Category-aware Graph Neural Networks for Improving E-commerce Review Helpfulness Prediction

Xiaoru Qu [1], Zhao Li [2*], Jialin Wang [1], Zhipeng Zhang [1], Pengcheng Zou [2], Junxiao Jiang [2], Jiaming Huang [2], Rong Xiao [2], Ji Zhang [3], Jun Gao [1*]
[1]Key Laboratory of High Confidence Software Technologies, Department of Computer Science, Peking University
[2]Alibaba Group, Hangzhou, China [3]Zhejiang Lab *Corresponding Authors
{quxiaoru,wangjialin,zhangzhipeng,gaojun}@pku.edu.cn,{lizhao.lz,xuanwei.zpc,junxiao.jjx,jimmy.hjm}@alibaba-inc.com,xiaorong.xr@taobao.com,zhangji77@gmail.com

ABSTRACT

Helpful reviews in e-commerce sites can help customers acquire detailed information about a certain item, thus affecting customers' buying decisions. Predicting review helpfulness automatically in *Taobao* is an essential but challenging task for two reasons: (1) whether a review is helpful not only relies on its text, but also is related with the corresponding item and the user who posts the review, (2) the criteria of classifying review helpfulness under different items are not the same. To handle these two challenges, we propose CA-GNN (***C**ategory-**A**ware **G**raph **N**eural **N**etworks*), which uses graph neural networks (GNNs) to identify helpful reviews in a *multi-task manner* — we employ GNNs with one shared and many item-specific graph convolutions to learn the common features and each item's specific criterion for classifying reviews simultaneously. To reduce the number of parameters in CA-GNN and further boost its performance, we partition the items into several clusters according to their category information, such that items in one cluster share a common graph convolution.We conduct solid experiments on two public datasets and demonstrate that CA-GNN outperforms existing methods by up to 10.9% in AUC. We also deployed our system in Taobao with online A/B Test and verify that CA-GNN still outperforms the baseline system in most cases.

CCS CONCEPTS

• **Information systems** → **Data mining**; **Electronic commerce**; • **Computing methodologies** → *Supervised learning*; • **Applied computing** → *Document management and text processing*.

KEYWORDS

Review Helpfulness Prediction; Graph Neural Networks; E-Commerce

ACM Reference Format:
Xiaoru Qu [1], Zhao Li [2*], Jialin Wang [1], Zhipeng Zhang [1], Pengcheng Zou [2], Junxiao Jiang [2], Jiaming Huang [2], Rong Xiao [2], Ji Zhang [3], Jun Gao [1*]. 2020. Category-aware Graph Neural Networks for Improving E-commerce Review Helpfulness Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412691

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412691

Types of Unhelpful Reviews	Example
1. Too General	"好"// "Good."
2. Not a helpful review of any items	"Ma armastan sindI love youÉg elska þigIs breá liom tú愛してるSeni seviyorumEs tevi miluJag älskar digЯ какаю цябеAmo il vostroEk is lief vir jouIch liebe dichKocham cię 我爱你"// "Love" in different languages
	"人气top周杰伦❗💎💖✨🌈 实力歌手周杰伦❗💎💖✨🌈 音乐鬼才周杰伦❗💎💖✨" // "The most popular singer Jay Chou ❗💎💖✨🌈 The talented singer Jay Chou❗💎💖✨🌈 The music wizard Jay Chou❗💎💖✨"
3. Not a helpful review of the current item	加油卡下面的评论："这款钢化膜非常好用，很贴合手机屏幕，很容易贴，质量也很好，保护手机的效果很强"// Review on a fuel card: "This screen protector is very useful. The protector and my smartphone screen are a good fit. It can be easily put on. It is of high quality, and it can protect the smartphone well."

Figure 1: Real-world examples of unhelpful reviews for different items from Taobao. The first two types (i.e., *Type 1* and *Type 2*) of reviews are often inaccurate and did not describe a certain item. The last one (*Type 3*) describes a certain item but not the one it is posted on.

1 INTRODUCTION

With the rapid development of e-commerce, the number of online items (products) has witnessed an explosive growth (e.g., Amazon, Taobao). Mining useful items may greatly ease and accelerate the buying process [17]. Experienced users (customers) leave reviews (comments) about the items and helpful reviews can help users acquire detailed information about a certain item, and affect other users' buying decisions. Thus identifying helpful reviews can bridge the gap between users and items and accelerate the business procedure. This leads to an important research problem — review helpfulness prediction [6], which aims to identify helpful reviews from the massive reviews posted by experienced users.

Taobao is one of the largest online e-commerce sites and tens of millions of reviews are posted by users every day.[1] However,

[1]According to https://www.alibabagroup.com/en/news/article?news=p160811

many of the reviews are not *helpful* for users to decide whether they should buy an item or not (See Figure 1). In this paper, we carefully re-investigate the real-world reviews from *Taobao* and summarize the challenges of review helpfulness prediction from industry in the following two aspects:

(1) **Combining multiple data sources.** The helpfulness of a review is not only determined by the review itself, but also related to the user who posted it and the item that this review is commented on. Moreover, the neighborhood of the review (reviews posted under the same item, reviews posted by the same user) could also help to predict the helpfulness of this review. Thus the first challenge is how to effectively combine these data sources and predict the helpfulness of a review.
(2) **Heterogeneity of criteria for classifying review helpfulness on different items.** The second challenge is that the criteria of classifying review helpfulness on different items are not always the same. For example in Figure 1, the criteria of classifying unhelpful reviews of the first two types (*Type 1* and *Type 2*) are the same — the reviews are labeled as unhelpful on any items since they are general or even not related to a certain item. However, for *Type 3*, the criteria of classifying helpful reviews are different for different items. For example, the last review in Figure 1 may be labeled as helpful if it is posted on a *screen protector*. However, here it is posted on *fuel card*, thus it is unhelpful.

Many machine learning methods have been proposed to tackle the above two challenges. They can be classified according to the data sources used as follows: (1) Researchers have proposed many solutions that only relies on the reviews, by either using domain-specific knowledge [23] or deep neural networks [4] to extract the features from the reviews, then apply them to traditional machine learning models like Logistic Regression or Random Forest. (2) Fan et al. [7] incorporated more information of the item and used deep neural networks to better predict the review helpfulness. (3) Li et al. [15] further involved the information of the users and adopted a graph structure to organize the multiple data sources. Based on this graph representation, the authors modeled the target as an edge-classification problem and proposed to use graph neural networks to train and classify the reviews.[2]

However, none of the existing methods can tackle all of the above two challenges. The first two types of methods did not combine all the data sources and the third type of the work did not consider the heterogeneity of criteria for classifying review helpfulness on different items — GNN maps two nodes to similar representations if these two nodes are close in the graph structure. For example, if a user posted two reviews on two different items, then these two items are connected to the same node (i.e., the user), which means that GNN would map these two items to similar representations. However, the criteria of identifying helpful reviews on different items are different, thus trivial GNN would mix up the criteria on these two items and fail to address the second challenge above.

Motivated by this, we propose CA-GNN to tackle both of the above two challenges, using graph neural network to identify helpful reviews in a *multi-task* manner. To combine multiple data sources, we follow [15] to organize them as a *user-review-item* graph. To handle the heterogeneity of criteria for classifying reviews, we employ GNN with one *shared* and many *item-specific* graph convolutions to learn the common features and each item's specific criterion for classifying reviews simultaneously. To further reduce the number of parameters in CA-GNN and boost its performance, we propose to partition the items into several clusters according to their *category information*, where the items in one cluster share a common graph convolution (category-aware graph convolution). We implement CA-GNN on TensorFlow [1] and conduct solid experiments to verify the effectiveness of our proposed method.

Our contributions can be summarized as follows:

(1) We re-investigate the challenges of review helpfulness prediction using real-world examples from *Taobao* and identify the heterogeneity of criteria for classifying helpful reviews on different items.
(2) To deal with the heterogeneity of criteria on different items, we proposed CA-GNN, which uses graph neural network to deal with review helpfulness prediction in a multi-task manner.
(3) To reduce the number of parameters and boost the performance of CA-GNN, we propose to partition the items into several clusters according to their category information, such that items in one category share a common graph convolution.
(4) We conduct solid experiments on two public datasets and show that CA-GNN outperforms existing methods by up to 10.9% in AUC. We also deploy our system in Taobao with online A/B Test and show that CA-GNN still outperforms the baseline system in most cases.

2 RELATED WORK

In this section, we present the existing works on review helpfulness prediction according to the data sources they used.

2.1 Only Reviews

Many conventional approaches on review helpfulness prediction use domain-specific knowledge to extract a wide range of hand-crafted features from only the text of reviews and then feed them into machine learning models like logistic regression, random forest and gradient boosting trees.

These features can be categorized into two classes. The first one is using hand-crafted features, like structural features (STR) [23], lexical features (LEX) [22], syntactic features (SYN) [11], emotional features (GALC) [19], semantic features (INQUIRER) [25] and argument features (ARG) [18]. The second one is using deep neural networks to automatically extract the features. Embedding-gated CNN (EG-CNN) [3, 4] adopted convolutional neural networks to extract multi-granularity text features from the text reviews. Fan et al. [8] introduced a multi-task neural learning paradigm (MTNL) for identifying helpful reviews. Apart from identifying the helpful reviews, they used the convoluted representation to predict the star ratings of reviews as an auxiliary task.

Different from the above methods, we propose to use all the data sources, not only the reviews but also the users and items together to predict the helpfulness of a review.

[2]Though this work is proposed for spam review detection, it also involves the information about reviews, users and items, and aims to classify the reviews. Thus we consider it similar to review helpfulness prediction problem in this paper.

2.2 Reviews + Items

Apart from the text of reviews, Fan et al. [7] believed that the helpfulness of a review should be fully aware of the metadata of the item (e.g., the title, the brand, the category and the description). In that work, the authors proposed an end-to-end deep neural architecture (PRH-Net) directly fed by both the meta data of the item and the raw text of the review, to get an item-aware representation. Thus the learned representation was expected to capture both the information about the reviews and the items.

Different from this work, we use more data like information of users and construct a *user-review-item* graph to learn the representation of reviews by mining the high-order information in graphs.

2.3 Reviews + Items + Users

To detect spam reviews in a second-hand goods app, Li et al. [15] further involved the information of the users and constructed a *user-review-item* graph. They formulated the problem as an edge classification task, and proposed GAS, which employed GNNs on the heterogeneous graph to further mine the high-order information in the graph, and generated a representation for each edge. GAS was proposed for spam review detection while we focus on review helpfulness prediction, in which the criteria of classifying reviews on different items are different as we discussed in Section 1 and that for spam detection are similar among all items.

To learn node representations in homogeneous graphs, Wu et al. [21] presented a degree-specific multi-task graph convolution function to learn the shared sub-structures and degree-specific neighborhood structures simultaneously. In this work we identify the heterogeneity of criteria for classifying helpful reviews on different items and tackle the heterogeneity by extending [21] to heterogeneous graphs with trainable edge embeddings.

3 PROPOSED METHOD

In this section, we present the preliminaries of review helpfulness prediction and introduce our proposed model.

3.1 Preliminary

Review Helpfulness Prediction. The goal of review helpfulness prediction is to predict whether a review on a certain item is helpful, which can help users quickly acquire detailed information of items. Review helpfulness prediction can be modeled as a supervised machine learning task. The input contains the information of the reviews (R), the items (I) that the reviews were commented on and the users (U) who posted the reviews. The output is a label (Y) indicating that whether this review is helpful or not. The machine learning task can be formulated as follows:

$$min_\theta L(f(\theta, U, I, R), Y) \tag{1}$$

Here L is the loss function, f is the model and θ is the model parameter. The target of this task is to find the best parameter θ that minimizes the above equation.

Graph Neural Networks. Graph neural networks (GNN) [5, 9, 13] introduced machine learning techniques to graphs, having good performance in scenarios such as anomaly detection and e-commerce recommendations [16, 27]. According to [24], the computation of GNN follows a layer-wise propagation manner. In each propagation layer, all the nodes aggregate their neighbors and combine them simultaneously. A propagation layer can be separated into two operations: aggregation and combination. Aggregation is a function for aggregating the embeddings of neighbors, which could be mean pooling, max pooling, attention mechanism, LSTM, etc. Combination is used to combine the self embedding and the aggregated neighbor embedding, which could be concatenation, etc.

Notations	Meanings
$U = \{u_1, u_2, ...\}$	the set of users
$I = \{i_1, i_2, ...\}$	the set of items
$R = \{r_1, r_2, ...\}$	the set of reviews
$G(U \cup I, R)$	the graph constructed from U, I and R
$N(q)$	the neighbors of node/edge q
$MEAN$	the mean aggregator
$Conv1D$	the 1-dimensional convolutional layer
h_q^l	the hidden state of node/edge q at the l-th layer

Table 1: Notations frequently used in the paper.

It is a natural idea to organize the information of users, items and reviews as a graph $G(U \cup I, R)$, where the users and items are nodes, and R is the set of edges(reviews). A user and an item is connected if the user has commented on this item. Based on the graph structure, we can apply GNNs to mine the structural information of the graph since the representation of one node is updated by not only its own value but also its neighbors. To better present the paper, we summarize the notations used in Table 1.

3.2 The CA-GNN Framework

As we discussed before, trivial GNNs cannot handle the heterogeneity of criteria for classifying reviews on different items well, since GNNs would map nodes that are close in the graph structure to similar embeddings. Motivated by this, in CA-GNN we customize graph convolutions for each item and those reviews posted on it. Borrowing the idea from [21], we model this as a multi-task optimization problem. We employ a graph neural network with a global and many local item-specific graph convolutions to learn the common features and each item's specific criterion for classifying reviews simultaneously. Specifically, we use a global trainable matrix shared by all items to learn the common features, and add a local trainable matrix for each item to learn its specific criterion. In each layer of CA-GNN, embeddings of users, items and reviews are updated separately. We next present how the node embeddings in each layer are updated.

Items. In each layer, each item combines information from both itself and its neighbors (e.g., the reviews and the users). To customize the graph convolution for each item, we add a local weight matrix for both the local state of itself and its neighborhood. We assume $W_{I\_G}^l$ and $W_{I\_L(T(i))}^l$ are the shared trainable matrix and the item-specific matrix, respectively. The subscript G and L means 'Global' and 'Local' separately, and $T(i)$ means task(i) that the item belongs to. Similarly, $V_{I\_G}^l$ and $V_{I\_L(T(i))}^l$ are the shared trainable matrix and the task-specific matrix for learning from the item itself. The computation process of each item node is presented in Figure 2. In detail, the process is as follows.

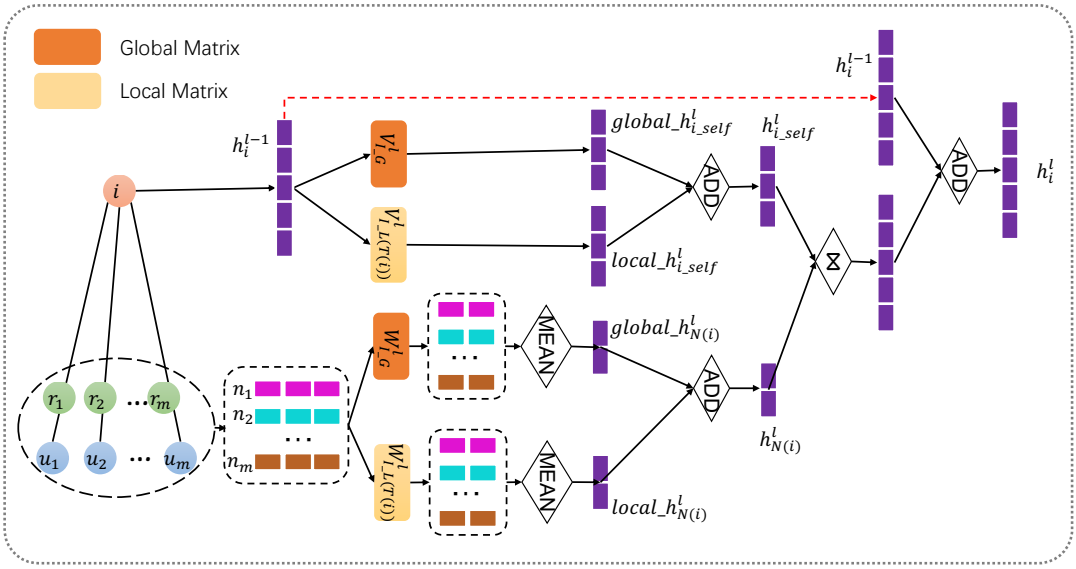

Figure 2: The procedure of updating one item's hidden state in the l-th layer. Each item combines the information from itself and its neighbors. Apart from the two global matrices (e.g., $W^l_{I\_G}$ and $V^l_{I\_G}$) shared by all items, we add two local item-specific matrices (e.g., $W^l_{I\_L(T(i))}$ and $V^l_{I\_L(T(i))}$) to learn each item's specific criterion. $T(i)$ is the task that item i belongs to in the *multi-task* setting.

(a) To compute the global and local aggregated embedding of the neighborhood, we follow:

$$global\_h^l_{N(i)} = \sigma\left(MEAN(Conv1D(H^{l-1}_{UR}, W^l_{I\_G}))\right) \tag{2}$$

and

$$local\_h^l_{N(i)} = \sigma\left(MEAN(Conv1D(H^{l-1}_{UR}, W^l_{I\_L(T(i))}))\right) \tag{3}$$

where H^{l-1}_{UR} denotes the neighborhood of the item:

$$H^{l-1}_{UR} = \left\{\text{concat}\left(h^{l-1}_u, h^{l-1}_r\right), \forall u, r \in N(i)\right\} \tag{4}$$

Here $Conv1D$ is an one-dimensional convolution layer we used to extract the key features. The aggregated neighbor embedding h_N is the sum of the global neighbor embedding and the local neighbor embedding $h^l_{N(i)}$:

$$h^l_{N(i)} = global\_h^l_{N(i)} + local\_h^l_{N(i)} \tag{5}$$

(b) To learn from the features of the item itself:

$$global\_h^l_{i\_self} = Conv1D(h^{l-1}_i, V^l_{I\_G}) \tag{6}$$

$$local\_h^l_{i\_self} = Conv1D(h^{l-1}_i, V^l_{I\_L(T(i))}) \tag{7}$$

Also, the self embedding $h^l_{i\_self}$ is the sum of the global self embedding and the local self embedding:

$$h^l_{i\_self} = global\_h^l_{i\_self} + local\_h^l_{i\_self} \tag{8}$$

(c) Finally the hidden state of item i at the l-th layer is the combination of $h^l_{i\_self}$ and $h^l_{N(i)}$:

$$h^l_i = \text{concat}\left(h^l_{i\_self}, h^l_{N(i)}\right) \tag{9}$$

Users. In CA-GNN, we treat all users equally and do not add a local graph convolution for each user. In each layer, there are two global trainable matrix for all users (W^l_U for its neighbors' state, V^l_U for its own state). Similar to items, the updating procedure of users are as follows:

(a) To compute the aggregate embedding of the neighborhood, we follow:

$$h^l_{N(u)} = \sigma\left(MEAN(Conv1D(H^{l-1}_{IR}, W^l_U))\right) \tag{10}$$

where H^{l-1}_{IR} presents the neighborhood of the user:

$$H^{l-1}_{IR} = \left\{\text{concat}\left(h^{l-1}_i, h^{l-1}_r\right), \forall i, r \in N(u)\right\} \tag{11}$$

(b) To learn from the features of user itself:

$$h^l_{u\_self} = Conv1D(h^{l-1}_u, V^l_U) \tag{12}$$

(c) After adding the residual connection, the user hidden state of the l-th layer can be written as:

$$h^l_u = \text{concat}\left(h^l_{u\_self}, h^l_{N(u)}\right) \tag{13}$$

Reviews. Reviews are edges in the graph and we treat them equally similar as we did in updating users. In each layer, there is only one global trainable matrix W^l_R for all reviews. To compute the aggregated embedding, we follow:

$$h^l_{N(r)} = W^l_R \cdot \text{concat}\left(h^{l-1}_{U(r)}, h^{l-1}_{I(r)}\right) + b^l_R \tag{14}$$

Then the hidden state of review r at the l-th layer is:

$$h^l_r = \sigma\left(\text{concat}\left(h^{l-1}_r, h^l_{N(r)}\right)\right) \tag{15}$$

Remark. We set initial state of users and items (i.e., h^0_u and h^0_i) as their input features, while the initial embedding of the review (h^0_r) is generated from the review text by embedding methods like

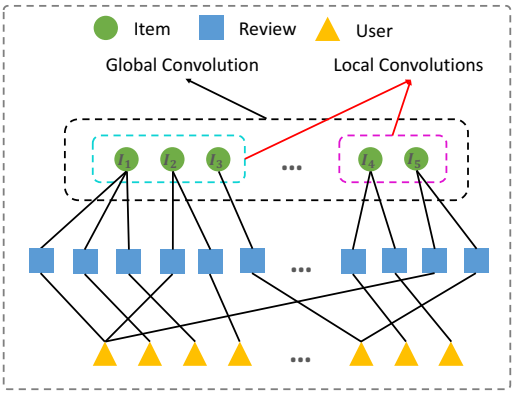

Figure 3: Use category-aware clustering to reduce number of local convolutions, in which the items in the same cluster share one local graph convolution.

Doc2Vec [14]. We provide two solutions for the training of CA-GNN. When a graph is small, we train CA-GNN in a full batch manner. All of the node embeddings and edge embeddings are updated simultaneously. When a graph is large, e..g, it contains hundreds of millions of nodes, we train CA-GNN in a mini-batch manner, where a fixed number of neighbors are randomly sampled for each node.

3.3 Using Clustering to Optimize CA-GNN

We first analyze the complexity of CA-GNN and identify the expensive memory usage of CA-GNN. Then we propose to use clustering to reduce the number of parameters in CA-GNN and further boost its performance.

3.3.1 *Memory Analysis of CA-GNN.*

We analyze the memory complexity of CA-GNN. Suppose we have K items and the amount of parameters for updating items, users and reviews in a trivial GNN are n_i, n_u and n_r, respectively.

Then the number of parameters in CA-GNN is $(K+1)n_i+n_u+n_r$ since we add one graph convolution for each item, while solely using a global and shared matrix for the users and reviews. Thus the number of parameters increases by $\frac{(K+1)n_i+n_u+n_r}{n_i+n_u+n_r}$. (When the number of items is huge, the size of parameters of users and reviews can be neglected, in which case the size of the model increases by $(K+1)\times$.)

3.3.2 *Category-Aware Clustering.*

According to the analysis above, we find that the memory complexity of training CA-GNN is too high and using one graph convolution for each item is unfeasible due to the following two reasons:

(1) It leads to a large number of model parameters, which indicates expensive memory usage.
(2) The number of reviews on one item is probably not enough. And the model tends to overfit with only a few samples.

To tackle the above two challenges, a possible solution is to let some items share one local graph convolution — we let each task contain multiple items (compared with one item per task in Section 3.2). As we observed from the real-world datasets (e.g., Amazon and Yelp), the criteria of predicting review helpfulness on similar items are similar. Thus we propose to use the category information to partition the items into multiple clusters, such that items in the same cluster share one local graph convolution. Figure 3 depicts a simple example of using clustering to reduce the number of parameters in CA-GNN. The items in one cluster (e.g., I_1, I_2, I_3) are treated as one task and share one local graph convolution.

In this case, the size of model is related to the number of clusters rather than the number of items. Moreover, the items in one cluster could help each other on training the model. As for item clustering, we generate the embeddings of item categories using the method described in [2], and use KMeans as the clustering method.

4 EXPERIMENTS

In this section, we evaluate CA-GNN on both public datasets and real-world industrial cases.

4.1 Experiments on Public Datasets

We first present the experimental setup and then compare CA-GNN with existing competitors on two public datasets.

4.1.1 *Experimental Setup.*

Datasets. We evaluate CA-GNN on two public real-world datasets, i.e., Amazon[3] and Yelp[4]. Yelp (Health&Medical) contains the health and medical information from Yelp, including 155,944 users, 16,157 items and 217,244 reviews. Amazon (Pet Supplies) contains the pet supplies information from Amazon, including 740,985 users, 103,288 items and 1,235,316 reviews. We follow Fan et al. [7] to process the labels of each data point. We mark a review with at least one vote as labeled, while others as unlabeled. Reviews receiving at least 0.75 ratio of helpfulness/usefulness are regarded as positive samples, and the rest are negative samples. The training, validation and testing sets are randomly split with a ratio of 7:1:2.

Baselines. We compare CA-GNN with the following methods:

(1) Existing methods that rely only on the text of reviews like STR [23], LEX [22], GALC [19], INQUIRER [25], EG-CNN [4] and MTNL [8].
(2) Existing method relying on reviews on items. PRH-Net [7] incorporates the review data and the meta data of the items (e.g., the title, the brand, the description, etc), and uses deep neural networks to better predict the helpfulness of reviews.
(3) Existing method relying on reviews, items and users. GAS [15] first applies GNN to spam review detection in e-commerce scenarios. We include GAS as the baseline since it is the only work that tries to apply graph neural network to online review classification. The result of GAS is similar to that we set number of clusters as one in CA-GNN.

Parameter Settings. We train GAS using the same hyperparameter as in Li et al. [15], except that we sample four closest reviews for each user and eight reviews for each item due to the limitation of GPU memory. In CA-GNN, the dimension of review embeddings pretrained by Doc2Vec [14] is set as 64. The dimension of the item category embeddings is set as 128. The number of GCN layers in CA-GNN is set as 2. CA-GNN is trained in a full-batch manner. All the experiments are repeated five times and we report the average.

[3] http://jmcauley.ucsd.edu/data/amazon/links.html
[4] https://www.yelp.com/dataset

Data source	Method	Yelp	Amazon
review	STR	0.525	0.560
	LEX	0.517	0.542
	GALC	0.538	0.585
	INQUIRER	0.576	0.603
	EG-CNN	0.580	0.580
	MTNL	0.596	0.619
review+item	PRH-Net	*0.665**	*0.679**
review+item+user	GAS	0.661	0.666
	CA-GNN (Ours)	**0.708**	**0.753**
		(6.5%)	(10.9%)

Table 2: Comparison of AUC of CA-GNN and previous works. (*italic fonts: the best performance among the baseline approaches. bold fonts: the state-of-the-art performance of all the approaches. (.): the improved percentage of CA-GNN against the best performance among the baselines.)**

Evaluation Metric. We use AUC (Area under Receiver Operating Characteristic) as the metric to compare the performance of all the methods, since the datasets we use are imbalanced. Note that a higher AUC indicates a better model.

4.1.2 Result Analysis.

Table 2 presents the experimental result of all baselines on Yelp and Amazon dataset.

We can observe several facts. First, the methods that only rely on reviews perform worse than methods that rely on more data sources like items and users. For example, PRH-Net is better than all review-based methods like STR, LEX, etc. The reason is simple that they exploited less data sources.

Second, the performance of GAS is worse than PRH-Net, though it employed GNN to combine all information about users, items and reviews. For example, GAS is worse than PRH-Net by 0.004 and 0.013 in terms of AUC on Yelp and Amazon dataset, respectively. The reasons are as follows: (1) The criteria of classifying review helpfulness on different items are not the same. As we discussed in Section 3.2, trivial GNN would map nodes that are close in graph structure to similar representation, thus often mix up the criteria of different items. (2) GAS adopted TextCNN [12] to extract the features from the reviews, which performs well when classification is determined by some local key phrases [26]. However, it is hard to determine whether a review is helpful based on several phrases extracted from the review text.

Finally, our proposed model performs better than all the baselines. For example, AUC of CA-GNN is higher by 6.5% and 10.9% than PRH-Net, and 7.1% and 13.1% than GAS, on Yelp and Amazon dataset, respectively. We can observe that CA-GNN is better than PRH-Net because we used more high-order information hidden in the graph structure. CA-GNN is better than GAS because we considered the heterogeneity of classifying reviews on different items and further used local graph convolutions to characterize the criteria of classifying helpful reviews on different tasks.

4.1.3 Parameter Sensitivity of Number of Clusters.

We study how the number of clusters affects the accuracy of CA-GNN. For each dataset, we set different number of clusters and report the AUC. Figure 4 presents the sensitivity analysis on Amazon. The results on Yelp are omitted due to the space limitation.

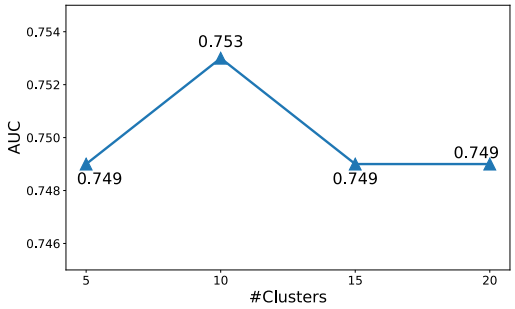

Figure 4: AUC of different number of clusters on Amazon.

We find when increasing the number of clusters, AUC of CA-GNN on Amazon first increases then decreases. For example, AUC of CA-GNN on Amazon dataset gets to the best when the number of cluster is 10. That the AUC first increases when the number of clusters increases validates our observation of the heterogeneity of criteria for classifying review helpfulness on different items in Figure 1. When using small number of clusters, criteria of different items are mixed, the model can not perform well. The AUC later decreases after it reaches the highest on both datasets. We speculate it is because of the overfitting: when using more clusters, there are fewer items in each cluster, which may cause the model too attuned to the training data, while decreasing its generalization ability.

4.2 Experiments on Taobao Review Ranking System

We have deployed our proposed method on *Taobao* to evaluate its effectiveness with a standard A/B test. We next present the experimental setup and the evaluation results.

4.2.1 Experimental Setup.

Workflow of the Online Evaluation. We integrate CA-GNN into a review ranking system and conduct a standard A/B test to evaluate the effectiveness of our proposed model as shown in Figure 5. We select seven categories of items for evaluation. Users from each category are randomly sampled and divided into two buckets for evaluation. For users in bucket A, we use CA-GNN to predict the helpfulness score of all reviews. Note that CA-GNN is designed for offline cases and does not employ online features (e.g., user behavior sequence). Thus CA-GNN only consumes the offline features. For users in bucket B, we adopt existing highly optimized model, gradient boosting decision tree (GBDT) [10]. GBDT is popular in industrial cases for several reasons: (1) The accuracy of GBDT is often better than other traditional machine learning models. (2) The output of tree models is often explainable and could further guide the engineering process. (3) The training and inference process of GBDT are fast and can facilitate the online requirement. The GBDT model used here consumes both offline features and online features. Specifically, given an unlabeled review, we collect its neighbors from the *user-review-item* graph constructed offline (offline features) and user temporal information (online features). We feed these features to a machine learning model (e.g., CA-GNN or GBDT) and the model outputs the helpfulness score of the review.

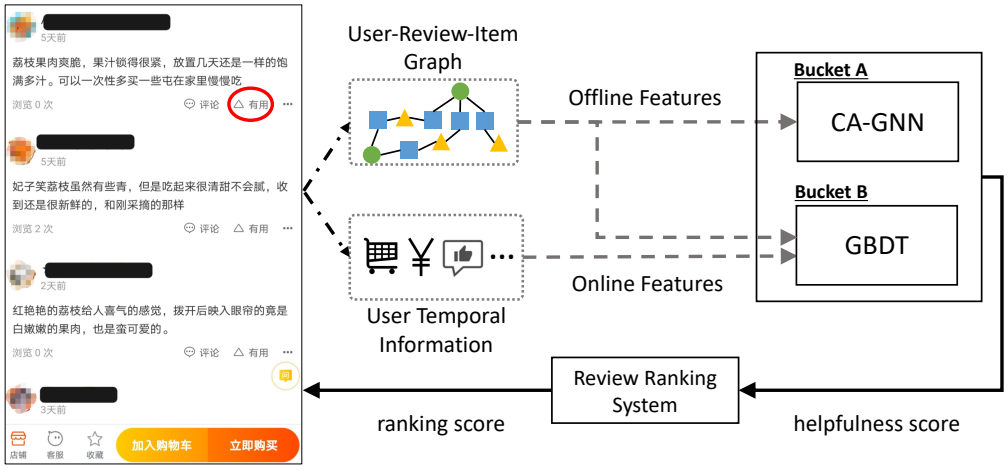

Figure 5: Workflow of the online evaluation. Given an unlabeled review, we first collect its neighbors from the *user-review-item* graph constructed offline (offline features) and user temporal information (online features). We then feed these features into a machine learning model (i.e., CA-GNN or GBDT) and the model outputs the helpfulness score of the input review. Finally the review ranking system takes the helpfulness score as input and outputs the ranking score to decide where to display the review. The *user-review-item* graph is re-constructed once per day. There is a helpful button for each review (the red circle on the Taobao UI in this figure). If a user thinks one review is helpful, he/she can click the buttion and vote for it.

Finally the review ranking system takes the helpfulness score as input and outputs a ranking score to decide where to present the review. Since there may exist new users and items, we re-construct the *user-review-item* graph once per day.

Implementation Details. We deploy CA-GNN using TensorFlow [1] on PAI[5], the Alibaba Cloud AI Platform. We use ten workers and ten parameter servers. Each worker has one Nvidia V100 GPU, six CPU cores, and 32GB memory. Each server has two CPU cores and 300GB memory. The training process takes 20 minutes, and the score generating process takes about 2 hours.

Evaluation Metric. The online A/B test lasted for 7 days. All settings of the two buckets are kept the same except the helpfulness scoring models. We collect the click actions of the users, and use the helpful review Click Through Rate (hrCTR) to measure the performance. The hrCTR can be calculated as the ratio between the number of clicks on the helpful button and the number of times that the review has been shown:

$$hrCTR = \frac{\#clicks\ on\ the\ helpful\ button}{\#reviews\ are\ shown} \tag{16}$$

The higher hrCTR is, the better the model is.

4.2.2 *Result Analysis.*

Figure 6 presents the evaluation results of CA-GNN and GBDT using A/B test on seven categories.

We observe two facts. First, CA-GNN outperforms GBDT on most categories. For example, CA-GNN achieves higher hrCTR than GBDT by 3.13%, 3.10%, 3.06%, 0.17%, 1.58% on *Women's Shoes, 3C Digital Accessories, Cooked Food, Snacks* and *Beauty Makeup*, respectively. Though GBDT consumes both offline and online features while CA-GNN only uses offlines features, CA-GNN still performs better on these five categories because CA-GNN can capture the neighborhood using the graph structure and model the heterogeneity of criteria for classifying review helpfulness on different items, which is in line with the analysis in Section 3.

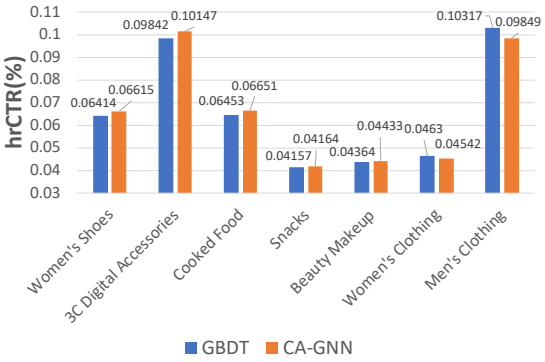

Figure 6: Evaluation of CA-GNN using A/B Test.

Second, CA-GNN fails to beat GBDT on two categories, i.e, *Women's Clothing and Men's Clothing*. When we dig into the data on these two categories, we find that there are many more online features than for the other five categories.

4.3 Experiments on Taobao Offline Dataset

As we mentioned above, GBDT consumes more features than CA-GNN since it can employ online features. Thus to make a fair comparison, we construct a large-scale offline dataset from Taobao with only offline features and compare CA-GNN with GBDT again.

4.3.1 *Experimental Setup.*

Dataset. We randomly sampled ~100 million users, ~21 million items and ~360 million reviews from *Taobao* to construct a large-scale heterogeneous graph. Moreover, we sampled 117, 127 helpful reviews and 40, 927 unhelpful reviews manually labeled by human experts. Note that the dataset is used for evaluating the performance

[5]https://www.aliyun.com/product/bigdata/product/learn

of CA-GNN, and any derived information (e.g. the numbers of items, users and reviews, and the helpful review-unhelpful review ratio) does not represent the true scenario of *Taobao*.

Parameter Settings. We use Word2Vec [20] to generate embedding of the words in each review, and use the average as the review's embedding. The dimension of the embedding is set as 32. For GBDT, we concatenate the features of the review, the user who posted this review and the item this review is about as the input feature. GBDT is trained using 200 trees with learning rate set as 0.1. We train CA-GNN in a mini-batch manner, where the fixed numbers of neighbors sampled for each user and item are 5 and 10 respectively. To handle the large scale graph with hundreds of millions of reviews, we set the number of propagation layer as one. We use AUC to compare the performance of CA-GNN and GBDT in this offline case.

4.3.2 Result Analysis.

Table 3 compares CA-GNN with GBDT on the offline dataset. We can find that CA-GNN outperforms GBDT significantly when they use same input features. For example, the AUC of CA-GNN is higher than that of GBDT by 10.3% on this offline dataset. Though GBDT combines features from users, items and reviews, it fails to address the heterogeneity of criteria for classifying helpful reviews. However, CA-GNN proposes to use many local convolutions to characterize different criteria of classifying helpful reviews on different items.

Method	AUC
GBDT	0.821
CA-GNN (Ours)	0.924

Table 3: Comparison Results on Offline Dataset

In summary, CA-GNN outperforms GBDT significantly when there is only offline features. Moreover, it even beats GBDT when GBDT uses both online and offline features while CA-GNN only uses offline features in most cases. This indicates that our proposed model is competitive on using offline features. We leave CA-GNN on online features as our future work.

5 CONCLUSION

In this paper, we carefully re-investigate the challenges of review helpfulness prediction using real-world examples. We identify the heterogeneity of criteria for classifying review helpfulness on different items in e-commerce scenarios, and propose CA-GNN that exploits graph neural networks to identify helpful reviews in a multi-task manner. Specifically, we use graph neural networks with one shared and many item-specific graph convolutions to learn the common features and each item's specific criterion for predicting review helpfulness. To reduce the number of parameters in CA-GNN, we partition the items into clusters based on the category information, such that items in the same cluster share a common graph convolution. We conduct experiments on two public datasets and results show that CA-GNN outperforms existing methods significantly. We also deploy our method in Taobao and show CA-GNN is competitive to existing online solutions. Our proposed model can serve as a complement to the online review ranking system in the days ahead.

ACKNOWLEDGMENTS

This work was partially supported by NSFC under Grant No. 61832001, Alibaba-PKU joint program, and Zhejiang Lab under Grant No. 2019KB0AB06.

REFERENCES

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *ODSI*. 265–283.
[2] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *ICLR*.
[3] Cen Chen, Minghui Qiu, Yinfei Yang, Jun Zhou, Jun Huang, Xiaolong Li, and Forrest Sheng Bao. 2018. Review Helpfulness Prediction with Embedding-Gated CNN. *CoRR* abs/1808.09896 (2018).
[4] Cen Chen, Yinfei Yang, Jun Zhou, Xiaolong Li, and Forrest Sheng Bao. 2018. Cross-Domain Review Helpfulness Prediction Based on Convolutional Neural Networks with Auxiliary Domain Discriminators. In *NAACL-HLT*. 602–607.
[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.
[6] Gerardo Ocampo Diaz and Vincent Ng. 2018. Modeling and Prediction of Online Product Review Helpfulness: A Survey. In *ACL*. 698–708.
[7] Miao Fan, Chao Feng, Lin Guo, Mingming Sun, and Ping Li. 2019. Product-Aware Helpfulness Prediction of Online Reviews. In *WWW*. 2715–2721.
[8] Miao Fan, Yue Feng, Mingming Sun, Ping Li, Haifeng Wang, and Jianmin Wang. 2018. Multi-Task Neural Learning Architecture for End-to-End Identification of Helpful Reviews. In *ASONAM*. 343–350.
[9] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.
[10] Jiawei Jiang, Bin Cui, Ce Zhang, and Fangcheng Fu. 2018. DimBoost: Boosting Gradient Boosting Decision Tree to Higher Dimensions. In *SIGMOD*.
[11] Soo-Min Kim, Patrick Pantel, Timothy Chklovski, and Marco Pennacchiotti. 2006. Automatically Assessing Review Helpfulness. In *EMNLP*. 423–430.
[12] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. 1746–1751.
[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[14] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*. 1188–1196.
[15] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. 2019. Spam Review Detection with Graph Convolutional Networks. In *CIKM*. 2703–2711.
[16] Zhao Li, Xin Shen, Yuhang Jiao, Xuming Pan, Pengcheng Zou, Xianling Meng, Chengwei Yao, and Jiajun Bu. 2020. Hierarchical Bipartite Graph Neural Networks: Towards Large-Scale E-commerce Applications. In *ICDE*. 1677–1688.
[17] Jerry Chun-Wei Lin, Jiexiong Zhang, Philippe Fournier-Viger, Tzung-Pei Hong, and Ji Zhang. 2017. A two-phase approach to mine short-period high-utility itemsets in transactional databases. *Advanced Engineering Informatics* 33 (2017), 29–43.
[18] Haijing Liu, Yang Gao, Pin Lv, Mengxue Li, Shiqiang Geng, Minglan Li, and Hao Wang. 2017. Using Argument-based Features to Predict and Analyse Review Helpfulness. In *EMNLP*. 1358–1363.
[19] Lionel Martin and Pearl Pu. 2014. Prediction of Helpful Reviews Using Emotions Extraction. In *AAAI*. 1551–1557.
[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.
[21] Jun Wu, Jingrui He, and Jiejun Xu. 2019. DEMO-Net: Degree-specific Graph Neural Networks for Node and Graph Classification. In *SIGKDD*. 406–415.
[22] Wenting Xiong and Diane J. Litman. 2011. Automatically Predicting Peer-Review Helpfulness. In *ACL*. 502–507.
[23] Wenting Xiong and Diane J. Litman. 2014. Empirical analysis of exploiting review helpfulness for extractive summarization of online reviews. In *COLING*. 1985–1995.
[24] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.
[25] Yinfei Yang, Yaowei Yan, Minghui Qiu, and Forrest Sheng Bao. 2015. Semantic Analysis and Helpfulness Prediction of Text for Online Product Reviews. In *ACL*. 38–44.
[26] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative Study of CNN and RNN for Natural Language Processing. *CoRR* abs/1702.01923 (2017).
[27] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. 2019. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN. In *IJCAI*. 4419–4425.