Spammer Detection using Deep Learning Methods

Keyuan Wang
cs230 Final Project Report
kwang64@stanford.edu

Abstract

The explosive growth in activities on social media and e-commerce platforms have led to significant challenges for platforms to manage fake reviews and spammers. As such, developing algorithms that can filter fake reviews and identify spammers have attracted enormous interest from the artificial intelligence community. Researchers have developed various models to tackle this problem, including models that are based solely on textual features (NLP models) and models that examine relationships between users (GNN). This project combined the advantages of both methods through implementing a state-of-art natural language processing model within a graph-neural-network architecture, and proved that this enhanced method can outperform each of the above architectures individually.

1 Introduction

Fake reviews on social media and commercial platforms have been rampant in the recent decade. As reviews on these platforms play a significant role in influencing customers' purchasing decisions, detecting and filtering spammers become crucial to ensure that the market is not distorted by forged opinions. Traditional spam filtering algorithms attempted to solve this issue by applying classification models such as logistic regression, SVM, RNN/CNN based on text, semantic or metadata features (Ren et al., 2019). Recent researchers (A. Li et al., 2019; Y. Ren et al., 2021; Y. Dou et al., 2020; Z. Liu et al., 2020) pointed to graph neural networks (GNN) as another promising solution that outperforms existing state-of-the-art models. However, these researchers have not explored the opportunity of using deep learning models in the NLP area, such as BERT to further improve their results. This project included one of the NLP state-of-the-art models in a GNN architecture, hoping that the improved model can further enhance the overall performance. The final outcome of the model will be a prediction of whether a reviewer is a spammer or not. Experiments demonstrated that the deep learning NLP model + GNN architecture outperforms traditional machine learning models applied only on textual features.

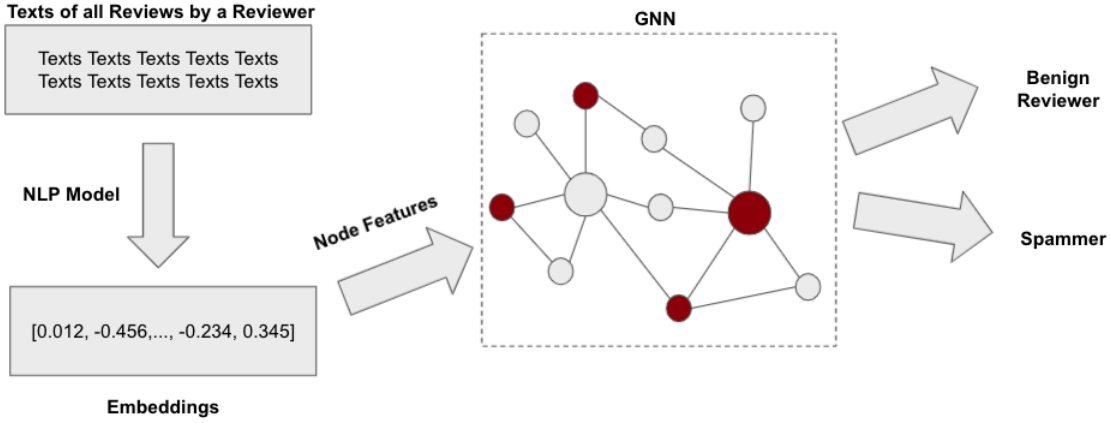
2 Dataset and Preprocessing

The dataset used in this project is the Amazon Home and Kitchen reviews dataset collected from Amazon.com by McAuley et al (2013). The dataset contains 991,794 reviews. Each item in the dataset includes product and user information, timestamp, ratings, plaintext review, and number of helpful/unhelpful votes of the review. Since the dataset does not contain labels for fraud / fake reviews or spammers, I used the proportion of total helpful votes that a user got as a proxy for spammers' label, which is a labeling strategy used by various researchers (Kumar et al., 2018; Dou et al., 2020). Specifically, users with more than 70% helpful votes are labelled as benign users and users with less than 30% helpful votes are labelled as spammers. Given that the raw data only contains less than 10% of spammer labels, and that many reviewers only have a limited number of reviews (which may not fully reflect whether a reviewer is a true spammer or a benign entity), I further filtered out any spammers with less than 5 helpfulness votes and benign reviewers with less than 30 helpfulness votes. The final dataset contains 21,341 benign reviewers and 9,835 spammers.

Next, to construct a graph of reviewers, I applied the graph construction method commonly employed by other researchers (Y. Dou et al., 2020; Z. Liu et al., 2020) on the Amazon dataset, which is to treat reviewers as nodes and connect reviews using U-P-U (connect reviewers by the common products they reviewed), and U-S-U (connect reviewers if they commented with the same rating and during the same month) relationships. The benefit of this construction method is that it retains the user, timing, and product information in the graph. Yuan, et al. (2019) have indicated that these are useful information for spam review detection.

3 Method

The model architecture includes two parts: the first part is a language model that converts the review texts of a user to a vector representation; the second part is a graph neural network, which takes in the review text embedding vectors as node features, and then propagates through the entire network to make final prediction of whether a reviewer is benign or spam. See below for an illustration of this model:



3.1 Language Models

The first language model I experimented with was a pre-trained GloVe model trained over the Wikipedia 2014 corpus (downloaded from https://radimrehurek.com/gensim/). The GloVe model (Pennington et. al., 2014) generates embeddings based on training over word co-occurrence with a weighted-least-square model. Before passing the review texts into the model, I lemmatized the review texts, and further cleaned them up by removing stopwords, numerical values, and punctuations. The result of the model is a 300 dimensional vector for each word. I then took the average value in each dimension of all words in a review and took the resulting vector as the embedding of this review. I didn't fine-tune or change any other hyperparameters in the pre-trained GloVe model.

The other model I experimented with was the uncased DistilBERT model downloaded from huggingface.co. DistilBERT (Sanh et al., 2019) is a variation of BERT (Devlin et al., 2018), a transformer based model that uses the attention mechanism to learn contextual relations between words in a text. BERT is able to achieve state-of-art results in a variety of NLP tasks. DistilBERT combines the idea of knowledge distillation with BERT to make the model more scalable to larger datasets, while at the same time retaining 95% of BERT's predictive power. Before implementing DistilBERT in generating embeddings, I fine-tuned the model over the review texts, using spam / non-spam classification derived using users' helpfulness votes as training labels. I also used the average method to create review texts embeddings from individual word embeddings.

3.2 GNN Models

The graph neural network models I experimented with were Graph Convolutional Network and GraphSAGE. Graph Convolutional Network (Kipf et al., 2017) is a spatial-based method that aggregates feature information from a node's neighborhood and then propagates through the network. GraphSAGE (Hamilton et al., 2017) uses an inductive approach to construct the representation of a node, which only considers a sample of its neighboring nodes in convolutions to improve scalability

and generalization. Following each GCN / GraphSAGE layer, I included a BatchNormalization layer to normalize the results and applied ReLU to add non-linearity. In the last layer I changed the ReLU layer to a sigmoid function to generate final class predictions.

Below is an illustration of the GNN architecture of each layer:

I also experimented with adding multiple MLP layers before passing the embeddings through GraphSAGE. The goal was to explore whether providing more flexibility to the model would improve overall performance. Each MLP layer uses ReLU as its activation and outputs a vector that has the same input dimension as GraphSAGE.

4 Experiments

In the experiments, I used logistic regressions and support vector machines as the baseline models. These baseline models take in the GloVe or DistilBERT review text embeddings as features and directly make predictions on the spammer label. When fine-tuning DistilBert, I used AdamW as the optimizer with a learning rate of $5 * 10^{-5}$, sentence length of 512, and batch size of 8, over a total of 2 iterations.

When training the GNN models, I used Adam as the optimizer with a learning rate of 0.01 and 300 training iterations. The loss function is the Binary Cross Entropy Loss. I also experimented with a range of hyperparameters to search for a set that can optimize performance. Based on the results, I decided to use the following in the final model (Refer to appendix B for the range of hyperparameters I experimented):

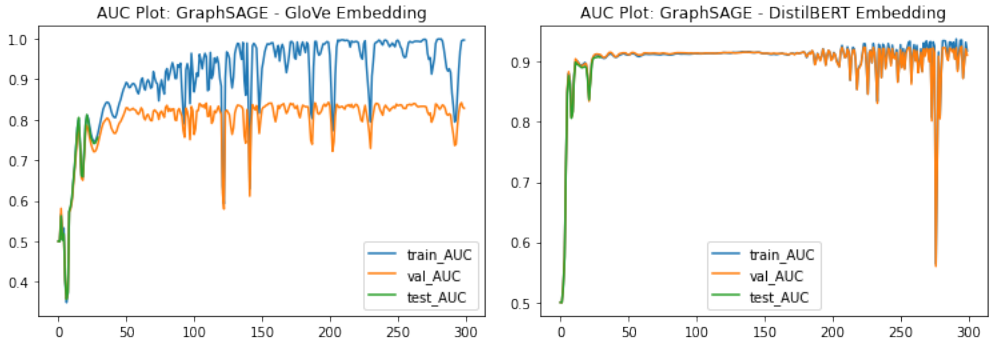
Number of GNN Layers	5
Number of Hidden Dimensions	1,024
Dropout Probability	0.5
Number of MLP Layers (MLP-GraphSAGE Only)	3

Table 1: Hyperparameters

	GloVe Embedding		**DistilBERT Embedding**	
	Validation AUC	Test AUC	Validation AUC	Test AUC
Logistic Regression	73.3%	72.9%	91.3%	90.5%
SVM	82.5%	81.8%	91.5%	90.9%
GCN	72.3%	72.0%	75.3%	74.1%
GraphSAGE	**85.3%**	**86.3%**	92.5%	92.2%
MLP-GraphSAGE	83.8%	83.7%	**92.6%**	**92.3%**

Table 2: Experiment Results

The results indicated that models based on DistilBERT embeddings significantly outperform models based on GloVe embeddings. The AUC plots below also indicated that graph based methods (using GraphSAGE as an example) converge better when using DistilBERT vs. using GloVe as the embedding method, which substantiated that DistilBERT is superior in representing the contextual features of review texts:



The results also showed that graph-based models including GraphSAGE and MLP-GraphSAGE can improve performance compared to models that only consider textual features. Among the five models, MLP-GraphSAGE outputs the highest AUC score when reviews were embedded using DistilBERT, proving that adding MLP layers in front of GraphSAGE can further benefit model performance.

5 Further Discussions

5.1 Usefulness of the reviewer graph network

As demonstrated in the experiment results, graph-based models such as GraphSAGE do have an edge over models that use only textual features, especially in the case when the word embedding method is less effective in fully reflecting the context of the reviews (i.e., GloVe). However, this is only true when the review user graph is able to capture sufficient meaningful relationships. The table below shows the experimental results of the above five models when I vary the size of the dataset (by changing the filtering threshold for benign users). A larger dataset means that the graph contains more nodes and edges and thus can capture a more accurate picture of nodes' relationships. Results indicated that although GraphSAGE and MLP-GraphSAGE still outperform the others when the dataset is smaller, the performance gap between them shrinks. It is reasonable to project that the performance gap between graph-based models and models based on only textual features will be indistinguishable if the number of nodes and edges continue to decrease.

Benign Reviewers Filtering Threshold	**# of Benign Reviewers**	**# of Spammers**	**# of Edges**
>50 helpfulness votes	11,337	9,835	1,084,563
>40 helpfulness votes	15,082	9,835	1,465,763
>30 helpfulness votes	21,341	9,835	2,140,274
>20 helpfulness votes	33,994	9,835	3,714,705

Table 3: Graph Statistics with Different Filtering Criteria for Benign Reviewers

Benign Reviewers Filtering Criteria	**Logistic Regression**	**SVM**	**GNN**	**GraphSAGE**	**MLP-GraphSAGE**	**Performance Gap**
>50 helpfulness votes	93.2%	93.0%	76.7%	**93.6%**	93.3%	0.4%
>40 helpfulness votes	91.6%	91.5%	75.7%	92.4%	**92.5%**	0.9%
>30 helpfulness votes	90.5%	90.9%	74.1%	92.2%	**92.3%**	1.3%
>20 helpfulness votes	86.3%	86.0%	70.0%	**89.3%**	88.7%	3.0%

Table 4: Test AUC of Experimented Models with Different Dataset Sizes (All using DistilBERT Embeddings). Performance gap is defined as the Test AUC differences between the best performing model based only on textual features vs. the best performing GNN.

5.2 GCN vs. GraphSAGE and other Baselines

The other key observation from the above results is that GCN significantly underperforms GraphSAGE and other baseline models. The advantage of GraphSAGE over GCN can be explained by the mathematical formulation of the two models:

GCN: $\mathbf{h}_{\mathbf{v}}^{(l)} = \sigma(\mathbf{W}^{(l)} \sum_{u \in N(v)} \frac{\mathbf{h}_{\mathbf{u}}^{(l-1)}}{|N(\mathbf{v})|})$

GraphSAGE: $\mathbf{h}_{\mathbf{v}}^{(l)} = \sigma(\mathbf{W}^{(l)} \cdot CONCAT(\mathbf{h}_{\mathbf{v}}^{(l-1)}, AGG(\{\mathbf{h}_{\mathbf{u}}^{(l-1)}, \forall \mathbf{u} \in N(\mathbf{v})\})$

where l represents a layer of the GNN model, h_v is the target node, h_u is the target note's neighbors, $N(v)$ are neighboring nodes of the target nodes, W is the weight matrix.

Note that one key difference between GCN and GraphSAGE is that GraphSAGE considers both the user-user relationships and the textual features of a node itself to construct a node's embedding, while GCN, on the other hand, represents a node's features only through aggregating its neighbor's features. Since review text features are strong predictors of node classification (as indicated by the high AUC scores of logistic regression and SVM), GCN's omission of a node's own textual features is unsurprisingly going to significantly impair the model's overall performance. The inferior performance of GCN compared to logistic regression and SVM also demonstrates that although users' relationships can help improve model performance, the textual features are still the dominating factors for this classification task. As such, including robust textual embeddings are indispensable in order for GNN models to achieve the same level or outperform the existing models based solely on textual features.

5.3 Additional Thoughts on Data-Centric vs. Model-Centric AI

One other interesting observation is that the selection of models may not play a significant role when the underlying data are more accurate and more reflective of its true relationship with the prediction target. Here I take the difference between GloVe vs. DistilBERT embedding as a proxy for input data quality (which DistilBERT is considered better). It is notable that the performance difference between SVM and Logistic Regression in Table 2 is much larger when using GloVe embedding as the input (Test AUC difference of 9.9%) vs. using DistilBERT as the input (Test AUC difference of 0.4%). The same is also true on the performance gap between the best performing GNN vs. the best performing traditional machine learning model trained only on textual features (4.5% when using GloVe embedding vs. 1.4% when using DistilBERT embedding). It appears that sophisticated models' strength is more evident in the case when high-quality data are not available, whereas when they are available, a vanilla model can achieve exceptional results as well. As such in practice, instead of merely focusing on training and developing complex models, paying more attention to improving data quality may in certain instances be another efficient way towards state-of-art results.

6 Conclusion

This project implemented an NLP + GNN model in an attempt to improve the existing models that perform spammer classification tasks solely based on graph-level data or on textual features. The results prove that combining a state-of-art language model (DistilBERT) with a classifical GNN architecture (GraphSAGE) or its variation (MLP-GraphSAGE) can achieve this purpose. This project also demonstrates that although user relationships can have a positive impact on model performance, features extracted from a user's review texts are still the strongest predictor of whether a user is a spammer. As such, practitioners should pay attention to the GNN architecture selected to make sure that the model does not overlook the review texts from a node itself in generating predictions.

Appendix A - References

T.N. Kipf and M. Welling. Semi-supervised classification with graph convo- lutional networks. In *ICLR*. 2017.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NeurIPS*. 2017.

C. Yuan, W. Zhou, Q. Ma, S. Lv, J. Han, S. Hu. Learning review representations from user and product level information for spam detection. *arXiv preprint 1909.04455*. 2019.

Y. Ren and D. J. Learning to Detect Deceptive Opinion Spam: A Survey. In *IEEE*. 2019.

A. Li, Z. Qin, R. Liu, Y. Yang, D. Li. Spam Review Detection with Graph Convolutional Networks. *arXiv preprint 1908.10679*. 2019.

Y. Ren, B. Wang, J. Zhang and Y. Chang. Adversarial Active Learning based Heterogeneous Graph Neural Network for Fake News Detection. *arXiv preprint 2101.11206*. 2021.

Y. Dou, Zhiwei Liu, L. Sun, Y. Deng, H. Peng, Philip S. Yu. Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudster. *arXiv preprint 2008.08692*. 2020.

Z. Liu, Y. Dou, Philip S. Yu., Y. Deng, H. Peng. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. *arXiv preprint 2005.00625*. 2020.

J. McAuley, J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. *RecSys*. 2013.

J. Pennington, R. Socher, C. D. Manning. GloVe: Global Vectors for Word Representation. In *Proc. EMNLP'14, pp. 1532–1543*. 2020.

J. Devlin, M. Chang, K. Lee, K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 2020

S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *WSDM. 333–341*. 2018.

J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. 2018

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*. 2019.

Appendix B - Hyperparameters Experimented

Number of GNN Layers	(2, 3, 4, 5, 6)
Number of Hidden Dimensions	(256, 512, 1,024)
Dropout Probability	(0.4, 0.5, 0.6, 0.7)
Number of MLP Layers (MLP-GraphSAGE Only)	(1, 2, 3, 4, 5, 6)

Table 5: Hyperparameters Experimented