can combine the messages from the spam category neighbors and benign category. It can be calculated as:

$$h_{N(i)}^{l} = \sigma\left(W_{I}^{l} \cdot [AGG_{Ispam}^{l}\left(\mathcal{H}_{UE}^{l-1}\right) \| AGG_{Ibenign}^{l}\left(\mathcal{H}_{UE}^{l-1}\right)]\right). \tag{4.12}$$

By separately training the representation, we can increase the influence of the minority class to reduce the disadvantage of data imbalance and can help us model different kinds of relations better.

After we get the embedding of the review embedding h_e^l, user nodes h_u^l, and

doi:10.6342/NTU202304108

product nodes h_v^l from our PR-HGAT model. In order to aggregate the message from high-order neighbors, we employ the approximate KNN Graph to create a graph based on K nearest neighbors of reviews that are the same as GAS [17] and feed into a single layer GCN model to capture the relation of reviews within different review targets.

$$P_E = GCN(H_E), \tag{4.13}$$

$$GCN(H) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}HW\right), \tag{4.14}$$

where:

- $H_E = \{h_e | e \in E\}$ to aggregate global messages from remote neighbors to get the position embedding p_e.
- $\hat{A}$ is the adjacency matrix of the graph with self-connections added.
- $\hat{D}$ represents the degree matrix of A.
- W is the weight matrix.

Ultimately, we employ an MLP as the classifier to predict whether a review is spam.

$$F\left(z_u, z_v, z_r, p_r\right) = \begin{cases} 0, & \text{if } e \text{ is benign review,} \\ 1, & \text{if } e \text{ is spam review.} \end{cases} \tag{4.15}$$

4.5 Optimization

We update our model with three parts of losses: model predicts loss, potential-label predicts loss, and embedding distance loss to train our model. For the model predicts loss, we employ weighted cross-entropy loss that is frequently used in class classification tasks that can define as:

$$\mathcal{L}_{CE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i}), \tag{4.16}$$

where y_i is the label of $i-$th sample, N is number of sample, and $\hat{y_i}$ is the predicted probability of the i-th sample. And we also employ focal loss [29] to optimize the potential label classifier, and we proposed a triplet-based loss inspired by FaceNet [30] to update the model based on the embedding distance of (review, product) pairs. The potential label predicts loss and embedding distance loss would introduce in the following sections.

doi:10.6342/NTU202304108

4.5.1 Triplet Loss

Triplet loss was first introduced in FaceNet [30]; they optimize the face recognition model by minimizing the loss of triplet samples. A triplet sample includes an anchor (a), a positive sample (p) with the same class as the anchor, and a negative sample (n). The illusion is shown in Figure 12. During training, they dissimilitude the negative samples and assimilated the positive samples with anchor samples. The triplet loss of a triplet $(a,\ p,\ n)$ defines as:

$$\mathcal{L} = max\left(0, \|d(a,p)\|_2^2 - \|d(a,n)\|_2^2 + \alpha\right), \tag{4.17}$$

where α is the margin, the goal is to let the positive is closer to the anchor than the negative by a margin α.

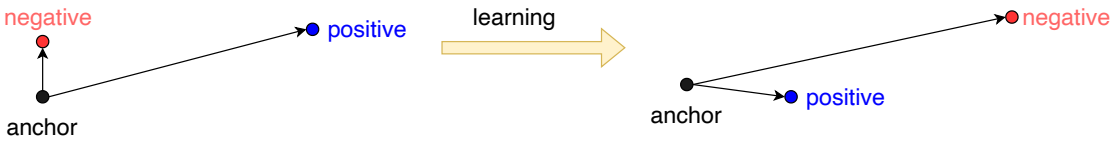


Figure 12: Triplet

Approaches based on Graph Neural Networks (GNNs) play a crucial role in capturing potential social interactions among reviewers, although they are exposed to the over-smoothing problem. This issue occurs when nodes and edges aggregate messages from their neighbors during the message-passing framework within the graph convolution layer, which causes an increase in the similarity of features between nodes and edges. Additionally, embedding fraudulent entities would be harder to distinguish because the sample of benign entities is much more than the fraudulent entities.

Furthermore, in our fraud detection tasks, different fraudster groups would have different behavior patterns; even the same fraud would behave differently when they attack different targets. Thus losses that only evaluate the class prediction performance may not optimize the model well.

Based on such a phenomenon, we proposed a triplet-based loss in our work to distinguish spam and benign reviews by evaluating their distance from products and overcoming the over-smooth issue. However, the production of every possible triplet of samples in our study would result in a significant computational load given the truth that we need to find all valid triplets on (batch_size $\times$ num_neighbors$^{\text{num_layers}}$) samples per batch. Slower convergence is also caused by calculating the loss for each potential triplet. We carefully choose the triplets to ensure that the distances between (negative, anchor) and (positive, anchor) pairs are within the median of all possible pairs. This method avoids poor convergence

doi:10.6342/NTU202304108

rates and prevents evaluating the loss of outlier pairs.

By leveraging the assimilation of samples from the same class and separate classes, we identify reviews with potential negative labels e_{benign} as negative samples, reviews with potential positive labels e_{spam} as positive samples, and product nodes v as anchors in our study. The layer-wise triplet loss defines as:

$$\mathcal{L}^l_{triplet} = \sum_{v \in V} max\left(0, \|d(h^l_v, \tilde{h}_{e^l_{benign}})\|^2_2 - \|d(h^l_v, \tilde{h}^l_{e_{spam}})\|^2_2 + \alpha\right), \tag{4.18}$$

where $\tilde{e}_{benign}, \tilde{e}_{spam}$ are edges connected to v with the median distance with potential positive and negative labels.

Since we treat the reviews with different potential labels as different relations, thus we also need to optimize the potential label classifier to train the model to get the correct relation. In our work, we employ layer-wise Hinge Loss [31] to train the potential-label classifier similar to H^2-FDetector.

4.5.2 Focal Loss

In order to attain a high-quality potential-label classifier to feed into our PR-HGAT model, we adopt the use of Focal Loss [29]. In contrast with traditional loss functions, Focal loss provides an appropriate approach to the common issue of class imbalance by boosting the importance of hard samples while decreasing the impact of easy samples. This reduces bias toward the majority class and enhances generalization to unseen data, both of which are crucial when working with imbalanced datasets. The addition of Focal Loss is essential in improving the performance of our potential-label classifier.

The focal loss is defined as:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t), \tag{4.19}$$

where:

- p_t is the predicted likelihood of the positive class according to the model.
- α_t is a class weighting factor typically employed to address the class imbalance.
- γ is the focusing parameter that balances the contribution of the easy samples and hard samples.

By incorporating Focal Loss into our model, we can maintain equilibrium between positive and negative samples as well as balance the influence of both hard and easy samples. This not only improves the model's performance by focusing

doi:10.6342/NTU202304108

more on difficult situations that are frequently categorized incorrectly, but it also lowers the impact of simple cases and keeps the model from being significantly affected by them. As a result, Focal Loss helps the model to perform better overall, especially when the dataset can be imbalanced.

doi:10.6342/NTU202304108

CHAPTER 5

PERFORMANCE EVALUATION

5.1 Experiment Setup

To show that our proposed method is more effective than related works, we compare the PL-RGNN model with GAS [17] and H^2-FDetector [1]. All of these models are implemented with Deep Graph Library (DGL) [32] library. And we employ pretrain SentenceBert [20] as the sentence embedding model for all reviews.

We generate node attributes for reviewers for both the GAS [17] model and our proposed model by averaging the embedding vectors of the reviews they have written; node attributes for products by averaging the embedding vectors of the reviews that target to.

For the H^2-FDetector [1], we used the embedding vectors of the reviews to represent the nodes. And we also used three relations proposed by CARE-GNN [7] to construct the multi-relation graph. The relations are:

- R-U-R: reviews are written by the same user,
- R-S-R: reviews on the same product that ranked the same star,
- R-T-R: reviews written in the same month on the same product.

In the following experiment, we use the pretrain SentenceBert [20] as the embedding model of review sentences. We directly employ the pretrain sentence-Bert [20] without fine-tuning to assign the embedding of reviews. The initial nodes and edges attribute of the bipartite comment graph for our proposed methods and GAS are given below:

- edges: sentence embedding of reviews,
- reviewer nodes: average sentence embedding of reviews that are given by that specific reviewer.
- product nodes: average sentence embedding of reviews that are given to that specific product.

We use all of the reviews to construct the comment graph, then assign the nodes and edges attributes for the comment graph. After that, we mask the edges to generate the training subgraph, validation subgraph, and testing subgraph.

doi:10.6342/NTU202304108

For our spam review detection task, we use two different dataset split approach to split the training set (70%), validation set (20%), and training set (10%) to learn the generalized power of models, which are:

- random split: the approach that H^2-FDetector [1] used. For GAS [17] and our model, we split by the review edge; while split by the nodes for H^2-FDetector [1].
- time-based: training set for the initial (70%) reviews, validation set for the following (20%), and the last (10%) for the testing set.

Due to the data imbalance issue, we employ AUC as the primary metric as well as the F1-macro and recall to evaluate the model performance. And each experiment is run with ten random seeds and reports the average scores in the following section.

5.2 Evaluate Method

5.2.1 Metrics

We use the recall and F1-score metrics mentioned in Eqs to measure the model performance. 5.1, and 5.2. Here, the terms "true positive" (TP) and "false positive" (FP) are used to denote different types of classifications of fraud: fraudulent classifications as fraudulent and non-fraudulent classifications as fraudulent. Moreover, the terms "false negative"(FN) and "true negative" means fraudulent, classified as non-fraudulent, and non-fraudulent, classified as non-fraudulent, respectively. Figure 13 shows these four terms' conditions.

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{5.1}$$

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}, \tag{5.2}$$

the F1 score is the harmonic mean of the precision and recall, and the relative contributions of each to the F1 score are equal. Since the label of the dataset is an imbalance, we employ F1-macro, which calculates F1-score for every class and find their unweighted mean.

In addition to recall and F1-score for measuring the model's performance, we also look at the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) and the Precision-Recall curve (PRC) to comprehend the trade-off the model makes between sensitivity and specificity.

The AUC-ROC curve is a performance indicator for categorizing faults at different threshold values. The TPR vs. the FPR with different threshold settings

doi:10.6342/NTU202304108

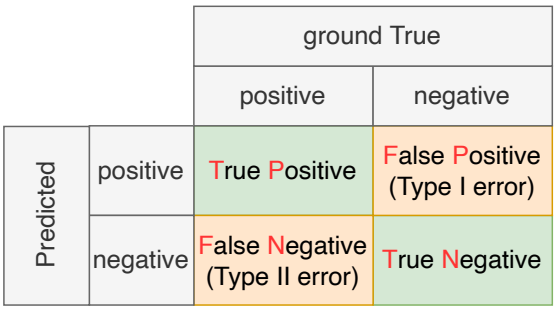


Figure 13: Confusion matrix

is displayed on a probability curve known as a ROC. The "area under the ROC curve" (AUC) is an overall performance measure across any prospective classifying criteria. The AUC of a model with 100% incorrect predictions is 0.0, while the AUC of a model with 100% correct predictions is 1.0. AUC evaluates how well predictions are scored rather than their absolute values, making it useful even when classes are highly unbalanced.

This is an additional tool to evaluate the classified model's efficacy. It plots the precision (y-axis) and recalls (x-axis) based on various thresholds, much like the ROC curve. Precision assesses the applicability of the findings, whereas recall reflects the volume of truly relevant results returned. The weight in the AUC for the PRC curve (AUC-PRC), which determines the weighted average of precisions at each threshold, is the increase in recall from the preceding threshold.

5.2.2 Visualization

To illustrate the similarity between review sentences, we employ t-Distributed Stochastic Neighbor Embedding (t-SNE), a widely used unsupervised machine learning technique, to map the high-dimensional data into a 2D environment. It facilitates data structure analysis of the local similarity patterns by mapping complicated, multidimensional data to a lower-dimensional space. This method is a manifold learning approach that is renowned for its potency in non-linear dimensionality reduction.

5.3 Average Performance Analysis

We conduct a thorough experimental evaluation to determine the effectiveness of our model. Three different datasets are used in this study, which is conducted under two key evaluation criteria. We aim to evaluate the model's associated effectiveness in its purpose and capacity to generalize and stabilize under different conditions. Besides these three yelp datasets, we also compare the model performance between baseline models and our models on the Amazon review dataset in

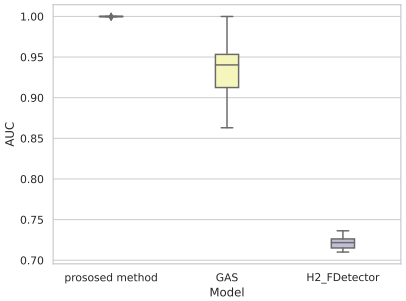
doi:10.6342/NTU202304108

Section 5.8.

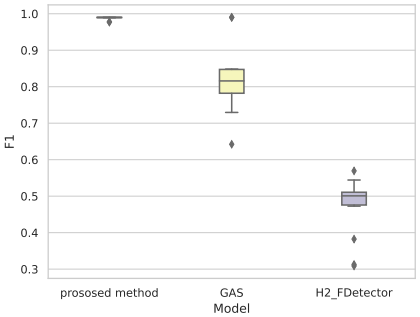
Table 4: Model performance on YelpChi

	Random			Split based on time		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
GAS	0.9999	0.9912	**1**	0.9375	0.8241	0.8450
H^2-FDector	0.8962	0.7220	0.8104	0.7216	0.4725	0.8151
our	**1**	**0.9966**	0.9966	**0.9998**	**0.9874**	**0.9882**

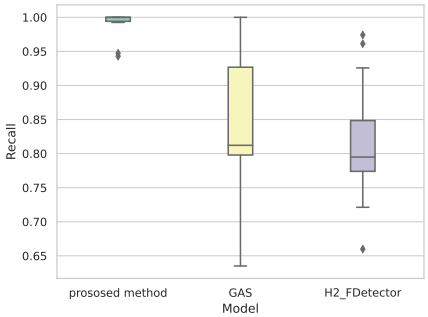
The experiment results on the yelpChi dataset are shown in Table 4. Overall, under both situations, our model beats both the GAS and H^2-FDector models in terms of all three metrics. Our model displays almost ideal results for all three metrics, especially for the split based on time, while others perform significantly worse. Since our model and GAS are almost ideal under the random split situation, we only report the box plot under the time-based split situation, shown in Figure 14.



(a) AUC score



(b) F1 score



(c) Recall

Figure 14: AUC, recall, and F1 of models on YelpChi under the time-based split

The lowest F1 score for the GAS model is much lower than the lowest F1 score for our model. The F1 values for the GAS model are often lower compared to our model, the AUC and recall scores for the GAS model are generally good but show

doi:10.6342/NTU202304108

greater variation, especially for the recall. Lastly, the H2-FDector model has high variability and typically receives worse results across the board.

Table 5: Model performance on YelpNYC and YelpZip under the random split.

	Yelp_NYC			Yelp_Zip		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
GAS	0.8246	0.6191	0.7151	0.8274	0.6341	0.7128
H^2-FDector	0.7800	0.5149	**0.7988**	0.6843	0.4838	0.7010
our	**0.8400**	**0.6224**	0.7424	**0.8410**	**0.6491**	**0.7333**

Table 6: Model performance on YelpNYC and YelpZip under the time-based split.

	Yelp_NYC			Yelp_Zip		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
GAS	0.6116	0.4214	0.6513	0.6116	0.4214	0.6513
H^2-FDector	0.6220	0.4229	**0.7217**	0.6285	0.4127	0.7282
our	**0.6782**	**0.4554**	0.7124	**0.6573**	**0.4580**	**0.7330**

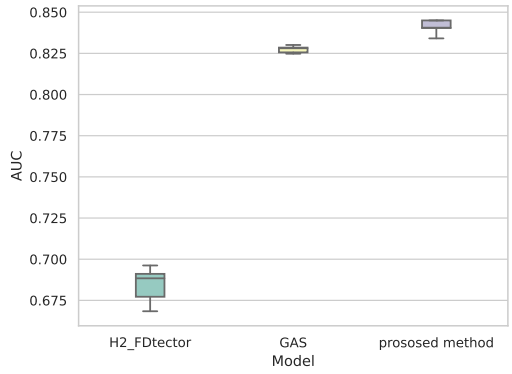
The experiment result of model performance under the two situations is provided in Table 5 and Table 6. Table 5 shows the models performance under the random split approach; we can find that our model slightly outperforms the other two baselines with both datasets on AUC and F1 scores but lower on recall in the yelpNYC dataset. Under the time-based split approach, the gap in model performance between our model and the baseline model is even greater.

In conclusion, our model consistently performs at a high score across metrics and datasets on average score, which shows that our model has a greater generalize ability. Besides the average scores of these metrics, we also report the box plot of our experiment result to help us illustrate the stability of these models.

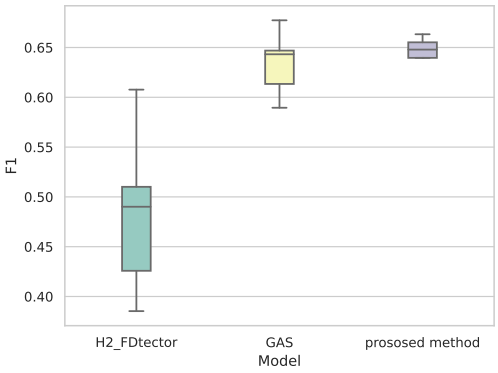
Analyzing the data displayed in Figure 15 reveals that compared to the two baseline models based on AUC score. Additionally, our model consistently outperforms the greatest results attained by the other two models, even at its worst performance. Though our model is sometimes worse on F1 score and recall, we perform better on average and have the lowest variance among all models, proving our method's stability.

The experiment result on YelpNYC is illustrated in Figure 16. Compared to the result in Figure 16, our model still outperform based on AUC, though sometimes lower on F1 but have a small variance among the three. Additionally, our average recall is greater than H^2-FDetector [1] by 3%, sometimes lower than it. However,

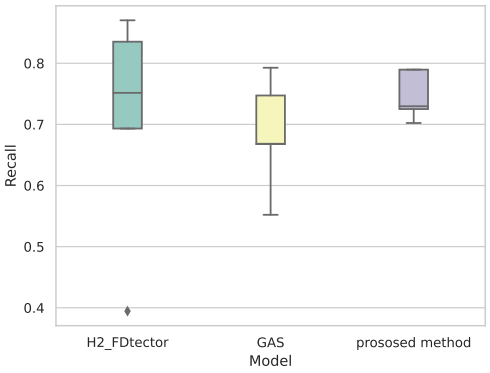
doi:10.6342/NTU202304108



(a) AUC score



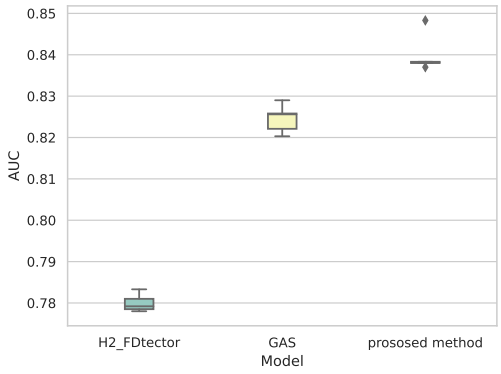
(b) F1 score



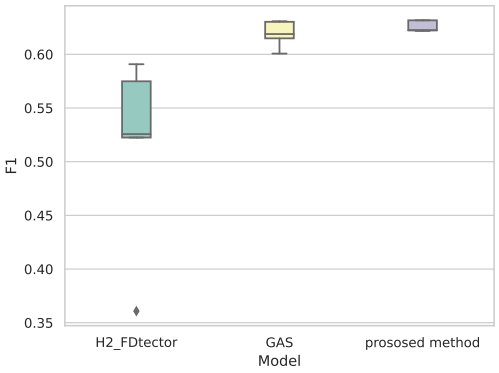
(c) Recall

Figure 15: AUC, recall, and F1 of models on YelpZip under the random split approach.

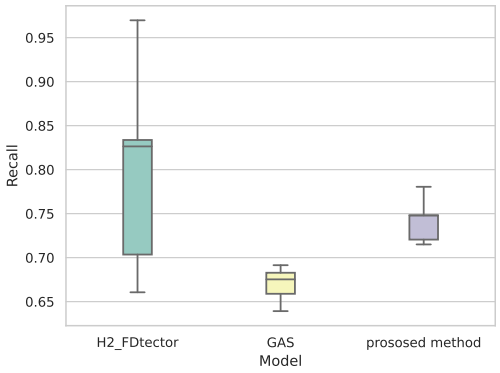
doi:10.6342/NTU202304108



(a) AUC score



(b) F1 score

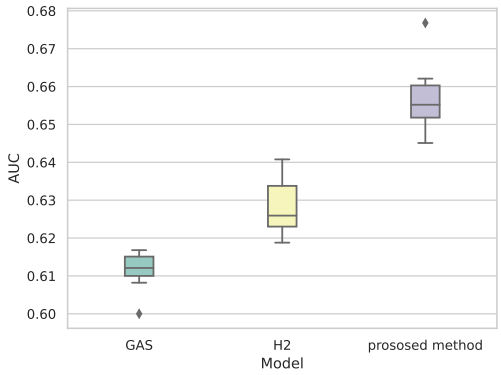


(c) Recall

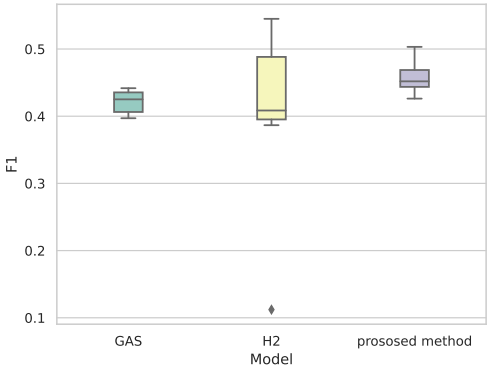
Figure 16: AUC, recall, and F1 of models on YelpNYC under the random split approach.

doi:10.6342/NTU202304108

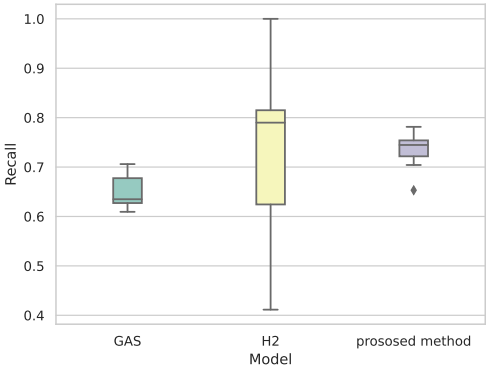
the variance of the F1 score on H^2−FDetector [1] is much higher, which means that it is very unstable.



(a) AUC score



(b) F1 score



(c) Recall

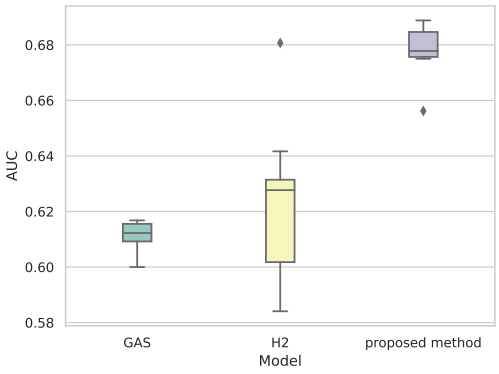
Figure 17: AUC, recall, and F1 of models on YelpZip under the time-based split approach.

Results in Figure 17 show that when compared to baseline models, our model continues to perform better in terms of AUC score. Interestingly, the AUC score of GAS is lower than H^2−FDetector, which deviates from the pattern shown in Figure 15(a). Additionally, as seen in Figure 17(b) and Table 6, our model has the highest average and median F1 scores out of the three models.

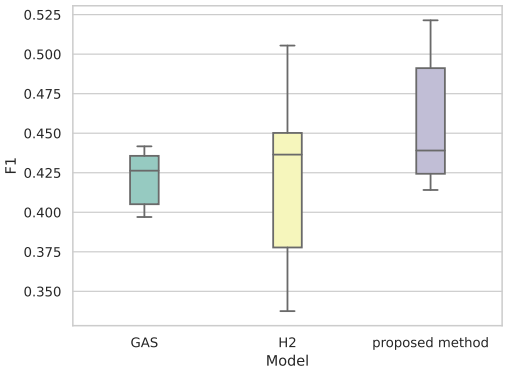
A closer look at the recall scores demonstrates that our model outperforms the H^2−FDector in terms of performance. Although the average Recall score of the H^2−FDector is slightly higher than that of our model, it is important to note that the recall of this model is much more unstable. This variation in recall scores raises the risk that the H^2−FDector is untrustworthy.

The result of the experiment of the yelpNYC dataset under the time-based split is shown in Figure 18. Our model still outperforms the baseline model in terms of the AUC score, though the medium of the AUC of the ^{2}−FDetector is

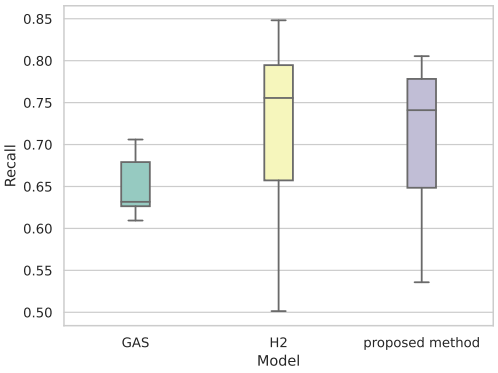
doi:10.6342/NTU202304108



(a) AUC score



(b) F1 score



(c) Recall

Figure 18: AUC, recall, and F1 of models on YelpNYC under the time-based split approach.

doi:10.6342/NTU202304108

higher than GAS, but the GAS is much more stable. And our F1 score performs better than others in general, though the medium of these three models is very similar; both the Q1 and Q3 of F1 with our model are greater than other baseline models. However, our performance in terms of Recall may not be as good as other models, but the Q1 of our model is greater than the Q2 of GAS, and the medium of our model is greater than the maximum of the GAS.

In conclusion, Our proposed model consistently beats the two baseline models—GAS and H^2-FDector—across all experiments and datasets in most cases. The model performs better overall regarding the AUC score, particularly in experiments under time-based splits. Even as the performance of the model reduces, it continues to outperform the top-performing baseline models.

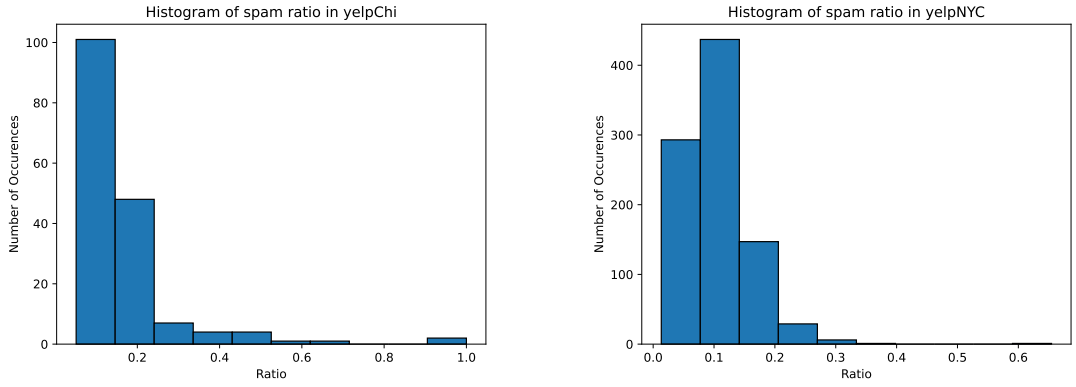
While there are some situations where it does sometimes get worse, the low variance in the F1 scores, as opposed to the higher variability shown in the H2−FDector model, provides evidence for the stability of our model.

With regard to recall, the model displays mixed performance. While the average recall is 3% higher than H^2-FDector, there are some situations where the recall is lower. Considering this, the model outperforms the H^2-FDector in recall scores, which shows significant unpredictability and worse reliability.

The model performs well across all situations, with especially high average scores across metrics and datasets, as a result. This demonstrates the model's outstanding generalizability. In addition to these average outcomes, the box-plot results demonstrate the model's consistency, demonstrating how it performs under various scenarios.

In our experiments, we find that the AUC score of our proposed methods is near 1 in terms of AUC, recall, and F1. To figure out the reason, we dive into the statistic of the dataset to have a clear picture. First, we calculate the spam review ratio of products to know if there are any biases in this dataset; the histogram of the spam review ratio for the yelpChi and yelpNYC is shown in Figure 20.

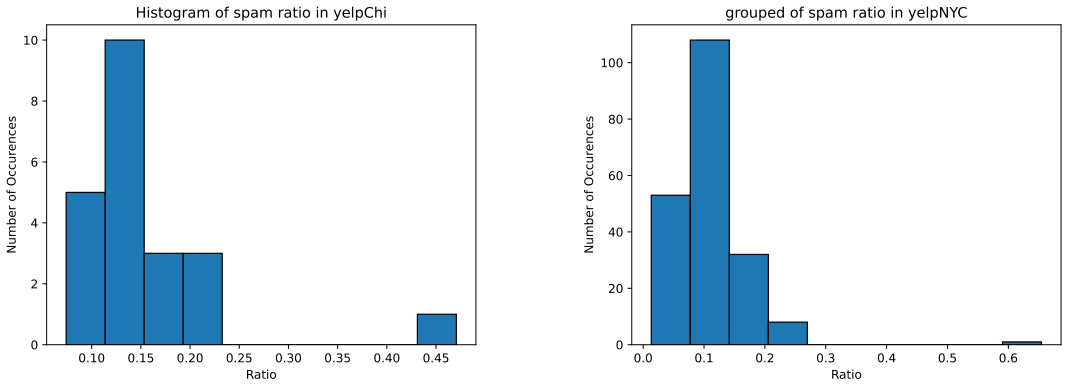
doi:10.6342/NTU202304108



(a) yelpChi (b) yelpNYC

Figure 19: Spam review ratio for products.

Both of the datasets have many products with a high ratio of spam reviews or a low ratio of spam reviews, so the spam review ratio within all products may not be a key reason for making the performance on yelpChi significantly better.

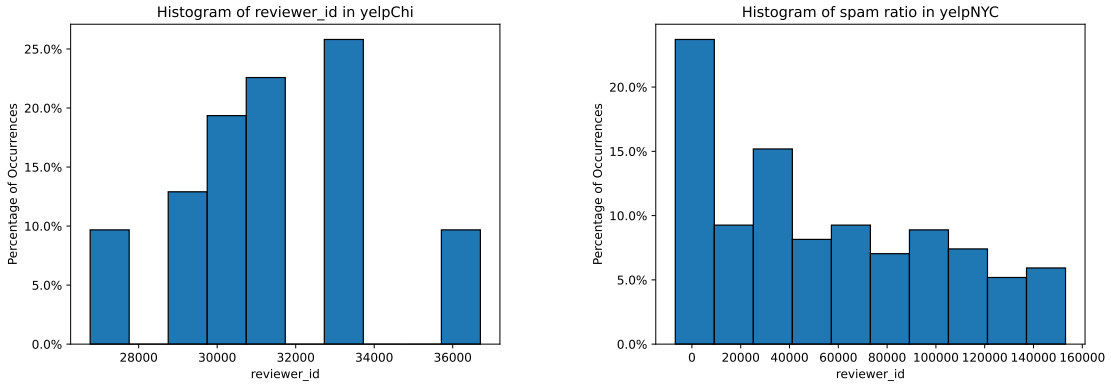


(a) yelpChi (b) yelpNYC

Figure 20: Spam review ratio for products that predict failure.

Then, we collect the product of reviews; if we predict whether a review is a spam failure and calculate the spam review ratio of that product in the entire dataset, we can observe that the distribution of the spam ratio change is notable. Therefore we further analyze whether the distribution of failure-predicted review of these datasets is different.

doi:10.6342/NTU202304108



(a) yelpChi (b) yelpNYC

Figure 21: Distribuction of reviewer id.

We can find that the distribution of reviewer id is very different. In the Yelp dataset we use, the record is sorted by the products, then we assign the reviewer id to reviewers increasing; the larger the reviewer id is, the later records the reviewer's first comment. In figure 21, we can find that the failure-predicted reviewer in yelpChi is distributed in a certain interval, which indicates that in this dataset, there are lots of reviewers who only review certain products. Therefore if we know which product the spam reviews target, we can identify spam reviews easily, moreover since we design the benign prototype extraction to lower the similarity between spam review embeddings and product nodes embedding and larger the similarity between benign review embeddings and product nodes embedding if a benign review given to a spam target product, we can better distinguish then, thus can further boost the performance to a higher level.

5.4 Model Trade-off Analysis

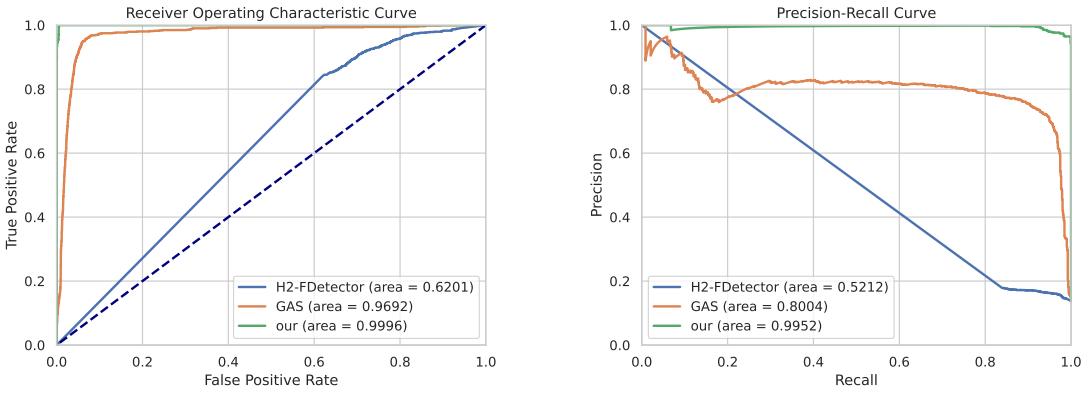
In this section, we examine Receiver Operating Characteristic (ROC) and Precision-Recall curve (PRC)analysis. These provide insight into the trade-offs that our proposed model and other baseline models suffer in performing at various threshold levels. This will allow us to comprehend these models' performance under different classification thresholds better.

- ROC Curve plots the FPR vs. TPR, which can help us illustrate the model trade-off between sensitivity and specificity.
- PRC illustrates the trade-off between precision and recall; a big area under the curve denotes both high recall and high precision.

In the following analysis, we used the model state of the last to test the model performance under different thresholds. Since two of the three models display

doi:10.6342/NTU202304108

almost ideal results on the yelpChi dataset under the random split, so we do not show the PRC and AUC-ROC curves on this dataset.



(a) yelpChi (b) yelpChi

Figure 22: ROC and PRC Curve under the random split.

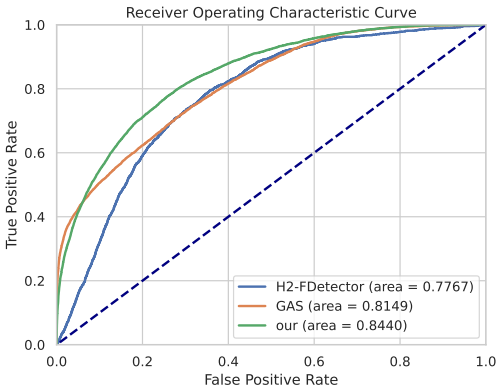
Figure 22 illustrates the results of the examinations we performed on the YelpChi dataset under the time-based split criteria. A full comparison of the evidence shows that our proposed model outperforms all other baseline models across various thresholds. Although it looks to have a larger area under the PRC than H^2-FDector, it's essential to note that H^2-FDector performs better when it comes to high precision.

Figure 23 illustrates the experiment with a random split; our model has a higher sensitivity and is more capable of accurately detecting positive examples. The area under the ROC curve is higher than others, which indicates that it has a superior trade-off between sensitivity and specificity with respect to various threshold settings.

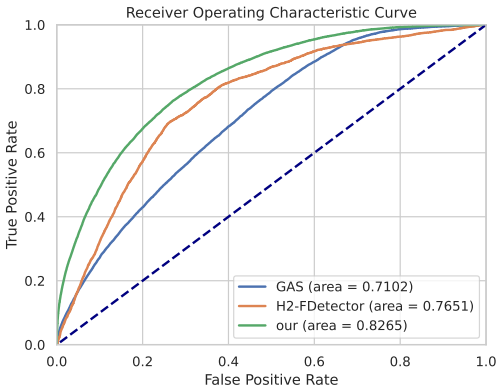
Furthermore, the PRC also indicates our model's better performance which has a wider area under the PRC, which suggests that it has a greater accuracy at various recall levels. As a result, not only is our model effective. As a result, in addition to being our model effective at detecting positive instances (high recall), it also makes sure that the majority of cases that are expected to be positive are really positive.

In summary, the ROC and PRC curves suggest our model performs better than others in terms of sensitivity and precision over a wide variety of threshold values. This makes our model a trustworthy and solid option for this work.

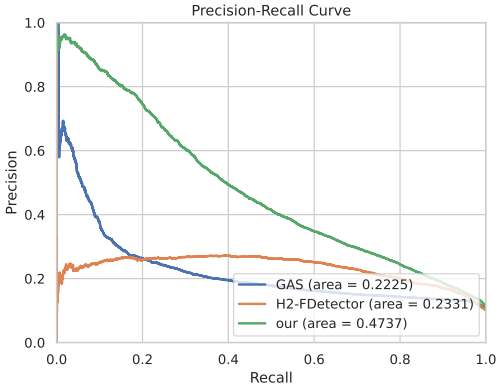
doi:10.6342/NTU202304108



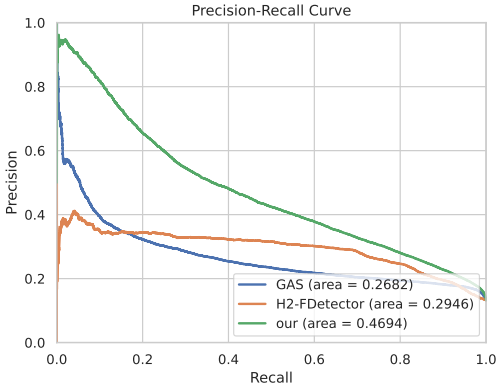
(a) yelpNYC



(b) yelpZip



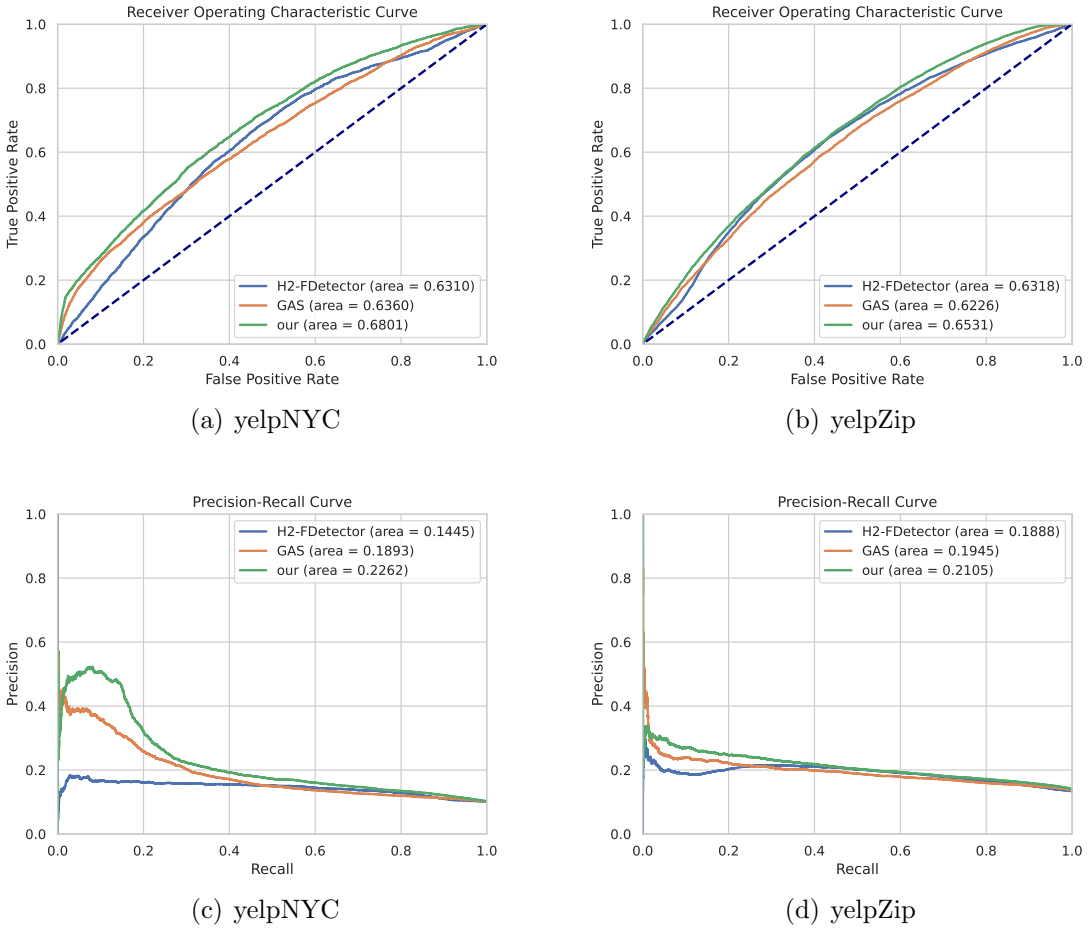
(c) yelpNYC



(d) yelpZip

Figure 23: ROC and PRC Curve under the random split.

doi:10.6342/NTU202304108



(a) yelpNYC (b) yelpZip

(c) yelpNYC (d) yelpZip

Figure 24: ROC and Curve under the time-based split.

Figure 24 illustrates the experiment with a time-based split. When evaluating the stability and reliability of the model in real-world settings where data is frequently temporal and unstable, this figure gives an overview of the model's performance. There is a noticeable variation between this figure and Figure 23, which shows the model's performance in the lack of a random split. Although there is a reduction in our model's performance in the time-based split scenario, our model still manages to outshine the other two baseline models in terms of both ROC and PRC across almost all thresholds.

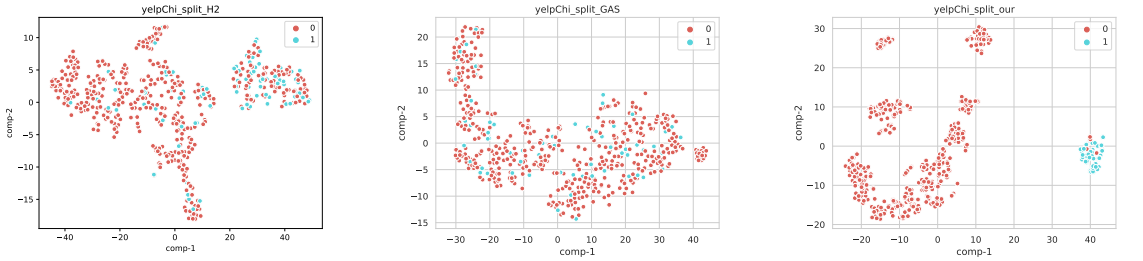
In conclusion, our model performs better than others across a variety of threshold settings and shows improved capacity to balance sensitivity and specificity under various assessment criteria.

5.5 Embedding Visualization

To visually illustrate the local similarity of the reviews inside an individual set, we use the method of t-SNE in this section. By assigning each data point a position on a two-dimensional map, t-SNE is an effective probabilistic approach

doi:10.6342/NTU202304108

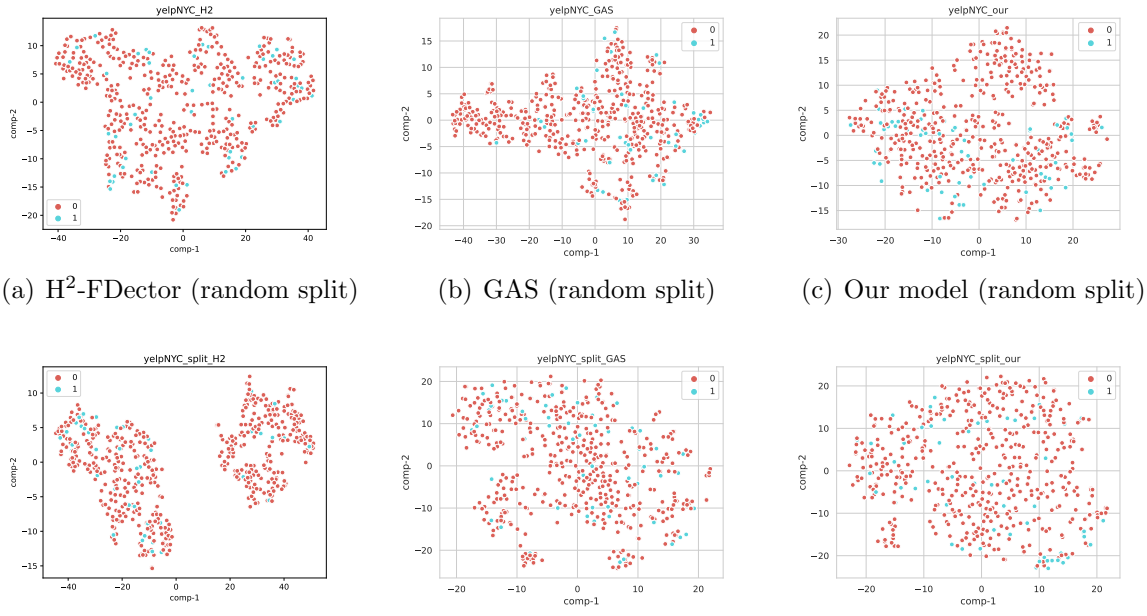
for displaying high-dimensional data. It works especially well for the display of complex data structures.



(a) H^2-FDector (b) GAS (c) Our model

Figure 25: t-SNE on the yelpChi under the time-based split.

Figure 25 presents the t-SNE embedding of the reviews from the yelpChi dataset under the time-based split; it shows that our methods successfully separate benign reviews from spam, which the other two baseline approaches cannot do. Although there is a distinct distribution for spam and benign reviews in the case of the H^2-FDetector, a significant amount of the spam reviews are tightly grouped with a group of benign reviews, making it difficult to distinguish between the two.



(a) H^2-FDector (random split) (b) GAS (random split) (c) Our model (random split)

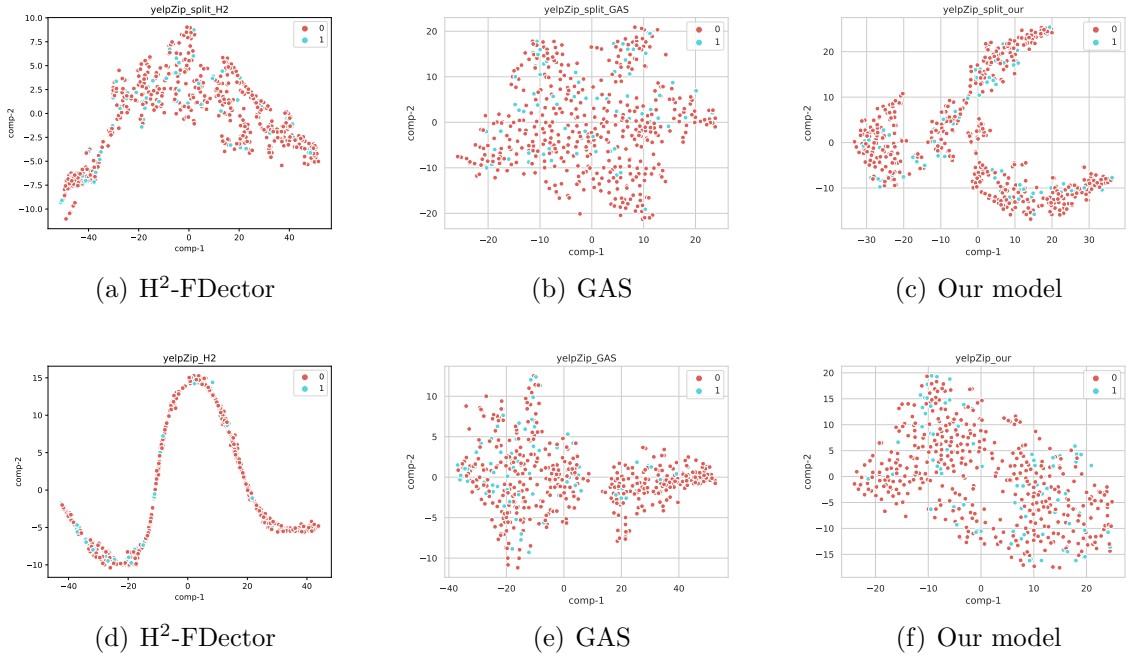
(d) H^2-FDector (time based split) (e) GAS (time based split) (f) Our model (time based split)

Figure 26: t-SNE on the yelpNYC.

Figure 26 illustrates the review embeddings of the experiment result on the yelpNYC dataset. The embedding of the experiment result of H^2-FDector under the time-based split is similar to Figure 25(a), which is also tightly grouped with

doi:10.6342/NTU202304108

a group of benign reviews. Comparing our experiment results with GAS reported similar results, but the embeddings of GAS are much close, which means that it may result in a more serious over-smoothing issue than our proposed model.



(a) H^2-FDector (b) GAS (c) Our model

(d) H^2-FDector (e) GAS (f) Our model

Figure 27: t-SNE on the yelpZip under the time-based split.

Figure 27 illustrates the experiment results on the yelpZip dataset; all of the models perform worse than the experiment results on the yelpZip dataset shown in Figure 26 in general, especially for H^2-FDector under the time-split approach result in a serious over-smoothing issue; we can not even distinguish any spam review from benign review, which means that they are very closed in the embedding space, though embedding for spam and benign reviews with both of our model and GAS are not easy to distinguish, the difference of distribution between benign reviews are better, and the distribute of the review embedding are more sparse. Thus we conclude that we have better performance.

5.6 Performance Comparison of Different Embedding Methods with Baselines

In the following experiments, we directly employ different sentence embedding methods to compare the performance of our model; the embedding methods we use are:

- SentenceBert: a BERT model modification made specifically for embedding sentences. It makes use of transformer architectures to take into account

doi:10.6342/NTU202304108

semantics at both the word and sentence levels, creating a dense vector space that encodes complex interactions between sentences.

- CBOW: The Word2Vec approach's Continuous Bag-of-Words paradigm. It attempts to forecast the term that best fits the context by using the context of each word as input. The embeddings in this context-sensitive model are trained by optimizing the prediction error.

- glove: The approach that places an emphasis on the statistical link between words is called Global Vectors for Word Representation. It generates representations that incorporate both local word meanings and overall corpus statistics by computing word co-occurrence matrices and factorizing them.

For CBOW embedding, we train the model on the entire review sentences first and then use this model to generate the word vectors of each word in the review sentences and take the average as the final sentence embedding of the given review.

For glove embedding, we employ the pretrain word vector that training on Wikipedia to vectorize the word and average these word vectors in review sentences to generate the final sentence embeddings.

The performance of our proposed methods with different embedding methods is reported in Table 7 and Table 8. In table 7, we can observe that the sentenceBert outperforms other embedding methods in terms of AUC and F1 but has a lower Recall than CBOW; in table 8, the sentenceBert only outperform others in term of AUC and recall on YelpNYC dataset. Though the Glove embedding method performance is better than others on the YelpZip dataset under the time-based split scenario, but the AUC scores under the random split scenario are the lowest among the three; therefore, we choose SentenceBert embeddings in our proposed methods.

Table 7: Model performance on YelpNYC and YelpZip under the random split.

	Yelp_NYC			Yelp_Zip		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
CBOW	0.7909	0.5653	0.6472	0.7939	0.5653	**0.7951**
Glove	0.7474	0.6119	0.4693	0.7705	0.5750	0.7205
SentenceBert	**0.8400**	**0.6224**	**0.7424**	**0.8410**	**0.6491**	0.7333

doi:10.6342/NTU202304108

Table 8: Model performance on YelpNYC and YelpZip under the time-based split.

	Yelp_NYC			Yelp_Zip		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
CBOW	0.6444	0.4248	0.7107	0.6412	0.4469	0.7436
Glove	0.6515	**0.4755**	0.6169	**0.6975**	**0.4825**	**0.7443**
SentenceBert	**0.6782**	0.4554	**0.7124**	0.6573	0.4580	0.7330

5.7 Ablation Study

In this section, we analyze the effect of the topology-aware sampler, our PR-HGAT model, and our optimization approach in our proposed method to figure out the effect of these three components.

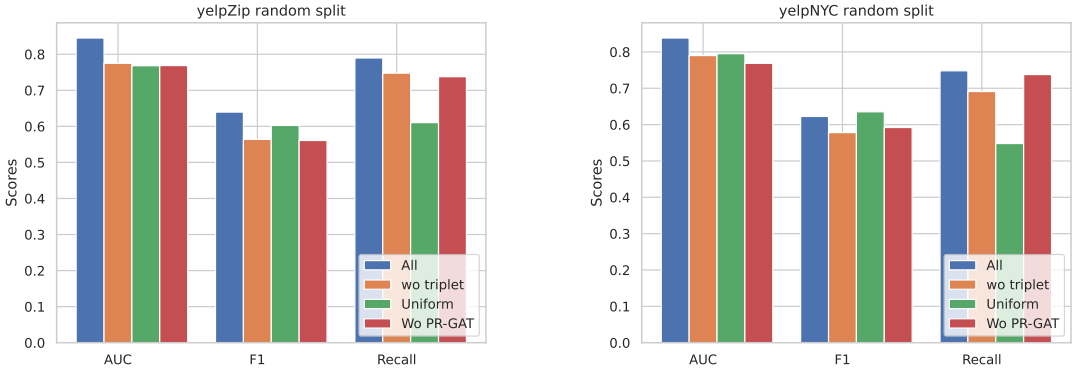
In the following experiment, **All** represents our full model with all components that we mention; **wo triplet** means that we removed the triplet to train our proposed model; and **Uniform** means that we employ a uniform sampler to sampler neighbors to form the computation graph to train our model; at the end **Wo_PR-GAT** means that use removes the pseudo-relation aggregation mechanism, teat spam reviews, and benign reviews as the same relation, which equal multi-head graph attention neural network.

Table 9: Ablation Study on YelpNYC and YelpZip under the random split.

	Yelp_NYC			Yelp_Zip		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
All	**0.8382**	0.6224	**0.748**	**0.845**	**0.6396**	**0.7896**
wo triplet	0.7899	0.5779	0.6910	0.7749	0.5639	0.7476
Uniform	0.7951	**0.6352**	0.5479	0.7681	0.6025	0.6104
Wo_PL-GAT	0.7686	0.5920	0.7377	0.7686	0.5608	0.7377

The experiment result of the ablation study under the random split approach is reported in Table 9 and illustrated in Figure 28, we can find that without one of any components, the AUC score would have a significant drop, and the performance are also outperforms others model in terms of Recall. Though the F1 score employs a uniform sampler, its recall score of it is the lowest of all, which indicates that it can not efficiently recognize spam reviews.

doi:10.6342/NTU202304108



(a) yelpZip (b) yelpNYC

Figure 28: Ablation study under the random split.

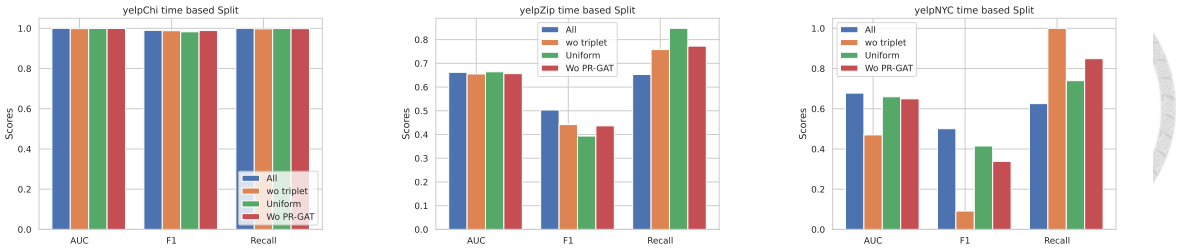
Table 10: Ablation study on YelpChi

	time-based split		
	AUC	F1 macro	Recall
All	0.9999	0.9899	1.0000
wo triplet	0.9983	0.9880	0.9972
Uniform	0.9994	0.9827	0.9989
Wo PR-GAT	0.9992	0.9889	0.9983

Table 11: Ablation Study on YelpNYC and YelpZip under the time-based split.

	Yelp_NYC			Yelp_Zip		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
All	**0.6778**	**0.5007**	0.6255	0.6621	**0.5032**	0.6531
wo triplet	0.4699	0.0912	**1**	0.6552	0.4418	0.7580
Uniform	0.6595	0.4147	0.7396	**0.6643**	0.3935	**0.8476**
Wo_PR-GAT	0.6493	0.3382	0.8490	0.6565	0.4364	0.7726

doi:10.6342/NTU202304108



(a) yelpChi (b) yelpZip (c) yelpNYC

Figure 29: Ablation study under the time-based split.

The numeral ablation study results under the time-based split approach are shown in Table 10 and Table 11. The illustration is shown in Figure 29. The "full" model has the highest AUC scores across two datasets, as well as the F1 score, which indicate that it has the best overall performance in terms of both precision and recall. and the ability to distinguish between spam and benign. However, the recall is lower than others.

As seen in Figure 29(c), the F1 score on the YelpNYC dataset decreases to almost zero when the triplet loss component is taken out, and the recall score of the model increases to 1, which means that the results are significantly skewed. Thus the triplet loss is the key component within our model.

Similar to the experiment result without the triplet loss, the model's performance suffers from biased predictions without our pseudo-relation aggregation technique, as shown by the decline in F1 scores. The effect is less severe than the triplet loss component; since we can consider it as multi-head graph attention, though.

The AUC score may occasionally somewhat improve if we swap from our topology-aware neighbor sampler to a uniform neighbor sampler. The F1 score, on the other hand, sharply declines, showing that the model's conclusions are likewise skewed and that its capacity to recognize good evaluations is reduced.

In summary, each of these components is essential to improving the performance and balance of our proposed approach. When we remove the triplet loss component, the F1 score, which measures a test's accuracy and recall, has dropped significantly. This implies that the outcomes of the model are strongly skewed in the absence of the triplet loss component. Therefore, the triplet loss component is crucial for preserving the model's predictions' balance and making sure they do not skew too far in the direction of false positives or false negatives.

Similarly, the absence of our pseudo-relation aggregation mechanism also results in a reduction in the model performance. This is evident from the decrease in F1 scores, which further indicates a reduction in the model's accuracy and recall.

doi:10.6342/NTU202304108

By lowering the bias in the model's predictions, the pseudo-relation aggregation method aids in improving prediction accuracy. Although the effects of removing this component are not as severe as those of removing the triplet loss component, they are nonetheless substantial and harmful to the model's overall performance.

Last but not least, we sometimes see a tiny improvement in the AUC score when we switch from our topology-aware neighbor sampler to a uniform neighbor sampler. A greater AUC implies better classifier performance. The AUC, or Area Under the Curve, is a performance indicator for the classification issue at various threshold values. The F1 score, however, drops significantly. This implies that the model's predictions are skewed in this situation as well, and its capacity to recognize benign reviews is diminished. As a result, the topology-aware neighbor sampler is essential to preserving the model's capacity to identify spam from benign reviews.

In general, the triplet loss, the pseudo-relation aggregation approach, and the topology-aware neighbor sampler all work together to enhance the performance and balance of our proposed approach. With all of these components in effect, we can provide a more balanced outcome, allowing us to distinguish between spam and benign reviews more successfully. This balance is essential to ensure that our model is accurate and trustworthy and that it doesn't unfairly penalize genuine reviews or fail to detect spam.

5.8 Performance Comparison on Amazon dataset.

in addition to the Yelp review dataset, we further use the Amazon reviews dataset of product reviews under the Musical Instruments category. [33]. We label reviews with more than 80% helpful votes as benign reviews and reviews with less than 20% helpful votes as spam reviews that are similar to previous work [34]. In the following experiments, we only use these labeled records to train and compare the performance of models. The total records we use are 36379 reviews, 85% of these reviews are labeled as benign, and 15% of these are labeled as spam.

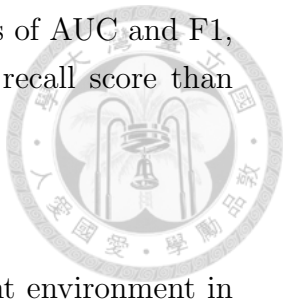
Table 12: Experiment results on Amazon review dataset.

	random split			time based split		
	AUC	F1 macro	Recall	AUC	F1 macro	Recall
proposed method	**0.7698**	**0.6450**	0.6099	**0.7078**	**0.6307**	0.4444
GAS	0.6948	0.5897	0.5222	0.6062	0.5663	0.3297
H2-FDector	0.6739	0.5223	**0.6493**	0.6497	0.5773	**0.6207**

The experiment results on Amazon are reported in table 12; we can observe

doi:10.6342/NTU202304108

that our proposed methods outperform baseline models in terms of AUC and F1, especially in the AUC scores. But our methods have a lower recall score than H^2-FDector.



5.9 Summary

In this chapter, we introduced how we set up our experiment environment in Section 5.1, then we introduced the evaluated metric we used to compare the model performance between our proposed method and baseline models that inspired us in Section 5.2. Then we compare the experiment results ten times to illustrate the model performance in Section 5.3. In Section 5.4, we dive into PRC and AUC-ROC to know the model performance under different thresholds setting. And we also observed the t-sne review embeddings to figure out how the model distinguishes the spam and benign review in Section 5.5. After that, we compare our model performance with different sentence embedding methods in Section 5.6. We also do the ablation study to show that the three components: triplet loss, the pseudo-relation aggregation approach, and the topology-aware neighbor sampler, are essential for our model in Section 5.7. Last but not least, we compare the model performance on the Amazon reviews dataset, which is also a popular dataset for spam review detection tasks.

doi:10.6342/NTU202304108

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this thesis, we directly construct a bipartite comment graph in which the review behavior can be naturally represented. Then we iterative sample sub-graph to perform stochastic training to generate the embedding that utilizes the social interaction between reviews.

We proposed a model that contains the topology-aware neighbors sampling to sample neighbors that can aggregate messages from more important neighbors and design a pseudo-relation heterogeneous graph attention network to extract the different patterns based on the predicted label to overcome the label imbalance of the dataset. Last, by not least, we carefully designed our loss function by employing focal loss to balance the influence of hard and easy samples and also adapt triplet loss to distinguish the benign and spam review. By these three components, we get a robust model compared to our baseline model and also have more generalized power than they.

In our work, we have concentrated on sampling neighbors based on the graph-topology structure that is purely generated from review behavior. We have not, however, added more details like star ratings and review counts. In the future, incorporating these more data points may allow us to improve our model. Utilizing data from reviews and star ratings allows us to capture a wider range of relations and interactions, which improves our comprehension and analysis of various interconnections.

doi:10.6342/NTU202304108

REFERENCES

[1] F. Shi, Y. Cao, Y. Shang, Y. Zhou, C. Zhou, and J. Wu, "H2-fdetector: a gnn-based fraud detector with homophilic and heterophilic connections," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1486–1494.

[2] J. Pitman. (2022) Local consumer review survey 2022. https://www.brightlocal.com/research/local-consumer-review-survey/?SSAID=314743&SSCID=81k6_t41ah.

[3] N. Hussain, H. Turab Mirza, G. Rasool, I. Hussain, and M. Kaleem, "Spam review detection techniques: A systematic literature review," *Applied Sciences*, vol. 9, no. 5, p. 987, 2019.

[4] S. K. Maurya, D. Singh, and A. K. Maurya, "Deceptive opinion spam detection approaches: a literature survey," *Applied intelligence*, vol. 53, no. 2, pp. 2189–2234, 2023.

[5] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, 2015, pp. 985–994.

[6] F. Abri, L. F. Gutierrez, A. S. Namin, K. S. Jones, and D. R. Sears, "Fake reviews detection through analysis of linguistic features," *arXiv preprint arXiv:2010.04260*, 2020.

[7] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*, 2020.

[8] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, "What yelp fake review filter might be doing?" in *Proceedings of the international AAAI conference on web and social media*, vol. 7, no. 1, 2013.

[9] N. Jindal and B. Liu, "Opinion spam and analysis," in *Proceedings of the 2008 international conference on web search and data mining*, 2008, pp. 219–230.

[10] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, "Shilling attacks against recommender systems: a comprehensive survey," *Artificial Intelligence Review*, vol. 42, no. 4, pp. 767–799, 2014.

[11] C. Yuan, W. Zhou, Q. Ma, S. Lv, J. Han, and S. Hu, "Learning review representations from user and product level information for spam detection," 2019.

[12] G. Wang, S. Xie, B. Liu, and S. Y. Philip, "Review graph based online store review spammer detection," in *2011 IEEE 11th international conference on data mining*. IEEE, 2011, pp. 1242–1247.

doi:10.6342/NTU202304108

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[14] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, p. 51, 2020.

[15] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[16] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He, "Pick and choose: A gnn-based imbalanced learning approach for fraud detection," in *Proceedings of the Web Conference 2021*, 2021, pp. 3168–3177.

[17] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li, "Spam review detection with graph convolutional networks," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2703–2711.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[19] A. Barushka and P. Hajek, "Review spam detection using word embeddings and deep neural networks," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2019, pp. 340–350.

[20] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. Online Available at: https://arxiv.org/abs/1908.10084

[21] S. Shehnepoor, R. Togneri, W. Liu, and M. Bennamoun, "HIN-RNN: A graph representation learning neural network for fraudster group detection with no handcrafted features," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2021. Online Available at: https://doi.org/10.1109%2Ftnnls.2021.3123876

[22] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[23] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," *arXiv preprint arXiv:1703.06103*, 2017.

[24] S.-j. Ji, Q. Zhang, J. Li, D. K. Chiu, S. Xu, L. Yi, and M. Gong, "A burst-based unsupervised method for detecting review spammer groups," *Information Sciences*, vol. 536, pp. 454–469, 2020.

doi:10.6342/NTU202304108

[25] Z. Wang, S. Gu, and X. Xu, "Gslda: Lda-based group spamming detection in product reviews," *Applied Intelligence*, vol. 48, pp. 3094–3107, 2018.

[26] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*, 2020.

[27] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[28] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[29] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[30] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. Online Available at: https://doi.org/10.1109%2Fcvpr.2015.7298682

[31] Y. Wu and Y. Liu, "Robust truncated hinge loss support vector machines," *Journal of the American Statistical Association*, vol. 102, no. 479, pp. 974–983, 2007.

[32] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[33] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[34] S. Zhang, H. Yin, T. Chen, Q. V. N. Hung, Z. Huang, and L. Cui, "Gcn-based user representation learning for unifying robust recommendation and fraudster detection," 2020.

doi:10.6342/NTU202304108