Spam Review Detection with Graph Convolutional Networks

Ao Li, Zhou Qin*
Runshi Liu, Yiqun Yang, Dong Li
Alibaba Group
Hangzhou, China
{jianzhen.la,qinzhou.qinzhou,runshi.lrs,yiqun.yyq}@alibaba-inc.com;shiping@taobao.com

ABSTRACT

Reviews on online shopping websites affect the buying decisions of customers, meanwhile, attract lots of spammers aiming at misleading buyers. Xianyu, the largest second-hand goods app in China, suffering from spam reviews. The anti-spam system of Xianyu faces two major challenges: scalability of the data and adversarial actions taken by spammers. In this paper, we present our technical solutions to address these challenges. We propose a large-scale anti-spam method based on graph convolutional networks (GCN) for detecting spam advertisements at Xianyu, named GCN-based Anti-Spam (GAS) model. In this model, a heterogeneous graph and a homogeneous graph are integrated to capture the local context and global context of a comment. Offline experiments show that the proposed method is superior to our baseline model in which the information of reviews, features of users and items being reviewed are utilized. Furthermore, we deploy our system to process million-scale data daily at Xianyu. The online performance also demonstrates the effectiveness of the proposed method.

CCS CONCEPTS

• **Information systems** → **Spam detection**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

spam detection, graph neural networks, heterogeneous graph

ACM Reference Format:
Ao Li, Zhou Qin and Runshi Liu, Yiqun Yang, Dong Li. 2019. Spam Review Detection with Graph Convolutional Networks. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357820

1 INTRODUCTION

Reviews of online shopping websites provide valuable information, such as product quality and aftersales service. These reviews, which straightforwardly influence purchase decisions of customers [17], have become a target place for spammers to publish malicious information. In our case, Xianyu, the largest second-hand goods app in China, which facilitates daily sales of over 200,000 products and achieves an annual Gross Merchandise Volume (GMV) over 13 billion dollars from August 2017 to July 2018, is also suffering from spam reviews. These spam reviews need to be cleaned out because they not only undermine experience of users but also provide a hotbed for internet fraud.

Figure 1: Xianyu App: the main page (left) and the comment area (right). The comment area highlighted with dashed rectangle provides the main communication tool of Xianyu.

Reviews at Xianyu differ from reviews in other e-commerce websites in several aspects. For instance, at Amazon or Taobao, reviews are usually made by customers who have bought the products, therefore review action usually happens *after* purchase. In contrast, users have no idea about the quality and possible lowest price of the second-hand goods. Thus reviews at Xianyu act as a communication tool for buyers and sellers (e.g., query for details and discounts) and review action usually happens *before* purchase, as shown in Figure 1 and Figure 2. So instead of *review* the term *comment* will be used in the rest of paper to underline the essential difference of spam types at Xianyu. Generally, there are two main kinds of spam comments at Xianyu: *vulgar comments* and *spam advertisements*. Given the fact that spam advertisements takes the majority of spam comments, we focus on spam advertisements detection in this work,

The main challenges of spam advertisements detection are:

- **Scalability**: Large-scale data of Xianyu with over 1 billion second-hand goods published by over 10 millions users.

*The first two authors contributed equally to this research

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357820

- **Adversarial Actions**: Same as most risk control system, the anti-spam system suffers from performance decay according to adversarial actions taken by spammers.

Spammers normally take the following two adversarial tricks to circumvent the anti-spam system:

- **Camouflage**:Using different expressions with similar meaning. For example, "Dial this number for a part-time job" and "Want to earn more money in your spare time? Contact me" are both spam advertisements for the purpose of guiding people to risky offline activities.
- **Deforming the comments**: Spammers replace some keywords in the comments with rarely used Chinese characters or typos deliberately. For example, "Add my vx", "Add my v" and "Add my wx" all mean "Add my WeChat[1] account".

These tricks bring some inconvenience but still understandable for human readers. On the contrary, a big challenge for the anti-spam system is to recognize various patterns designed by spammers.

At the same time, it's noticed that the impact of adversarial actions can be alleviated by introducing the context of comments. We define the context into two categories: local context and global context. The local context refers to the information from the commenter and the item related, while the global context refers to the information offered by the feature distribution of all comments.

In this work, we present a highly-scalable anti-spam method based on graph convolutional networks (GCN), dubbed GCN-based Anti-Spam method (GAS).

In summary, the contributions of the work are listed below:

(1) We propose a GCN-based heterogeneous graph spam detection algorithm which works on a bipartite graph with edge attributes at Xianyu. The designed model significantly outperforms the baseline model that can be easily generalized to a meta-path[22] based heterogeneous GCN algorithm for various heterogeneous graphs and applications.
(2) Besides the heterogeneous graph which utilizes local context of comments, we make use of global context and propose GAS, which further improves the result.
(3) We deploy the proposed anti-spam model with distributed Tensorflow framework to process million-scale comments daily at Xianyu. According to offline experiments and online evaluation, our system remarkably identifies much more spam comments and alleviates the impact of adversarial actions while satisfying the efficiency requirement.

The rest of the paper is organized as follows. Section 2 lists related works. In Section 3, we elaborate the proposed GAS model. Offline and online experiments are presented in Section 4. Then we introduce the implementation and deployment of the system at Xianyu in Section 5. The work is summarized in Section 6.

2 RELATED WORK

Most existing spam detection methods focus on extracting robust engineered features from review contents or reviewer behaviors. [7] collected review centric, reviewer centric and product centric features, and fed them to a logistic regression model. [13] summarized domain expert features for opinion mining, then a set of elaborate designed features are used for review classification task.

The first graph-based spam detection method was presented in [25]. They constructed the "review graph" with three types of nodes — reviewers, stores, and reviews. Then reinforced the reviewer trustiness, store reliability and honesty of review in a HITS[10] like way. Soliman[21] proposed a novel graph-based technique that detects spam using graph clustering on a constructed user similarity graph which encodes user behavioral patterns within its topology. The NetSpam framework[20] defined different meta-path types on review graph and used them in classification.

Recent years have witnessed a growing interest in developing deep-learning based algorithms on graph, including unsupervised methods[5, 12, 16] and supervised methods[6, 9, 11, 24]. One of the most prominent progress is known as GCN[9], in which the "graph convolution" operator is defined as feature aggregation of one-hop neighbors. Through iterative convolutions, information propagates multiple hops away in the graph. William et al.[6] proposed GraphSAGE, an inductive framework that leverages node sampling and feature aggregation techniques to efficiently generate node embeddings for unseen data. Graph Attention Networks (GAT)[24] further incorporates attention mechanism into GCN.

Most of the graph methods focus on the homogeneous graph, while in many real-world applications, data can be naturally represented as heterogeneous graphs. Heterogeneous graphs were seldom studied before and attract growing interests nowadays[18, 32]. EAGCN[19] focuses on the case where multiple types of links connecting nodes in a graph and calculates heterogeneous node embeddings using attention mechanism. Similarly, [14, 27] utilize attention mechanism to aggregate information from different nodes or different meta-paths.

Graph methods have been applied in many domains, e.g., recommendation system[4, 26, 28, 31, 33], chemical properties prediction[19], healthcare[2], malicious accounts detection[14] and so on. In this paper, a GCN-based method is first applied to the spam review detection problem, to the best of our knowledge.

3 PROPOSED METHOD

In this section, we first present preliminary contents of graph convolutional networks, then we illustrate the anti-spam problem at Xianyu. Finally, we will demonstrate our GAS method in two aspects: we first introduce how to extend GCN algorithm for heterogeneous graph and then improve GAS by further incorporating global context.

3.1 Preliminaries

Previous work[6, 9, 24] focus mainly on homogeneous graphs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a homogeneous graph with node $v \in \mathcal{V}$, edge $(v, v') \in \mathcal{E}$, node feature $x_v = h_v^0 \in \mathbb{R}^{d_0}$ for $v \in \mathcal{V}$ where d_0 denotes the feature dimension of the node. The hidden state of node v learned by the l-th layer of the model is denoted as $h_v^l \in \mathbb{R}^{d_l}$, d_l denotes the dimension of the hidden state at l-th layer.

The GCN-based methods follow a layer-wise propagation manner. In each propagation layer, all the nodes update simultaneously. As summarized in [29, 30], a propagation layer can be separated

[1]WeChat is the largest messaging and social media app in China.

into two sub-layers: aggregation and combination. In general, for a GCN with L layers, aggregation and combination sub-layers at l-th layer ($l = 1, 2, \cdots L$) can be written as:

$$h^l_{N(v)} = \sigma\left(W^l \cdot AGG(\{h^{l-1}_{v'}, \forall v' \in N(v)\})\right) \quad (1)$$

$$h^l_v = COMBINE\left(h^{l-1}_v, h^l_{N(v)}\right) \quad (2)$$

where $N(v)$ is a set of nodes adjacent to v, AGG is a function used for aggregate embeddings from neighbors of node v, this function can be customized by specific models, e.g., max-pooling, mean-pooling[6] or attention based weighted summation[24]. W^l is a trainable matrix shared among all nodes at layer l. σ is a non-linear activation function, e.g., Relu. $h^l_{N(v)}$ denotes the aggregated feature of node v's neighborhood at l-th layer. $COMBINE$ function is used to combine self embedding and the aggregated embeddings of neighbors, which is also a custom setup for different graph models, e.g., concatenation as in GraphSAGE[6].

In GCN[9] and GAT[24], there are no explicit combination sub-layers. Self information of v is introduced by replacing $N(v)$ with $\tilde{N}(v)$ in Eq.(1), where $\tilde{N}(v) = v \cup N(v)$. Hence $COMBINE$ step actually happens inside of AGG step.

3.2 Problem Setup

Our purpose is to identify spam comments at Xianyu, which can be formulated to an edge classification problem on a directed bipartite graph with attributed nodes and edges.

Comments on Xianyu can be naturally represented as a bipartite graph $G(U, I, E)$, where U is the set of user nodes (vertices), I is the set of item nodes (vertices) and E is the set of comments (edges). An edge $e \in E$ from a user $u \in U$ to an item $i \in I$ exists if u makes a comment e to i. Additionally, given a vertex $v \in I \cup U$, let $N(v)$ be the set of vertices in node v's one-hop neighbors, i.e. $N(v) = \{v' \in I \cup U | (v, v') \in E\}$. In the bipartite graph case of Xianyu, $N(i) \in U$ and $N(u) \in I$. $E(v)$ denotes the edges connected to v. Let $U(e)$ and $I(e)$ denote the user node and item node of edge e. This bipartite graph is named **Xianyu Graph**. See Figure 2 for a real word example.

3.3 Heterogeneous Graph Convolutional Networks on Xianyu Graph

As introduced in Section 3.1, in the GCN-based node classification task on a homogeneous graph, node embedding from the last layer is used as the input of a classifier.

Instead, we utilize the edge embedding from the last propagation layer together with embeddings of the two nodes this edge links to. We concatenate these three embeddings for the edge classification task as shown in Figure 3, where z_e, z_u and z_i denote the edge, user and item embedding, i.e., $z_e = h^L_e$, $z_u = h^L_{U(e)}$ and $z_i = h^L_{I(i)}$.

We will concretely demonstrate how to tailor the standard GCN for a bipartite graph with edge attributes. The keypoint is to customize aggregation sub-layer and combination sub-layer in Eq.(1) and Eq.(2).

3.3.1 Aggregation Sub-layer. Aggregation sub-layer in GCN treats all kinds of nodes the same and ignores the edge attributes. To fit

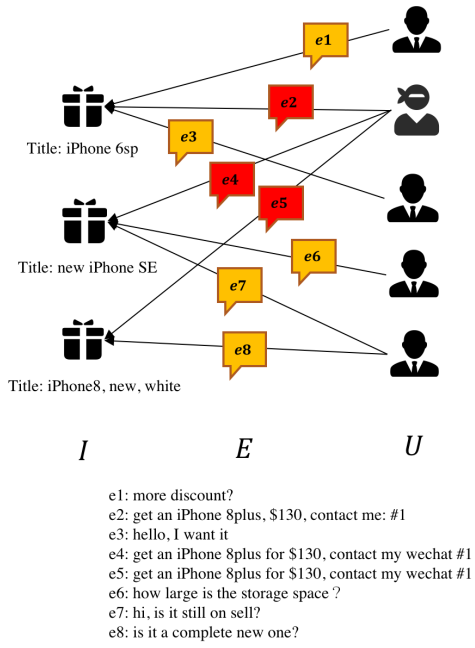


Figure 2: A miniature of Xianyu Graph. In this setting, spammer wants to mislead buyer to offline transactions, so he post an eye-catching comment saying that he has a cheaper phone for sale under many related items. I, E, U represents item nodes, comments, user nodes respectively. Here #1 stands for a specific WeChat account ID.

the general framework Eq.(1) to Xianyu Graph, three aggregation functions for each kind of entities (user, item, comment) are defined.

For a comment, i.e., an edge, the hidden state is updated as the concatenation of previous hidden states of the edge itself and two nodes it links to. So the aggregation sub-layer is defined as

$$h^l_e = \sigma\left(W^l_E \cdot AGG^l_E(h^{l-1}_e, h^{l-1}_{U(e)}, h^{l-1}_{I(e)})\right) \quad (3)$$

where

$$AGG^l_E\left(h^{l-1}_e, h^{l-1}_{U(e)}, h^{l-1}_{I(e)}\right) = concat\left(h^{l-1}_e, h^{l-1}_{U(e)}, h^{l-1}_{I(e)}\right) \quad (4)$$

For the user node $u \in U$ and item node $i \in I$, besides the information from neighbor nodes, the attributes of edges connected to them are also collected. The aggregated neighbor embedding $h^l_{N(u)}$ and $h^l_{N(i)}$ are calculated as

$$\begin{aligned} h^l_{N(u)} &= \sigma\left(W^l_U \cdot AGG^l_U(\mathcal{H}^{l-1}_{IE})\right) \\ h^l_{N(i)} &= \sigma\left(W^l_I \cdot AGG^l_I(\mathcal{H}^{l-1}_{UE})\right) \end{aligned} \quad (5)$$

where

$$\begin{aligned} \mathcal{H}^{l-1}_{IE} &= \left\{concat(h^{l-1}_i, h^{l-1}_e), \forall e = (u, i) \in E(u)\right\} \\ \mathcal{H}^{l-1}_{UE} &= \left\{concat(h^{l-1}_u, h^{l-1}_e), \forall e = (u, i) \in E(i)\right\} \end{aligned} \quad (6)$$

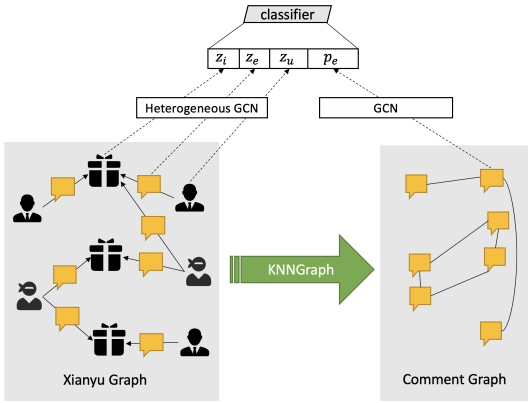


Figure 3: An illustration of GAS which incorporates two graph models. Heterogeneous GCN which works on the Xianyu Graph provides user, item, comment embeddings z_u, z_i, z_e, respectively. GCN which works on the homogeneous Comment Graph provides comment embedding p_e. In GAS, these embeddings are concatenated together as the input of the classifier: $y = classifier(concat(z_i, z_u, z_e, p_e))$.

The two kinds of nodes maintain different parameters (W_U^l, W_I^l) and different aggregation functions (AGG_U^l, AGG_I^l).

As for the specific forms of AGG_U^l and AGG_I^l, we adapt the attention mechanism:

$$\begin{aligned} AGG_U^l(\mathcal{H}_{IE}^{l-1}) &= ATTN_U\left(h_u^{l-1}, \mathcal{H}_{IE}^{l-1}\right) \\ AGG_I^l(\mathcal{H}_{UE}^{l-1}) &= ATTN_I\left(h_i^{l-1}, \mathcal{H}_{UE}^{l-1}\right) \end{aligned} \tag{7}$$

$ATTN$ here is a function $f : h_{key} \times \mathcal{H}_{val} \rightarrow h_{val}$ which maps a feature vector h_{key} and the set of candidates' feature vectors $\mathcal{H}_{val}$ to an weighted sum of elements in $\mathcal{H}_{val}$. The weights of the summation, i.e. attention values are calculated by the scaled dot-product attention [23].

3.3.2 Combination Sub-layer. After aggregating the neighbors' information, we follow a combination strategy in [6] for the user and item nodes as

$$\begin{aligned} h_u^l &= concat\left(V_U^l \cdot h_u^{l-1}, h_{N(u)}^l\right) \\ h_i^l &= concat\left(V_I^l \cdot h_i^{l-1}, h_{N(i)}^l\right) \end{aligned} \tag{8}$$

Where V_U^l and V_I^l denote trainable weight matrices for user node and item node, the h_u^l and h_i^l are the user hidden state and item hidden state of l-th layer.

The forward propagation process of our algorithm is described in Algorithm 1. Note that this method can actually be generalized to a meta-path based heterogeneous GCN algorithm for various heterogeneous graphs with edge attributes. In detail, for a meta-path P in the form of $A^0 \xrightarrow{R^0} A^1 \cdots A^{l-1} \xrightarrow{R^{l-1}} A^l \xrightarrow{R^l} \cdots \xrightarrow{R^{L-1}} A^L$, where A^l and R^l corresponds to the node type and edge type on P. For a node v of type A^l at l-th layer on P, the aggregation and combination process can be written as:

$$\begin{aligned} \mathcal{H}_{A^{l-1}E^{l-1}}^{l-1} &= \left\{concat\left(h_{v'}^{l-1}, h_e^{l-1}\right) \quad \forall e \in (v, v') \in E_{R^{l-1}}(v)\right\} \\ h_{N_{A^{l-1}}(v)}^l &= \sigma\left(W_{A^{l-1}\rightarrow A^l}^l \cdot AGG_{A^{l-1}\rightarrow A^l}^l\left(\mathcal{H}_{A^{l-1}E^{l-1}}^{l-1}\right)\right) \\ h_v^l &= concat\left(V_{A^l}^l \cdot h_v^{l-1}, h_{N_{A^{l-1}}(v)}^l\right) \end{aligned} \tag{9}$$

where $E_{R^{l-1}}(v)$ denotes edges link to v with edge type R^{l-1} and $N_{A^{l-1}}(v)$ denotes neighbor nodes of v with node type A^{l-1}. In our case and a 2-layer setting, two implicit meta-paths are $U \xrightarrow{E} I \xrightarrow{E} U$ and $I \xrightarrow{E} U \xrightarrow{E} I$.

3.3.3 Incorporate Graph Networks with Text Classification Model. Comments should be transformed into embeddings before merging with user and item features. TextCNN[8] is a satisfactory text classification model that balance effectiveness and efficiency. Therefore we employ the TextCNN model to get comment embeddings. TextCNN is integrated into our GCN model as an end-to-end classification framework.

Specifically, we use word embedding pre-trained on a million-scale comment dataset by word2vec[15] as the input of TextCNN. The output of TextCNN is then used as the embedding of the comment.

$$h_e^0 = \text{TextCNN}(w_0, w_1, w_2, \cdots, w_n) \tag{10}$$

where w_i represents word embedding of i-th word of comment e, and h_e^0 is the initial embedding of comment e in Eq.(3). Therefore the parameters of TextCNN are trained together with others in the model described in Section 3.3.

3.4 GCN-based Anti-Spam model

Spam comments at Xianyu are often deformed by malicious spammers as a countermove to our spam detection system. For example, deliberate typos and abbreviations are used to circumvent our detection. These spams have minor impact for human to read but often confuse our NLP model. Especially when spams are posted by different users and under different items. Obviously, the local context can not help in this situation. For this kind of spams, we want to find some supplementary information from the whole graph like "How many similar comments on the whole site and what do they mean?".

It's noticed that even if increasing the number of propagation layers help nodes capture the global context, the noise introduced can not be ignored, as other researchers had reported[9, 11]. Experiments in Section 4.1 show that the performance can hardly benefit from increasing the number of propagation layers in our case.

Therefore, we takes a shortcut way to capture global context of nodes. More specifically, we construct a homogeneous graph named **Comment Graph** by connecting comments with similar contents. In this way, the edges(comments) in the heterogeneous Xianyu Graph now become vertices in Comment Graph. See Figure 4 for a real world case of Comment Graph.

As demonstrated in [11], GCNs on homogeneous graph can be viewed as a special form of Laplacian smoothing. Node classification tasks can benefit from this theory based on the assumption that nodes with same label are often grouped together in the graph, the features of nodes can be smoothed by its neighbors, thus make the

Algorithm 1: The forward propagation process of Heterogeneous Graph Convolutional Networks on Xianyu Graph.

Input: Set of edges $E_b \subset E$, number of layers L, functions $U(E_b)$ and $I(E_b)$ which map E_b to the user nodes and the item nodes E_b linked, respectively. Xianyu Graph $G(U, I, E)$

Output: Hidden states of the L-th layer, include the hidden states of edges: $z_e, \forall e \in E_b$, the hidden states of users: $z_u, \forall u \in U(E_b)$ and the hidden states of items: $z_i, \forall i \in I(E_b)$

. **begin**

$E^l \leftarrow E_b$;

$U^l \leftarrow U(E_b)$;

$I^l \leftarrow I(E_b)$;

// Sampling ;

for $l = L, \cdots, 1$ **do**

$U^{l-1} \leftarrow U^l$;

$I^{l-1} \leftarrow I^l$;

for $u \in U^l$ **do**

$U^{l-1} \leftarrow U^{l-1} \cup SAMPLING(N(u))$;

end

for $i \in I^l$ **do**

$I^{l-1} \leftarrow I^{l-1} \cup SAMPLING(N(i))$;

end

end

// Go through neural network ;

for $l = 1, \cdots, L$ **do**

for $e \in E^l$ **do**

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

end

for $u \in U^l$ **do**

$\mathcal{H}_{IE}^{l-1} \leftarrow \left\{concat(h_i^{l-1}, h_e^{l-1}), \forall e = (u, i) \in E(u)\right\}$;

$h_{N(u)}^l \leftarrow \sigma\left(W_U^l \cdot AGG_U^l(\mathcal{H}_{IE}^{l-1})\right)$;

$h_u^l = concat\left(V_U^l \cdot h_u^{l-1}, h_{N(u)}^l\right)$;

end

for $i \in I^l$ **do**

$\mathcal{H}_{UE}^{l-1} \leftarrow \left\{concat(h_u^{l-1}, h_e^{l-1}), \forall e = (u, i) \in E(i)\right\}$;

$h_{N(i)}^l \leftarrow \sigma\left(W_I^l \cdot AGG_I^l(\mathcal{H}_{UE}^{l-1})\right)$;

$h_i^l = concat\left(V_I^l \cdot h_i^{l-1}, h_{N(i)}^l\right)$;

end

end

for $e \in E_b$ **do**

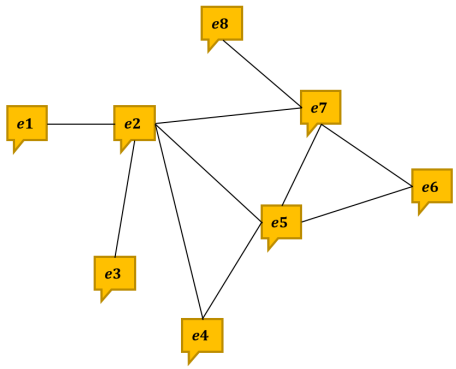
$z_e = h_e^L$;

$z_u = h_{U(e)}^L$;

$z_i = h_{I(e)}^L$;

end

end



e1: for more certified products, please add #2
e2: for more styles, please add #3
e3: need more styles and discounts, please add v #4
e4: more styles please add wecha #5
e5: people who like it please add my wechat, different styles
e6: if you like it, leave me a message, add v
e7: I have others styles, if you like it please add: #6
e8: anything you like? please add #5

Figure 4: A miniature of Comment Graph, in which "wechat", "wecha" and "v" all mean WeChat account and #2, #3, #4, #5, #6 stand for different account IDs, all the comments in this subgraph are spam advertisements.

classification task easier. Therefore an inductive GCN algorithm[6] is performed on the constructed homogeneous Comment Graph to learn the comment embedding.

By incorporating the inductive GCN which works on Comment Graph with the heterogeneous GCN which works on Xianyu Graph, we propose an algorithm called GCN-based Anti-Spam(GAS) to train the model in an end-to-end manner. See Figure 3 for the whole structure of GAS.

Comment embedding of $e \in E$ learned by GCN from the Comment Graph is denoted as p_e. The final embedding of GAS is the concatenation of p_e and other embeddings learned from Xianyu Graph,

$$y = classifier(concat(z_i, z_u, z_e, p_e)), \tag{11}$$

where z_e, z_u and z_i denote the embeddings of e, $U(e)$ and $I(e)$ learned by the proposed heterogeneous GCN model, respectively.

A non-trivial problem needed to be discussed is how to generate Comment Graph, namely, how to identify similar comments. This can be naively done by scanning all the comments, finding the closest k peers of each. However, it is impractical for the $O(|E|^2)$ time complexity. In practice, we use approximate KNN Graph algorithm[3] to construct a graph based on K nearest neighbor of nodes.

In detail, the Comment Graph is constructed as follows:

- Remove all the duplicated comments to avoid the trivial solution, i.e., two comments with same content are always most similar for each other.
- Generate comments embeddings by the method described in [1].

- Obtain the similar comment pairs by employing the approximate KNN Graph algorithm.
- Remove comment pairs posted by same user or posted under same item, since the local context has been taken into consideration on Xianyu Graph.

We assume in this way various spam reviews can be smoothed by integrating features of their neighbors. We also provide an visualization of a subset of training samples is provided to show that comments are more separable after the smooth process, see Figure 5 for details.

Figure 5: Sample space visualization. Green points represent non-spam comments, and red points represent spam comments. Left: the original comment embeddings projected into 2-D space by PCA directly; Right: the smoothed comment embeddings (by averaging features of self and neighbors) projected into 2-D space by PCA.

A quantitative analysis is also conducted to prove that comments are more separable after the smooth process. Two logistic regression models are trained and tested on raw embeddings and smoothed embeddings in Figure 5. AUC and F1-score are reported in Table 1. These results demonstrate that smoothed embeddings are more linear-separable, which further indicates that a classifier built on smoothed embedding will perform better.

Table 1: Quantitative analysis of the effect of embedding smoothing.

Model feature	AUC	F1 score
Raw embedding	0.9342	0.8332
Smoothed embedding	0.9373	0.8448

4 EXPERIMENTS

In this section, we evaluate the performance of proposed method on true Xianyu dataset. First, we compare our method with several models using the offline dataset, then the online performance on Xianyu app is reported. Last, we present some real-world cases to give insight into the proposed methods.

4.1 Offline Evaluation

4.1.1 Dataset. To evaluate the proposed method at Xianyu, we construct a heterogeneous graph with overall 37,323,039 comments published by 9,158,512 users on 25,107,228 items during a time period. We collect 1,725,438 regular comments along with 74,213 spam comments marked by human experts.

Training, validation and test set are randomly split with a ratio of 6:1:3.

4.1.2 Comparing Methods. To compare our method with traditional review mining method, we follow the instructions in [13]. Specifically, we design lots of hand-made features for comment, item and user(e.g. comment length, whether it is the first comment of an item, whether it is the only comment of an item, cosine similarity between the comment and item features, item price, number of comments made by the user, etc.). To encode the comment content, we calculate mutual information of each word. Top 200 words with largest mutual information values are selected out and then used to construct a binary value vector for each comment. Each entity of this vector indicate whether a word occurs or not. Lastly, we concatenate these features as the input of a GBDT model. We call this model GBDT as an abbreviation.

Instead of the binary value encoded by mutual information, a TextCNN model is also used to extract comment embedding. The comment embedding is then concatenated with user and item features described above as the input of a 2-layer MLP(Multilayer Perceptron) model. This was the model deployed online at Xianyu, thus it's regarded as the baseline model, named TextCNN+MLP.

To demonstrate the effectiveness of global context introduced by Comment Graph, we also compare the the model which only utilize the local context Xianyu Graph as in Section 3.3. We call this model GAS-local.

In summary, the experiment configurations are detailed below:

- GBDT: Domain expert features with GBDT model.
- TextCNN+MLP(**baseline**): TextCNN + user features + item features with 2 layer MLP model.
- GAS-local-1: 1 propagation layer on Xianyu Graph(i.e., 1-hop neighbors).
- GAS-local-2: 2 propagation layers on Xianyu Graph.
- GAS: GAS model with 2 propagation layers on Xianyu Graph and 1 propagation layer on Comment Graph.

The GBDT model is trained using 100 trees with a learning rate of 0.05. For other models, the TextCNN structure used by all the methods share the same hyperparameters, e.g., filter sizes are set to $\{3, 4, 5\}$ with 128 filters for each. All the methods except for GBDT are trained for 8 epochs. The max number of sampling M is 16 for Xianyu Graph and the max number of sampling for Comment Graph is set to 64. The TextCNN+MLP model is trained with a standalone Tensorflow program and the learning rate is 0.001, while the proposed methods are all trained in a distributed manner with 8 GPUs. The learning rate is set to 0.005 and the batch size is set to 128.

4.1.3 Result Analysis. We evaluate these methods in terms of *AUC*, *F1-score* and *recall at 90% precision*. The metric *recall rate at 90% precision* is chosen since the detected spam reviews will be disposed immediately in practice. We must ensure high model precision to avoid disturbing normal users. so recall at 90% precision becomes an essential criteria to compare different models.

The results are shown in Table 2 and the PR curves are shown in Figure 6. We can see that GAS-local-1, GAS-local-2 and GAS outperform GBDT and TextCNN+MLP model in our dataset. This demonstrates the superiority of the proposed methods. The comparison of GAS-local-1 and TextCNN+MLP shows that the performance gain is significant attributed to the introduction of the local context. The *recall%at%90%precision* is improved from 54.86% to 66.90%,

Table 2: Result comparison of offline experiments in terms of *AUC*, *F1-score* and *recall at 90% precision* which is denoted as *recall@90% precision*.

method	AUC	F1 score	recall at 90% precision
GBDT	0.9649	0.7686	50.55%
TextCNN+MLP	0.9750	0.7784	54.86%
GAS-local-1	0.9806	0.8138	66.90%
GAS-local-2	0.9860	0.8143	67.02%
GAS	0.9872	0.8217	71.02%

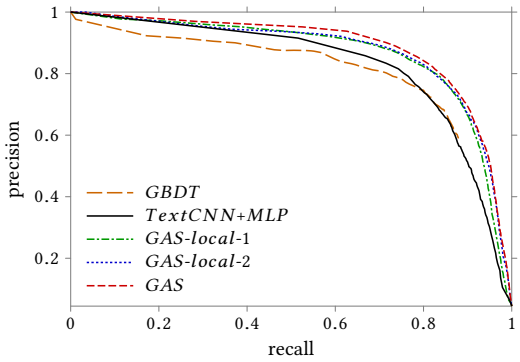


Figure 6: P-R chart of the offline evaluation. GAS, GAS-local-1 and GAS-local-2 significantly perform better and GAS gains further improvement compared to GAS-local-2.

which means 12.04% more spams can be detected and removed from the app. When comparing GAS-local-2 with GAS-local-1, the improvement is not prominent, which indicates the performance lift by incorporating 1-hop local context dominates. It may be due to the fact that besides the information introduced by 2-hop local context, noise is also introduced. As many authors had reported[9, 11], the performance gain attenuates dramatically as the hop increases. When comparing GAS and GAS-local-2, we can see a further improvement which demonstrates the effectiveness of incorporating of the global context. When precision is fixed to 90%, compared to GAS-local-2, we detect extra 4% spam comments.

Overall, our proposed method outperforms the deployed baseline system with a 4.33% F1-score lift. Crucially, under the fixed disposal threshold of 90% accuracy, our method brings an extra recall of 16.16% spam reviews.

4.2 Online Performance

We conduct online experiments on our platform as introduced in Section 5. The goal of the experiment is to compare the number of spam comments detected by different models at 90% precision which is checked by human experts. Detected spam comments will be removed from Xianyu app.

We deploy TextCNN+MLP, GAS-local-1 and GAS in our daily production environment and compare their performance. As depicted in Figure 7, the GAS-local-1 and GAS outperform TextCNN+MLP consistently in terms of the amount of detected spam comments. On the other hand, GAS outperforms GAS-local-1 consistently which further demonstrates that the system benefits from the global context introduced by the Comment Graph.

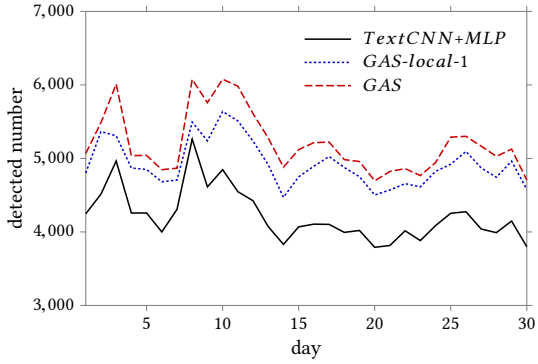


Figure 7: Online evaluation result. The proposed methods consistently outperforms the GBDT and TextCNN+MLP model at Xianyu.

4.3 Case Study

The spam comments detected by different methods are manually checked.

4.3.1 GAS-local-1 vs. TextCNN+MLP. We first compare the false negative samples of TextCNN+MLP (denoted as $TextCNN{+}MLP_{FN}$) and the samples in $TextCNN{+}MLP_{FN}$ recalled by GAS-local-1 (denoted as $TextCNN{+}MLP_{FN} \cap GAS\text{-}local\text{-}1_{TP}$. The result is shown in Table 3.

Table 3: Comparison of TextCNN+MLP and GAS-local-1. Obviously, the samples recalled by GAS-local-1 from $TextCNN{+}MLP_{FN}$ have more "spam neighbors" in local context.

samples	average #spams within 1 hop on Xianyu Graph
$TextCNN{+}MLP_{FN}$	2.60
$TextCNN{+}MLP_{FN} \cap GAS\text{-}local\text{-}1_{TP}$	3.24

We analyze extra spam samples covered by GAS-local-1, finding that they are mostly similar advertisements published by the same people or under the same item. For instance, a typical spam comment is "check my profile photo for surprises", in which the spam information is hidden in the profile photo. These advertisements contains no specific keywords so they can not recognized by TextCNN+MLP. But GAS-local-1 correctly detects these advertisements by associating the comment with other comments published by this user.

4.3.2 GAS vs. GAS-local-1. Likewise, we compare the false negative samples of GAS-local-1 (denoted as $GAS\text{-}local\text{-}1_{FN}$) and the samples in $GAS\text{-}local\text{-}1_{FN}$ recalled by GAS (denoted as $GAS\text{-}local\text{-}1_{FN} \cap GAS_{TP}$. The result is shown in Table 4.

Table 4: Comparison of GAS and GAS-local-1. Obviously, the samples recalled by GAS from *GAS-local-1*$_{FN}$ have more "spam neighbors" in global context.

samples	average #spams within 1 hop on Xianyu Graph	average #spams within 1 hop on Comment Graph
GAS-local-1$_{FN}$	2.23	17.23
GAS-local-1$_{FN} \cap GAS_{TP}$	3.53	36.68

In detail, we analyze the results and find that two kinds of spam comments are more favoured by GAS compared to GAS-local-1:

- **adversarial advertisements published by spammers** This kind of spam advertisements are deformed by spammers with similar meaning (see Table 5 for a typical example). Most of these spam reviews are not published by the same account or under the same item, but they are connected to each other on the Comment Graph. GAS capture them by taking advantage of the Comment Graph that introduces the global context. These spams may published by a group of spammers, they may collect many accounts and publish several advertisements using each account.
- **coupon messages published by different users** This kind of coupon messages aim to lead people to another app. Once someone use the invitation code in the message, the publisher of the comment will be paid. Unlike the normal spammer that publishes a lot of similar comments under different items, this kind of spam comments is published by many different people, which may not be malicious spammers. But these coupon messages actually disturb other customers. This kind of publisher does not publish too much reviews as malicious spammers do, and will not gather under a particular item. That make it hard to recognize with only local context through Xianyu Graph. By introducing Comment Graph, this kind of comments group together, then recognized by GAS.

Table 5: Examples of extra spams detected by GAS compared to GAS-local, where "+V xxxxxx" and "Vx:xxxxxx" means WeChat app account.

user_id	item_id	comment
8737661	12381953	This is the bonus I got at Taobao, contact me and I will teach you how to do that
8737661	26771502	Get Bonus from Taobao, I can teach you
420310	27063522	Contact me to to learn to get the bonus from Taobao
653613	20374180	Teach you to get bonus: +V xxxxxx
8806574	20634558	Vx:xxxxxx to teach you to get bonus

5 SYSTEM IMPLEMENTATION AND DEPLOYMENT

In this section, we introduce the implementation and deployment of the proposed GCN-based anti-spam method at Xianyu. We first give an overview of the system and then elaborate on the modules relevant to our method, especially the distributed Tensorflow model.

5.1 System Overview

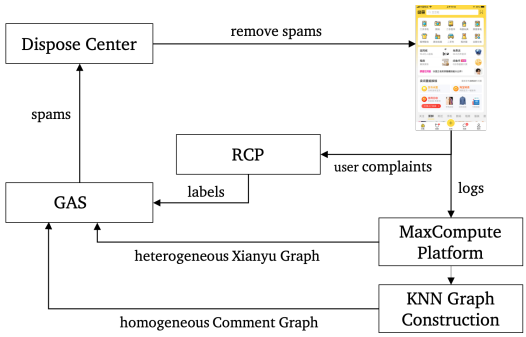


Figure 8: System Overview.

In Figure 8, we show the architecture of the anti-spam platform at Xianyu. The workflow is illustrated in the following:

- When a user comments on Xianyu App, the logs will be stored on the MaxCompute platform, which is a data processing platform for large-scale data warehousing. In practice, we choose the logs in the recent month to construct the heterogeneous graph per day.
- Based on the logs, the KNN Graph is constructed daily on the MaxCompute platform.
- A distributed Tensorflow implementation of GAS is employed to detect spams.
- The detected spams are removed from the app and the malicious accounts may be disabled.
- RCP (Risk Check Platform) is used to check the complaints from users being punished. The complaints reviewed and supported by human experts will lead to punishment withdrawal and be treated as mistakes of the model. The result of RCP will be used as labeled samples for further optimization of our model.

5.2 Implementation

We conduct a distributed implementation of the proposed method. Considering the large scale data in Xianyu, i.e., billions of items, millions of users and billions of comments, the parameter-server architecture of Tensorflow is adopted to provide a distributed solution for storage, data-fetching, training and predicting. Specifically, we use 8 parameter servers along with 8 workers, each worker is equipped with an Nvidia V100 GPU card, 6 CPU cores, 32GB memory. Parameter server has 2 CPU cores with 300 GB memory each.

5.2.1 storage. First, the graph data must be stored and readily available when the model needs to fetch the data from the graph. The Xianyu Graph is enormous and thus impractical to be saved in one machine, so the graph structure as well as the features of vertices and edges are saved in parameter servers. Note that graph structure is stored as adjacency list for memory efficiency.

A time-consuming step is data loading, which is both CPU-bound (parsing) and I/O-bound (fetching). To accelerate the loading process, we split the adjacency list and feature matrices into several parts, evenly scattered them to parameter servers to perform a distributed loading task. Specifically, each worker is responsible for loading a particular part of the table to fill the corresponding sub-matrix in parameter server. By this means, a linear acceleration of $O(\#workers)$ is reached for loading data.

5.2.2 data fetching. During the computation period, workers will look up for necessary information on parameter server first, fetch them, then perform computations locally. Parameter server and workers are connected through the network but network throughput is far slower than the memory access. In our experiment, on each worker, there is about 41% time wasted on fetching information from the server. To accelerate the lookup phase, we use cache mechanism on each worker, i.e., each worker will cache the features and adjacency lists locally when performing a search on parameter server. Cache technique save about 30% training time.

Finally, the training time of the offline experiment described in Section 4.1 is reduced to 2 hours for GAS.

6 CONCLUSION

The spam detection problem at Xianyu faces two main challeges: scalability and adversarial actions. To address these two challeges, we proposed an end-to-end GCN-based Anti Spam(GAS) algorithm which incorporates the local context and the global context of comments. The offline evaluation and online performance demonstrate the effectiveness of our method at Xianyu. Real-world cases are studied to further prove the effect of the different context introduced which alleviates the impact of adversarial actions. Finally, we elaborate on the implementation, deployment and workflow of the proposed method at Xianyu.

7 ACKNOWLEDGEMENTS

We would like to thank Yuhong Li, Jun Zhu, Leishi Xu for their assistance on KNN Graph algorithm, and thanks to Huan Zhao for reviews and discussions. We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

REFERENCES

[1] S. Arora, Y. Liang, and T. Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *ICLR*.
[2] E. Choi, M.T. Bahadori, L. Song, W.F. Stewart, and J. Sun. 2017. GRAM: graph-based attention model for healthcare representation learning. In *SIGKDD*. 787–795.
[3] W. Dong, C. Moses, and K. Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW*. 577–586.
[4] M. Grbovic and H. Cheng. 2018. Real-time Personalization using Embeddings for Search Ranking at Airbnb. In *SIGKDD*. 311–320.
[5] A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. 855–864.
[6] W. Hamilton, R. Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[7] N. Jindal and B. Liu. 2008. Opinion spam and analysis. In *WSDM*. 219–230.
[8] Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. Association for Computational Linguistics, 1746–1751.
[9] T.N Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[10] J.M. Kleinberg. 1999. Hubs, authorities, and communities. *ACM computing surveys (CSUR)* 31, 4es (1999), 5.
[11] Q. Li, Z. Han, and X.M. Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. 3538–3545.
[12] L. Liao, X. He, H. Zhang, and T.S. Chua. 2018. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 30, 12 (2018), 2257–2270.
[13] B. Liu and L. Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*. Springer, 415–463.
[14] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song. 2018. Heterogeneous Graph Neural Networks for Malicious Account Detection. In *CIKM*. 2077–2085.
[15] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.
[16] B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. 701–710.
[17] A.K. Samha, Y. Li, and J. Zhang. 2014. Aspect-based opinion extraction from customer reviews. *arXiv preprint arXiv:1404.1982* (2014).
[18] M. Schlichtkrull, T.N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. 593–607.
[19] C. Shang, Q. Liu, K.S. Chen, J. Sun, J. Lu, J. Yi, and J. Bi. 2018. Edge Attention-based Multi-Relational Graph Convolutional Networks. *arXiv preprint arXiv:1802.04944* (2018).
[20] S. Shehnepoor, M. Salehi, R. Farahbakhsh, and N. Crespi. 2017. NetSpam: a network-based spam detection framework for reviews in online social media. *IEEE Transactions on Information Forensics and Security (TIFS)* 12, 7 (2017), 1585–1595.
[21] A. Soliman and S. Girdzijauskas. 2017. Adaptive graph-based algorithms for spam detection in social networks. In *NETYS*. 338–354.
[22] Y. Sun and J. Han. 2012. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery* 3, 2 (2012), 1–159.
[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *NeurIPS*. 5998–6008.
[24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. 2018. Graph Attention Networks. In *ICLR*.
[25] G. Wang, S. Xie, B. Liu, and Philip S.Y. Yu. 2012. Identify online store review spammers via social review graph. *ACM Transactions on Intelligent Systems and Technology (TIST)* (2012).
[26] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D.L. Lee. 2018. Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba. In *SIGKDD*. 839–848.
[27] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P.S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.
[28] W. Xiao, H. Zhao, H. Pan, Y. Song, V. W. Zheng, and Q. Yang. 2019. Beyond Personalization: Social Content Recommendation for Creator Equality and Consumer Satisfaction. In *SIGKDD*. 235–245.
[29] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. 2018. How Powerful are Graph Neural Networks?. In *ICLR*.
[30] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*. 5449–5458.
[31] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, and J. Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *SIGKDD*. 974–983.
[32] Z. Zhang, P. Cui, and W. Zhu. 2018. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202* (2018).
[33] H. Zhao, Q. Yao, J. Li, Y. Song, and D.L. Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *SIGKDD*. 635–644.