Detecting Offensive Language on Social Networks: An End-to-end Detection Method based on Graph Attention Networks

Zhenxiong Miao\textsuperscript{a}, Xingshu Chen\textsuperscript{a,b}, Haizhou Wang\textsuperscript{a,}\textsuperscript{*}, Rui Tang\textsuperscript{a}, Zhou Yang\textsuperscript{a}, Wenyi Tang\textsuperscript{a}

\textsuperscript{a}School of Cyber Science and Engineering, Sichuan University, Chengdu, 610207, China
\textsuperscript{b}Cyber Science Research Institute, Sichuan University, Chengdu, 610207, China

Abstract

The pervasiveness of offensive language on the social network has caused adverse effects on society, such as abusive behavior online. It is urgent to detect offensive language and curb its spread. Existing research shows that methods with community structure features effectively improve the performance of offensive language detection. However, the existing models deal with community structure independently, which seriously affects the effectiveness of detection models. In this paper, we propose an end-to-end method based on community structure and text features for offensive language detection (CT-OLD). Specifically, the community structure features are directly captured by the graph attention network layer, and the text embeddings are taken from the last hidden layer of BERT. Attention mechanisms and position encoding are used to fuse these features. Meanwhile, we add user opinion to the community structure for representing user features. The user opinion is represented by user historical behavior information, which outperforms that represented by text information. Besides the above point, the distribution of users and tweets is unbalanced in the popular datasets, which limits the generalization ability of the model. To address this issue, we construct and release a dataset with reasonable user distribution. Our method outperforms baselines with the F1 score of 89.94\%. The results show that the end-to-end model effectively learns the potential information of community structure and text, and user historical behavior information is more suitable for user opinion representation.

Keywords: Social networks, Deep learning, Offensive language detection, Graph attention networks

2010 MSC: 00-01, 99-00

1. Introduction

In recent years, more and more offensive language appears in social networks, which leads to cyber violence and abusive behavior. The offensive language has become a serious concern for governments and social media platforms \cite{1}. There are several reasons why offensive language is proliferating in social networks. On one hand, users hide their identities in social networks easily. The anonymity of social networks increases the chances that users adopt aggressive behavior. On the other hand, social networks provide an environment where offensive language spread quickly \cite{2}.

Usually, social platforms will adopt detection algorithms to find offensive language automatically and eliminate them. However, it is not easy to automatically identify these contents, especially the offensiveness is contained in the hidden rhetoric and irony \cite{3}. Offensive language can be divided into explicit offensive language and implicit offensive language \cite{11,15}. Explicit offensive language refers to language that is explicitly abusive in its meaning, such as racial discrimination. Implicit offensive language refers to language that does not directly imply abuse, and it is difficult to judge whether it is offensive without understanding the context \cite{4,5}.

For offensive language detection, existing research \cite{7,8,9,10} have done some works based on text, confirming that n-gram features have good performance. Meanwhile, some researchers \cite{11,12,13} have combined user profile features such as age and gender with text features to improve the performance of detection. Recently, more and more methods based on deep learning were applied \cite{14,15,16,17}. The BERT-based methods started to appear in offensive language-related tasks of International Workshop on Semantic Evaluation (SemEval) \cite{18,19,20}. In addition, some works \cite{21,22} have further studied the representation method of user features. They introduced community structure into offensive language detection, which significantly improved the detection performance.

However, there are limitations in existing methods using community structures. These methods \cite{21,22} deal with community structure independently, which seriously affects the effectiveness of user representations. This limits the performance of the offensive language detection model. On the other hand, the existing datasets are mainly collected by searching with the keyword in tweets, only considering the tweets containing explicit offensive

\textsuperscript{*}Corresponding author
Email address: whzh.nc@scu.edu.cn (Haizhou Wang)

Preprint submitted to Journal of BTE\textsuperscript{X} Templates

\textcopyright{} 2022 arXiv:2203.02123v1 [cs.SI] 4 Mar 2022
words, such as [12, 10, 1]. Some abusive tweets that do not contain predefined keywords cannot be collected. This causes the existing models unable to learn relevant feature information and reduces the detection performance and generalization ability of the model in detecting implicit offensive language [23]. Furthermore, some popular datasets may be affected by “author bias” [24, 25]. This means that the classifier can be affected by the unbalanced distribution of users, which limits the generalization ability of the model. For the dataset of Waseem and Hovy [12], more than 70% of the sexist tweets originate from two users, and 99% of the racist tweets originate from one user. Existing research shows that detection models based on such datasets could produce severe user overfitting [25].

To solve these problems, we conduct the work in this paper with the following contributions.

- An end-to-end method based on community structure and text features are proposed for offensive language detection (CT-OLD). It uses Graph Attention Networks (GAT) [26] to learn user representations from community structure and fine-tuned BERT [27] to learn the text representations. The model learns these features at the same time. In addition, attention mechanisms and position coding are used for feature fusion. The CT-OLD achieves an F1 score of 89.94% with our dataset. Moreover, our method has better robustness in different ratios of training and test set. The experimental results show that the end-to-end model significantly improves the current detection performance.

- A novel method is proposed to represent users with user historical behavior information. The user historical behavior information indicates the opinion tendency of users to post offensive language. This method reduces the number of parameters in the graph neural network and improves the performance of the model. For users whose historical behavior is unknown, four feature initialization methods are compared. Our method improves the F1 score by 2.57% over the user representation method in [22].

- A new dataset with community structure and reasonable user distribution is constructed[1]. A method that depend on the lexicon indirectly is used to construct the dataset. Specifically, we first collect the users and relationships rather than tweets. Then the tweets of the users are collected. We collect over 400,000 tweets and over 500,000 users. Then we annotate 12,780 tweets posted by 1260 users. The offensive tweets are posted by 220 users, accounting for 17.5% of the users.

2. Related Work

Recently, research based on social networks has been increasing [28, 29, 30, 31]. In this section, we summarize the research on offensive language detection in social networks in recent years. The methods in offensive language detection research are divided into two main categories: classical machine learning-based methods [7, 9, 11, 12, 13, 21, 35, 36], and deep learning-based methods [14, 15, 16, 20, 22, 23, 27, 30, 37, 38, 39, 40].

2.1. Classical machine learning-based methods

Classical machine learning methods have been widely used in offensive language detection task. It is divided into two categories: methods based on text features [7, 9, 12, 32, 33, 34], and methods combining text and user features [11, 13, 21, 35].

2.1.1. Text-based feature methods

Most of the text feature-based offensive-language detection methods involve supervised text classification tasks, which generally requires feature engineering, such as text features, sentiment features, linguistic features. Text features are calculated by bag-of-words models or word embeddings. Sentiment features of the sentence are obtained by predefined word sentiments. Linguistic features refer to the statistical results of uppercase letters, lowercase letters, and punctuation marks. Using feature engineering methods, various classifiers can be supervised training.

Razavi et al. [32] constructed a lexicon containing insulting and abusive language. Assigning a weight to each word in the lexicon to indicate its impact, which effectively improved the accuracy of detection. Sood et al. [33] pointed out that sometimes some words appearing in the blacklist would not cause offensiveness. At the same time, they realized that offensive language may hide offensive words through accidental or deliberate spelling errors, so edit distance was used to find potentially offensive words. Finally, a combination of Support Vector Machine (SVM), lexicon, and edit distance is used for classification. The integrated method achieved the highest F1 score compared to the single method. Moreover, since pronouns can be used to avoid detecting offensive terms, Magu et al. [17] added known offensive word counterparts to their lexicon.

In addition, some studies used n-gram features for offensive language detection. Waseem and Hovy [12] compared word n-grams and character n-grams with Logistic Regression (LR) for classification. The results showed that character n-grams performed best. Nobata et al. [19] combined multiple features to detect offensive language. They combine features such as n-gram features obtained by text, linguistic features obtained by statistics, and distributed semantic features obtained by word2vec to detect offensive language. Through experiments on multiple datasets, they found that fusing multiple features greatly improves detection performance compared to single feature. Salminen et al. [34] proposed an improved n-gram method. To improve

---

[1] The dataset will be released on https://github.com/mzx4936/CT-OLD-Dataset
the effectiveness of n-grams, they tried to weight n-grams using word frequency and TF-IDF. The classification was performed using LR, Decision Tree, Random Forest (RF), Adaboost, and SVM. Finally, the n-grams weighted using TF-IDF with SVM achieved 96% of F1 score in binary classification.

2.1.2. Methods of combining text features with user features

Several researchers have combined personal characteristics such as age, gender, and activity history of the users in detecting offensiveness. Dadvar et al. [11] combined textual content and user-based features. They considered the history of user activity, the number of offensive appearances in previously posted content, and the age of the user. The addition of user features improved the F1 score by 4%, with the highest contribution from the number of users' historical offensive language. Unsvåg and Gambäck [13] used user profile information (gender, geographic location, etc.), social network-based user information (number of followers and followees), and activity-based user information (number of favorites, etc.) as user features in combination with n-grams of text for classification. Their work showed that user features could effectively improve the performance of offensive language detection.

Offensive language often comes from users who share common stereotypes [21], while existing offensive language detection methods completely ignore the structure of user communities. Mishra et al. [21] proposed a new method of using community information. They collected user and social relationship information on the basis of the Waseem and Hovy dataset [12]. The users were used as nodes and follower-followee relationships were used as edges to form an undirected social graph. Then the user node embeddings were obtained by node2vec [35] and text features were obtained by n-grams. Finally, the LR classifier was trained for classification, which greatly improved the performance of detection and obtained state of the art at that time. Their work illustrated the importance of using community-based information for offensive language detection.

Classical machine learning methods need to analyze the data and complex features manually, which is probably losing potential features. Therefore, more and more researchers have started to apply deep learning methods to offensive language detection.

2.2. Deep learning-based methods

Deep learning-based methods automatically represent text or users through deep learning models, greatly reducing the reliance on feature engineering. It is divided into the following two types of methods: the method of using text [14, 18, 19, 20, 27, 36, 37, 38], and the method of combining text information with user information [22, 23, 39, 40].

2.2.1. Text-based embedding methods

Recently, deep learning techniques have been applied to offensive language detection task. Such methods usually rely on word embeddings to obtain text representation and then train them by sequential or convolutional models. Badjatiya et al. [14] proposed the methods based on deep learning models and compared them with machine learning methods. The authors compared two word vector initialization methods, random embedding and Glove embedding. After that, the classifiers of CNN, LSTM, and FastText were used to classification. The highest F1 score of deep learning models reached 83.9%, while the highest F1 score of the baseline method using machine learning was 81.6%. In addition, the method combining LSTM and GBDT achieves the highest F1 score of 93.0%. In this method, the word vectors were trained by a deep neural network after random initialization of the word vectors, and then input into GBDT as features for classification. Chen et al. [36] considered that for short texts in social networks, it was difficult to fully learn the information of each word with a highly context-dependent method such as word embeddings. Therefore, they proposed a method based on TF-IDF and CNN, which transformed the 1-dimensional vector obtained by TF-IDF into 2-dimensional matrix, allowing CNN to convolve the word features in a larger field. Compared with randomly initialized word embeddings, the accuracy rate was increased by 9% and the AUC value was increased by 0.12. Rajamanickam et al. [37] proposed a multi-task learning detection model for emotion and offensive language. They used offensive language detection as the primary task and emotion detection as the auxiliary task. The output of the auxiliary task was added to the primary task by encoding the text with BiLSTM. And the information flow was controlled by learnable parameters. When compared on two popular datasets, the method using multi-task learning improved F1 by 1% over the single-task method.

With the rise of large-scale pre-trained language models, BERT [27] has occupied an important position in natural language processing. In the SemEval-2019 Task 6 task, Liu et al. [13] obtained first place in detecting whether a text is offensive (subtask A) by fine-tuning the BERT model for transfer learning only. Dai et al. [19] achieved better results by fine-tuning the BERT model for multi-task learning based on Liu et al. and experimenting on the same dataset.

Although the BERT model has achieved good performance on numerous NLP tasks, when it is applied to non-standard language types such as social network posts, the detection performance may fluctuate greatly due to its chaotic nature. Thus Caselli et al. [38] proposed the HateBERT model, a retraining BERT model for detecting offensive language in English. The model was trained on a large-scale Reddit English comment dataset that included posts from communities banned for being offensive, abusive, and promoting hate speech, containing a to-
tal of 1,492,740 messages. The authors compared with the generic BERT model on three English general-purpose datasets, and the F1 score of HateBERT were all higher than those of generic BERT. Safaya et al. [20] combined BERT and CNN to obtain better detection results than BERT. The authors split the tweets obtained from Twitter, transforming Hashtags into raw text, e.g., “#Some-HashtagText” into “Some Hashtag Text”. Subsequently, the text was split by Wordpiece, a BERT splitter, and the text was fed into a twelve-layer BERT model. The output context embeddings of the last four hidden layers of BERT were provided to the CNN to maximize the semantic knowledge in BERT. Finally, the output of the CNN was used for classification. Experiments were performed on three language datasets (Arabic, Greek, Turkish) and the F1 score was improved by 1% over BERT alone.

2.2.2. Methods of combining text information with user information

The user information provides information that is difficult to obtain at the textual granularity. Several researchers have attempted to combine user information with textual information.

Qian et al. [39] proposed a method for offensive language detection using intra-user and inter-user representations. They collected and analyzed users’ historical tweets to model intra-user tweets. For inter-user tweets, they could be collected from the dataset by locally sensitive hashing (LSH). Adding the inter-user representation and intra-user representation improved the F1 score by 7.1% over the baseline using BiLSTM. Mishra et al. [22] used GCN [10] to learn user embeddings and combined with the bag-of-words representation of the text for offensive language detection. They constructed a heterogeneous graph of users and tweets. The user representation was learned through two layers of GCNs, with the labels of the tweet nodes as the targets of GCN learning. The output of the first layer of GCNs was used as the user embedding. After that, LR was applied to detection with the user embeddings and the bag-of-words representation of the text. This heterogeneous graph representation was able to model the community structure and the linguistic behavior of authors in the community. Therefore, their method achieved an F1 score of 85.4%, refreshing the state of the art of offensive language detection.

Due to the noise of social network texts and the subjectivity of offensive language, the dataset faced shortcomings in construction. There might be an oversimplification in offensive language detection. Therefore, Ribeiro et al. [23] turned to identifying whether users were offensive or not. They collected over 100,000 users and labeled 4,972 of them. For each user, the authors used the number of followers and followees, the number of favorites, centrality measurements (including betweenness, eigenvector centrality, and the in/out degree of each node), and the tweet features obtained by GloVe. The results of the three methods of GradBoost, AdaBoost, and GraphSage for classification were compared. The semi-supervised node embedding algorithm GraphSage was found to outperform the other methods, with a 15% improvement in F1 in the task of detecting offensive users.

In existing studies, n-grams are effective for offensive language detection task. Therefore, classical machine learning methods mostly combine n-grams with other features. With the development of deep learning, especially the emergence of BERT, researchers apply word embeddings to text representations rather than complex feature engineering. During the research on offensive language detection task, the question of how to use user features has been explored. Some researchers applied personal information such as gender and age to improve detection performance. Furthermore, the relationships between users have been taken into consideration. The structure of communities was composed into graphs for graph embedding learning. In the past, user features were obtained after the user model was trained. Then user features are combined with text features. However, the user features obtained by this method may not work well for the offensive language detection task. Meanwhile, the “author bias” in the existing datasets also limits the generalization ability of the model. In our work, we construct the dataset with a method that relies on lexicon indirectly. The user historical behavior feature [11] is used to represent the opinion of users. Finally, the end-to-end model is used to learn user features and text features for classification.

3. Methodology

In this section, we will introduce the data collection and annotation module, text pre-processing module, social graph construction module, and detection module.

The framework of this paper is shown in Figure 1.

1. Data collection and annotation module: This module is responsible for collecting users, relationships, and tweets from Twitter to provide data support for other modules. In this module, a web crawler is developed to collect data from Twitter. Then the data is annotated according to the proposed seven annotation criteria.

2. Social graph construction module: This module processes the collected user and social relationship into a social graph. It takes users as nodes, social relationships as edges, and historical behavioral information as node features to construct a directed isomorphic social graph.

3. Text pre-processing module: This module is used to pre-process the collected tweets. In this module, the emojis in tweets are replaced by words with the same meaning. The hashtag is reduced to words or phrases. In addition, URLs, usernames, etc. are identified uniformly.

4. Detection model: We detect offensive language by an end-to-end model that combines the text representation with community structure. The model is based on the GAT, the BERT, and the attention mechanism.
3.1. Dataset

3.1.1. Collection and annotation

Data discuss. The widely used datasets, including Waseem and Hovy [12], Davidson et al. [10], and Zampieri et al. [1], start collecting through the tweets containing discrimination-related words such as religion, gender, and minority. Such a way of collecting the data would make the dataset have a much higher percentage of offensive language. However, it is difficult to collect tweets with more subtle offensive features. In addition, the datasets such as Waseem and Hovy [12] have severe “author bias”. More than 70% of the sexist tweets originate from two users, and 99% of the racist tweets originate from one user [24]. This bias affects the generalization ability of the model. In this case, Ribeiro et al. [23] suggest collecting users firstly, and then collecting their tweets. Table 1 shows the percentage of offensive language in English language datasets.

| Dataset                     | The number of offensive tweets | The number of tweets | The ratio of offensive tweets | Data type |
|-----------------------------|-------------------------------|----------------------|-------------------------------|-----------|
| Waseem and Hovy [12]        | 5,355                         | 16,914               | 31.66%                        | tweets    |
| Davidson et al. [10]        | -                             | 24,802               | 76.00%                        | tweets    |
| Zampieri et al. [1]         | 4,640                         | 14,100               | 32.91%                        | tweets    |
| Mandli et al. [11]          | 4,456                         | 7,005                | 63.61%                        | tweets    |
| Founta et al. [12]          | -                             | 80,000               | 17.50%                        | tweets    |
| Mishra et al. [21]          | 5,355                         | 16,914               | 31.66%                        | tweets, users, relationships |
| Ours                        | 1,009                         | 12,780               | 7.90%                         | tweets, users, relationships |

Data collection. Based on the above discussions, we want to build a dataset that the distribution of users and tweets is closer to the real-world environment. In most existing datasets, tweets are collected by keyword [23]. Specifically, they first collect tweets using keywords from the lexicon. Then they collect the authors of the tweets and the relationships of the authors. However, they can’t collect the tweets without keywords from the lexicon. In our method, we first collect the users based on the lexicon. Then we collect their relationships and tweets. This method will not lose any tweets.

Finally, our program collected 443,400 tweets, 387,680 users, and 559,735 relationships from Twitter. These tweets were posted from January 2018 to March 2021. Details of data collection are as follows.

(1) Instead of using offensive words, we construct lexicon by topic-related words. Then the users and the social relationships are collected with this lexicon. Specifically, the first-degree and second-degree friends and tweets are collected. Due to a large number of second-degree friends, we only collect tweets from first-degree friends. Finally, raw data includes 443,400 tweets, 387,680 users, and 559,735 relationships.

(2) In the raw data, we randomly select users and expand them to second-degree nodes. Then the users without tweets and social relationships are removed. Meanwhile, we focus on English offensive language detection and filter the others. After the tweets are annotated, only the users with annotated tweets and their relationships are retained. Finally, 1,260 users, 8,877 relationships, and 12,780 tweets are obtained. Among them, the number of offensive tweets is 1,009, accounting for 7.90%.

Annotation. The offensiveness of tweets includes racism, sexism, geography, etc. [2]. The offensiveness of many
tweets is not obvious, so the determination of offensive tweets is a difficult task. Therefore, we need a clear discriminant to guide us in labeling tweets. Our dataset is annotated using the following criteria. The criteria refer to the annotation recommendations of Waseem et al. [12] and Zampieri et al. [1].

(1) Criticizing or satirizing the target without well-founded arguments [12].
(2) Abusing directed at individuals and groups [1].
(3) Blatant misrepresentation of facts or attempting to misrepresent the view of the subject with unfounded claims [12].
(4) Negative stereotyping of an object.
(5) Discrimination against an aspect of the subject.
(6) Criticism of negative facts with a strong subjective negative bias.
(7) Causing discomfort.

We label the tweets offensively and non-offensively, and this labeling is done by three graduate students. Two graduate students label the same tweet, and if the results are different, the third graduate student will make a judgment.

3.1.2. Dataset analysis

To analyze the user distribution and community distribution of the dataset, we perform the following statistics.

User analysis. Among the 1,260 users, the median number of labeled tweets for each user is 11. There are 1,009 offensive tweets in the dataset, and these tweets are posted by 220 users. The distribution is shown in Figure 2.

Community analysis. All users are divided into 22 communities by the algorithms of Blondel et al. [43] and Lambiotte et al. [44]. We show the user distribution of communities with the number of nodes greater than one. The communities are sorted by the proportion of offensive nodes, as shown in Table 3.

Table 3: Statistics on community distribution of offensive users

| Community id | The number of nodes in community | The number of offensive nodes in community | The percentage of offensive nodes in community |
|--------------|----------------------------------|-------------------------------------------|---------------------------------------------|
| 1            | 250                              | 70                                        | 27.97%                                      |
| 2            | 241                              | 15                                        | 6.22%                                       |
| 3            | 219                              | 55                                        | 25.10%                                      |
| 4            | 201                              | 34                                        | 16.90%                                      |
| 5            | 130                              | 15                                        | 11.52%                                      |
| 6            | 88                               | 6                                         | 6.81%                                       |
| 7            | 75                               | 20                                        | 26.63%                                      |
| 8            | 23                               | 1                                         | 4.34%                                       |
| 9            | 21                               | 1                                         | 4.75%                                       |

The Percentage of offensive nodes shows the ratio of offensive nodes in the total nodes of each community. More than 20% of users in communities 1, 3, and 7 have posted offensive language, and they are shown in Figure 3(a). Figure 3(b) shows the users of other communities. The blue nodes are users who have not posted offensive language. And the red nodes are users who have posted offensive language.

Combined with the discussions in dataset analysis, we draw the following conclusions.

(1) Users who have posted offensive language are likely to continue to post offensive language.
(2) Users who post offensive language congregate in some communities. These communities contain the majority of users who post offensive language.

3.2. Social graph building

The online social network is characterized by a complex network in which users are represented as nodes and the follower-followee information as edges [43, 44, 45]. We construct a directed social graph. The direction of the edge is from the follower to the followee. If user $x$ follows user $y$ then an edge will be generated that points from $x$ to $y$. The attributes of the graph nodes only include information about the user historical behavior.

The offensiveness of users is associated with user historical behavior information rather than community structure [22]. Therefore, we use user historical behavior information to represent the offensiveness of users. The ratio of
non-offensive language and offensive language in the training data is counted as user historical behavior information. For example, the user posted three offensive tweets and five non-offensive tweets, and the user historical behavior information is (3, 5). It is used to indicate the opinion of this user to post offensive language. This feature is a node attribute on the social graph. If a user has no tweets, the default value (1, $1e^{-6}$) is used. Training model with this social graph allows the user features to represent the offensiveness of that user.

In this paper, the built graph consists of 1,260 nodes with 8,878 edges. We add self-loops to each node. The nodes have an average degree of 6.66. During the model prediction, the labels of tweets and user historical behavior information of the users are unknown. Therefore, the relevant information of the test set is removed from the social graph. The social graph we used for training and testing is shown in Figure 4. The red node indicates that the user’s tweets are in the test set. The green node indicates that the user’s tweets are in the training set. The purple node indicates that both the training set and the test set contain its tweets.

3.3. Text pre-processing

The processing of this part is directly related to the effectiveness of the offensive language detection task. We process the text from the following aspects.

1. **Emojis.** Tweets contain a large number of emojis. Emojis play an important role in the sentiment tendency of tweets. If the emojis are removed directly, this part of sentiment information will be lost. We use Python’s third-party libraries emoji and ekphrasis to convert emojis into words with similar meanings.

2. **URLs.** To avoid the impact of URLs on semantics of tweet embeddings, we replace all URLs with “http”. For example, “Refinancing during Covid-19. https://t.co/EzjVxwoLq7” into “Refinancing during Covid-19. http”

3. **Hashtag segmentation.** To utilize the textual information in hashtags, they are converted into phrases. This is done using the Python third-party library ekphrasis, e.g. “#SomeHashtagText” into “Some Hashtag Text”.

4. **Other standardization.** Tweets containing user references, email addresses, time, currency, date, etc. may have an impact on the semantics of tweet embeddings. Thus, we convert these uniformly to “<category>”, e.g. “@Kevin” to “<user>”, which uses ekphrasis.
3.4. Then end-to-end model

We design the end-to-end model for offensive language detection. The model consists of graph attention network layer, BERT, attention layer, and feed-forward layer. The learning rate of the GAT layer is set to 1e-2 and the rest are set to 5e-5. The parameters are initialized by the Xavier normal distribution [19]. In addition, the Adam optimizer [20] is used as optimization. The structure of this model is shown as Figure 1.

3.4.1. GAT layer

Graph Attention Networks were proposed by Velicković et al. [26] in 2018. GAT uses the attention mechanism to assign different weights to different nodes, which is suitable for graph representation on social networks. The social graph is represented as $G = (V, E)$, $V$ is the set of user nodes and $E$ is the set of edges. The node features are represented as $h = \{\tilde{h}_1, \tilde{h}_2, \ldots, \tilde{h}_n\}, \tilde{h}_i \in \mathbb{R}^F$, where $n$ is the number of nodes and $F$ is the dimensionality of the node features. We can obtain a high-level representation of the feature $h$ by the following linear transformation.

$$z^{(l)}_i = W^{(l)}h^{(l)}_i$$  \hspace{1cm} (1)

Where $W$ is the weight matrix, $h_i$ is the feature of each node, and $z_i$ is the transformed expression. The $l$ is the index of layer $l$.

For node $i$, the correlation coefficient is calculated by node $i$ and its one-degree neighbor $j$. The nonlinear function LeakyReLU is used as the activation function to obtain the attention scores. After that, the attention scores are normalized by softmax.

$$e^{(l)}_{ij} = \text{LeakyReLU}(\tilde{a}^{(l)^T}(z^{(l)}_i | z^{(l)}_j))$$  \hspace{1cm} (2)

$$a_{ij} = \frac{\exp(e^{(l)}_{ij})}{\sum_{k \in N(i)} \exp(e^{(l)}_{ik})}$$  \hspace{1cm} (3)

Where $a$ is a single-layer feedforward neural network and $e_{ij}$ is the attention scores. The $a_{ij}$ is the normalized attention coefficient.

After that, the attention coefficient is used to perform linear transformation with the corresponding node features. Finally, it is passed through the activation function as the output.

$$h^{(l+1)}_i = \sigma(\sum_{j \in N(i)} a^{(l)}_{ij} h^{(l)}_j)$$  \hspace{1cm} (4)

User attention representations are obtained by the multi-head attention mechanism, which stacks the attention of each head. In addition, to reduce the impact of the network on the original features, we perform a linear transformation on the original features and then stack them with the features calculated by attention. The formula is as follows.

$$h^{(l+1)}_i = [[|K|_{k=1}^{K}\sigma(\sum_{j \in N(i)} a^{(l)}_{ij} W^Kh^{(l)}_j)]][(W^{(l)}h^{(l)}_i)]$$  \hspace{1cm} (5)

This part uses a single-layer graph attention network with eight attention heads and the hidden layer size is set to 768. The output of this layer is represented as $H = \{H_1, H_2, \ldots, H_n\}$, where $n$ is the number of users. For the user $i$, $H_i = \{\tilde{r}^{(1)}_i, \tilde{r}^{(2)}_i, \ldots, \tilde{r}^{(N)}_i, \tilde{r}_i\}$, and $N$ is the number of attention heads. The $\tilde{r}_i$ is the residual, $\tilde{r}_i = h_i$.

3.4.2. BERT

Google released Bidirectional Encoder Representation from Transformer (BERT) [27] based on the Encoder of Transformer [51] in 2018 with improved performance for several NLP tasks. The BERT uses a large-scale corpus for pre-training. The model can be fine-tuned for downstream tasks when performing specific tasks.

This part mainly consists of the BERT Base model. The tweets are fed into the pre-trained BERT model. Then they are passed through 12 layers of self-attention to obtain the embeddings of the context. The attention dropout is set to 0.5 and the hidden dropout is set to 0.1. Meanwhile, texts longer than 512 are truncated. The input texts are denoted as $T = \{t_1, t_2, \ldots, t_s\}$, and $s$ is the number of texts in the current batch. For a text, it is expressed as $t_i = \{t^{(1)}_i, t^{(2)}_i, \ldots, t^{(M)}_i\}$, and $M$ is the length of the text. The output vectors are denoted as $F = \{F_1, F_2, \ldots, F_n\}$, where $F_i = \{\tilde{f}^{(1)}_i, \tilde{f}^{(2)}_i, \ldots, \tilde{f}^{(M)}_i\}$.

3.4.3. Attention layer

The attention mechanism focuses the model’s attention on key information [52]. We use the attention mechanism to fuse the user features with the tweet features. The inputs of this layer are the user features $H$ from the GAT layer and the text features $F$ from the BERT. The users and the texts are not in one-to-one correspondence, but one user corresponds to multiple texts. Since the final task is at the text level, only the user features corresponding to the text are retained in the current batch. Then the user features are appended onto the text features, denoted as $X_i = \{\tilde{f}^{(1)}_i, \ldots, \tilde{f}^{(M)}_i, \tilde{h}^{(1)}_i, \ldots, \tilde{h}^{(N)}_i, \tilde{r}_i\}$. In addition, to reinforce the location information of the sequence, we inject the position encoding according to the method in Transformer [51]. Note that the user features from different attention heads of GCNs use the same position encoding. The resulting features are multiplied by the three weight matrices $W$ to obtain the matrices $Q, K, V$ after performing layer normalization [53], and the attention is computed as follows.

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$  \hspace{1cm} (6)

$$head_i = \text{Attention} \left( QW^Q_i, KW^K_i, VW^V_i \right)$$  \hspace{1cm} (7)

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$  \hspace{1cm} (8)
Where $d_k$ is the dimensionality of matrix $K$ and $W^Q$, $W^K$, $W^V$, $W^O$ is the parameter matrix. The output of multi-headed attention is denoted as $X^i_t = \{\bar{x}_1, \ldots, \bar{x}_M, \bar{x}_{M+1}, \ldots, \bar{x}_{M+N+1}\}$. After this, we add residual linking and layer normalization.

3.4.4. Feed-Forward layer

Following the attention layer is a feed-forward network layer, which contains a ReLU activation function and a linear transformation.

$$FFN(x) = \max(0, x)W + b$$

Where $W$ is the weight of the fully connected layer and $b$ is the bias.

3.4.5. Loss function

For the data imbalance problem, we use Focal Loss \[54\] to mitigate the classification difficulty of imbalanced samples.

$$L_{fl} = \begin{cases} -\alpha (1 - y')^{\gamma} \log y', & y = 1 \\ -(1 - \alpha)y^\gamma \log (1 - y'), & y = 0 \end{cases}$$

Where $\alpha$ and $\gamma$ are both adjustable hyperparameters, we set $\alpha = 0.25$ and $\gamma = 2$. The $y'$ is the model prediction value, which is between $(0 - 1)$. When $y = 1$, $y'$ tends to 1, indicating easy positive samples, and its contribution to the weights tends to 0. When $y = 0$, $y'$ tends to 0, indicating easy negative samples, and its contribution to the weights tends to 0.

4. Experiments and Results

In this section, we evaluate the performance of the proposed method. First, we describe the experimental setup. Second, our method is compared with the current state-of-the-art methods. Third, we evaluate the effectiveness of the modules in CT-OLD. Then we compare the performance of feature initialization methods. Finally, we explore the differences introduced by different social graph modeling methods.

4.1. Experimental setup

To evaluate the effectiveness of the end-to-end detection method proposed in this paper for offensive language detection, we conducted a series of experiments. Our method uses a batch size of 64, and the rest of the methods are executed according to the original papers. All experiments are conducted on 12,780 samples, 1,260 users, and 8,877 relations. The dataset is divided 7:3 by tweets and the average results are calculated by 10 independent replications. All experiments are executed on a server with 128G RAM, Intel Xeon Gold 6130 2.10GHz, and NVIDIA Tesla V100. In experiments, all models are trained with maximum 20 epochs. Besides, the early stop strategy is used if the F1 score doesn’t increase in five continuous epochs.

4.1.1. Baseline analysis

To perform performance evaluation, we conducted a series of baseline studies of these state-of-the-art methods for offensive language detection as follows. BERT. Liu et al. \[18\] won the first place in subtask A of OffensEval-2019 with the fine-tuned BERT model. The
text embeddings are obtained by the BERT model and then classified by linear layer.

**BERT+CNN.** Safaya et al. [20] combined BERT with CNN for offensive language detection. Their method ranked in the top four on all three languages in subtask A of OffensEval-2020. BERT+CNN learns text representations through the BERT model. Then the output of the last four layers of BERT is fed into CNN networks with different size filters.

**LR+AUTH.** LR+AUTH [21] is the first work to combine community structure and texts for offensive speech detection. First, social relationships are constructed as social graphs. Then user representations are learned by node2vec. For each tweet, the user features are appended to the n-gram features of the tweet, and the LR classifier is trained.

**LR+GCN.** LR+GCN [22] models users and tweets to construct a heterogeneous graph. It uses bags of words as the feature of users. This method uses two layers of GCNs to train by minimizing the cross-entropy of the labeled nodes in the graph. The output of the first hidden layer of the trained GCNs is used as the user embeddings. For each tweet, user features are appended onto the character n-gram features of the tweet for training the LR classifier. State-of-the-art results were obtained at that time.

The LR+AUTH code is publicly available and directly used in our experiments. The rest of the baseline is our implementation based on the original paper.

### 4.1.2. Performance metrics

To quantify model effects, we show AUC, Accuracy, Macro Recall, Macro Precision, and Macro F1 score.

\[
AUC = \frac{\sum_{i \in \text{positive class}} \text{rank}_i - \frac{M(M+1)}{2}}{M \cdot N} \tag{11}
\]

Where \(\sum_{i \in \text{positive class}} \text{rank}_i\) is the sum of the positive samples ranked by probability score from smallest to largest, \(M\) is the number of positive samples, and \(N\) is the number of negative samples. When the probability scores are the same, the rank value is the average of the rank of the same probability scores.

For the other four metrics, the confusion matrix is calculated first, and for each classification, the true positive (TP), the false negative (FN), the true positive (FP), and the true negative (TN) are calculated as follows:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}
\]

\[
\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{13}
\]

\[
\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{14}
\]

The subscript \(i\) of each metric represents the \(i\)-th category, and the number of categories is represented by \(L\). After calculating the Recall, Precision, and F1 for each category, the average value is used to obtain the macro value. The formula for calculating Macro-F1 is given below, and the rest of the metrics are the same.

\[
\text{Macro-}F1 = \frac{1}{L} \sum_{i=1}^{L} F1_i \tag{15}
\]

### 4.2. Evaluation of the proposed detection approach

#### 4.2.1. Comparison with baseline method

| Model          | Accuracy | Precision | Recall | F1   |
|----------------|----------|-----------|--------|------|
| LR+AUTH [21]   | 93.87    | 82.97     | 71.57  | 75.80|
| LR+GCN [22]    | 95.62    | 89.70     | 79.23  | 83.49|
| BERT [18]      | 95.41    | 83.70     | 79.23  | 83.49|
| BERT+CNN [20]  | 95.64    | 84.65     | 86.51  | 86.51|
| CT-OLD (Ours)  | **96.92**| **89.83** | **90.06**| **89.94**|

The code of this baseline is taken from [https://github.com/pushkarmishra/AuthorProfilingAbuseDetection](https://github.com/pushkarmishra/AuthorProfilingAbuseDetection). Figure 6: ROC-AUC of baselines and CT-OLD
BERT-CNN has further improvement, but it is not significant.

CT-OLD obtains the first place in all evaluation metrics, and the F1 score was 3.43% higher than the second. For the ROC curve, we find that CT-OLD’s curve covers all other methods. This proves that our method has a great advantage in the offensive language detection task.

4.2.2. Effect of different training ratios

This experiment shows the performance of the baseline method with CT-OLD at different training ratios from 0.1 to 0.9.

It can be seen from Figure 7 that different training ratios have an impact on all methods. Meanwhile, each performance metric gradually increases with the increase of the training ratio. The difference between BERT and BERT-CNN in different training ratios is small. The F1 score of LR+AUTH fluctuates the most. And it is lower than the other methods at all ratios.

The F1 score of CT-OLD increases more obviously when the training ratio is from 0.1 to 0.4. In addition, the difference between the F1 scores of CT-OLD and BERT is also increasing. This is because when the training set is small, the node features are relatively sparse on the social graph. With increasing the training data, the opinion of users to post offensive language is learned obviously. As a whole, CT-OLD achieves a significant advantage in all metrics.

4.2.3. Detection model ablation tests

To explore the influence of the graph attention network layer, the BERT, the residuals of the GAT layer, the multi-headed attention of the GAT layer, and the attention layer used in the CT-OLD, we perform ablation tests. Specifically, we remove only one of the above modules from the CT-OLD and show the performance of the model. The results are shown in Table 5.

Table 5: The impact of modules on detection performance

| Model                     | AUC  | Accuracy | Precision | Recall | F1     |
|---------------------------|------|----------|-----------|--------|--------|
| CT-OLD\GAT                | 0.9775 | 95.83    | 87.02     | 84.90  | 85.92  |
| CT-OLD\BERT               | 0.9225 | 92.44    | 75.48     | 85.33  | 79.31  |
| CT-OLD\Residual of GAT    | 0.9802 | 96.35    | 88.18     | 87.75  | 87.96  |
| CT-OLD\Multi-head Attention of GAT | 0.9824 | 96.50    | 87.63     | 90.40  | 88.95  |
| CT-OLD\Attention Layer    | 0.9852 | 96.56    | 88.13     | 89.86  | 88.97  |
| CT-OLD                    | 0.9853 | 96.92    | 89.83     | 90.06  | 89.94  |

We find that the removal of any module will reduce the model performance. All modules contribute to the effectiveness of the CT-OLD. The CT-OLD\GAT removes the graph attention network layer, with similar results to the two BERT-based methods used as baselines. The CT-OLD\BERT removes the text embeddings obtained by BERT. The connection between the GAT and our detection target is only the user historical behavior information. Meanwhile, it only detects the granularity of the author. In such a coarse-grained detection, the F1 score has reached 79.31%, exceeding the LR+AUTH. The GAT uses community structure with user behavior information rather than textual content. So, the information it learns is different from the text-based method. In addition, the text-based BERT captures semantic information well. Therefore, the end-to-end detection method combining the two features significantly improves the detection performance.

In the remaining four experiments, the CT-OLD\Residual of GAT removes the residual links from the graph attention network and has the most degraded performance. This is because the residual connection enhances the influence of the original features.

4.2.4. Comparison of initialization of user features

The node features are an important factor for graph representation learning. However, for users whose tweets exclude in the training set, their initial features are not available during the training. This requires feature initialization for this subset of users. Therefore, we consider four initialization methods.

(1) All-0 initialization. The number of tweets for both categories is unknown. It is set to [0, 0] by default.

(2) All-1 initialization. The probability of a user posting tweets in both categories is the same. Set to [1, 1].

(3) Average initialization. For each category of tweets, the average number of user posts is used as the initial value.

(4) Non-offensive initialization. There are a few users who post offensive language in social networks. And it is assumed that users only post non-offensive tweets. It is set to [1, 1e−6].

In addition to the default case of 0.7 training ratio, we also show the detection performance in the sparse data case of 0.1 training ratio. Table 6 shows the performance of different feature initialization methods.

Table 6: Performance of different feature initialization methods

| training ratio | Model    | Accuracy | Precision | Recall | F1     |
|----------------|----------|----------|-----------|--------|--------|
| Sparse data    |          |          |           |        |        |
| All-0          | 96.45    | 87.05    | 91.22     | 89.00  |
| All-1          | 96.40    | 86.79    | 91.34     | 88.90  |
| Avg            | 96.61    | 87.70    | 91.31     | 89.40  |
| Non-off        | 96.92    | 89.83    | 90.06     | 89.94  |

The effectiveness of user historical behavior information decreases in the case of sparse data. The detection performance is more dependent on the user feature initialization. The experiments show that non-offensive initialization performs the best. By default, only a small part of
users rely on feature initialization. The performance of the four methods is similar. The difference between the highest and lowest F1 score is only less than 1%.

4.2.5. Comparison of social graph modeling

In the past, modeling of community information could be done by computing node embeddings through the network structure of the communities [21]. This method is able to capture the structure of communities but ignores the association with offensive language detection.

Mishra et al. [22] built user features through the n-gram features of tweets, which effectively combined community structure and text information. In this method, community structure and linguistic features are captured together. The user features obtained by this method contain all the information of the tweets posted by the user. However, the user representations of this method depend on the quality of the tweet representations deeply.

We compare three social graph modeling methods for users.

(1) Using the modeling method of Mishra et al. [22], but we only model the users. The user features are constructed by binary bag-of-words representations of the user’s posted tweets. This method is called the bow feature.

(2) We tag users based on whether they post offensive language. If the user has posted offensive language, the tag is 1, otherwise, it is 0. This method we call the hard feature.

(3) In our proposed method, we use the user historical behavior information as user features. This method is called the soft feature.

Table 7 shows the comparison in different methods of social graph modeling. Among the three methods of GAT, the F1 score using the soft feature is the highest, the bow feature is the second-highest, and the hard feature is the lowest. This may be because the hard feature contains too little information. As long as users have posted offensive language, their tweets are judged to be offensive. The bow feature contains complete information, but large-scale features also have more uncertainty during training. The soft feature has more constraints than the bow feature and more complete information than the hard feature.

The soft feature still gets the highest F1 score among the three CT-OLD methods, while the difference between the bow feature and hard feature is not obvious. The reason for this variation is that the graph learning module using the hard feature and soft feature has a clear purpose to learn the representation of users to post offensive language. The bow feature, on the other hand, uses raw features of text embeddings with complex features, resulting in difficulties in learning effective information. There-
fore, the hard feature has stronger performance than the bow feature. However, the hard feature will be mislabeled if the training set excludes the offensive tweets of a user and the test set does, affecting the detection performance. The soft feature based on user historical behavior information has the advantages of others and achieves better performance.

5. Conclusion

In this paper, we proposed an end-to-end method for detecting offensive language with graph attention networks and BERT. The model captures community structure and text information at the same time and fuses them by attention mechanisms. Specifically, we used user historical behavior information to represent user opinion. This information is a node attribute on the social graph that associates the social graph with the offensiveness of the user. We evaluated the effectiveness of our method on the dataset we constructed. The experimental results showed that the proposed method is effective for detecting offensive language, and it has better performance than other methods.

In online social networks, the generalization ability of the offensive language detection model that learns community structure and text at the same time will exceed that of the text-based model. But it is difficult to obtain more datasets with different user distribution and community structure. In the future, we hope to further investigate the impact of social graphs on generalization ability.

References

[1] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, Predicting the type and target of offensive posts in social media, in: Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, USA, 2019, pp. 1415–1420.
[2] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys 51 (4) (2018) 1–30.
[3] E. Mosca, M. Wich, G. Groh, Understanding and interpreting the impact of user context in hate speech detection, in: Proceedings of the 9th International Workshop on Natural Language Processing for Social Media, Online, 2021, pp. 91–102.
[4] Z. Waseem, T. Davidson, D. Warmsley, I. Weber, Understanding abuse: A typology of abusive language detection subtasks, in: Proceedings of the 1st Workshop on Abusive Language Online, Vancouver, Canada, 2017, pp. 78–84.
[5] P. Kapil, A. Ekbal, A deep neural network based multi-task learning approach to hate speech detection, Knowledge-Based Systems 210 (2020) 106458.
[6] A. Sheth, V. L. Shalin, U. Kursuncu, Defining and detecting toxicity on social media: context and knowledge are key, Neu-rocomputing (2021) Online.
[7] R. Magu, K. Joshi, J. Luo, Detecting the hate code on social media, in: Proceedings of the 11th International Conference on Web and Social Media, Montréal, Canada, 2017, pp. 608–611.
[8] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, N. Bhamidipati, Hate speech detection with comment embeddings, in: Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 2015, pp. 29–30.
[9] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, Y. Chang, Abusive language detection in online user content, in: Proceedings of the 25th International Conference on World Wide Web, Montréal, Canada, 2016, pp. 145–153.
[10] T. Davidson, D. Warmsley, M. Macy, I. Weber, Automated hate speech detection and the problem of offensive language, in: Proceedings of the 11th International Conference on Web and Social Media, Montréal, Canada, 2017, pp. 512–515.
[11] M. Dadvar, D. Trieschnigg, R. Ordelman, F. de Jong, Improving cyberbullying detection with user context, in: Proceedings of the 35th European Conference on Information Retrieval, Moscow, Russia, 2013, pp. 693–696.
[12] Z. Waseem, D. Hovy, Hateful symbols or hateful people? predictive features for hate speech detection on twitter, in: Proceedings of the 14th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, USA, 2016, pp. 88–93.
[13] E. F. Unsväg, B. Gambäck, The effects of user features on twitter hate speech detection, in: Proceedings of the 2nd Workshop on Abusive Language Online, Brussels, Belgium, 2018, pp. 75–85.
[14] P. Badjatiya, S. Gupta, M. Gupta, V. Varma, Deep learning for hate speech detection in tweets, in: Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, 2017, pp. 759–760.
[15] J. H. Park, P. Fung, One-step and two-step classification for abusive language detection on twitter, in: Proceedings of the 1st Workshop on Abusive Language Online, Vancouver, Canada, 2017, pp. 41–45.
[16] S. Yoon, J. Shin, K. Jung, Learning to rank question-answer pairs using hierarchical recurrent encoder with latent topic clustering, in: Proceedings of the 16th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, Louisiana, 2018, pp. 1575–1584.
[17] P. Le-Hong, Diacritics generation and application in hate speech detection on vietnamese social networks, Knowledge-Based Systems 233 (2021) 107504.
[18] P. Liu, W. Li, L. Zou, Nali at senseval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Montreal, USA, 2019, pp. 87–91.
[19] W. Dai, T. Yu, Z. Liu, P. Fung, Kungfupanda at senseval-2020 task 12: Bert-based multi-tasklearning for offensive language detection, in: Proceedings of the 14th Workshop on Semantic Evaluation, Online, 2020, pp. 2060–2066.
[20] A. Safaya, M. Abdiellatif, D. Yuret, Kuisail at senseval-2020 task 12: Bert-cnn for offensive speech identification in social media, in: Proceedings of the 14th Workshop onSemantic Evaluation, Online, 2020, pp. 2054–2059.
[21] P. Mishra, M. Del Tredici, H. Yannakoudakis, E. Shutova, Author profiling for abusive detection, in: Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, USA, 2018, pp. 1088–1098.
[22] P. Mishra, M. Del Tredici, H. Yannakoudakis, E. Shutova, Abusive language detection with graph convolutional networks, in: Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, USA, 2019, pp. 2145–2150.
[23] M. H. Ribeiro, P. H. Calais, Y. A. Santos, V. A. Almeida, W. Meira Jr, Characterizing and detecting hateful users on twitter, in: Proceedings of the 12th International Conference on Web and Social, Stanford, USA, 2018, pp. 676–679.
[24] M. Wiegand, J. Ruppenhofer, T. Kleinbauer, Detection of abusive language: the problem of biased datasets, in: Proceedings of the 17th conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, USA, 2019, pp. 602–608.
[25] A. Arango, J. Pérez, B. Poblete, Hate speech detection is not
as easy as you may think: A closer look at model validation, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 2019, pp. 45–54.

[20] P. Velicković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: Proceedings of the 6th International Conference on Learning Representations, Vancouver, Canada, 2018, pp. 1–12.

[21] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 17th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, USA, 2019, pp. 4171–4186.

[22] W. Tang, G. Luo, Y. Wu, L. Tian, X. Zheng, Z. Cai, A second-order diffusion model for influence maximization in social networks, IEEE Transactions on Computational Social Systems 6 (4) (2019) 702–714.

[23] W. Tang, L. Tian, X. Zheng, G. Luo, Z. He, Susceptible user search for defending opinion manipulation, Future Generation Computer Systems 115 (2021) 531–541.

[24] W. Tang, B. Hui, L. Tian, G. Luo, Z. He, Z. Cai, Learning disentangled user representation with multi-view information fusion on social networks, Information Fusion 74 (2021) 77–86.

[25] W. Tang, X. Xu, G. Luo, Z. He, K. Zhan, Budgeted persuasion on user opinions via varying susceptibility, in: Proceedings of the 39th IEEE International Performance Computing and Communications Conference, Austin, USA, 2020, pp. 1–8.

[26] A. H. Razavi, D. Inkpen, S. Uritsky, S. Matwin, Offensive language detection using multi-level classification, in: Proceedings of the 23rd Canadian Conference on Artificial Intelligence, Ottawa, Canada, 2010, pp. 16–27.

[27] S. O. Sood, J. Antin, E. Churchill, Using crowdsourcing to improve profesion detection, in: Proceedings of the 21st AAAI Spring Symposium Series: Wisdom of the Crowd, Stanford, USA, 2012, pp. 69–74.

[28] J. Salminen, H. Almerekhi, M. Milenković, S.-g. Jung, J. An, H. Kwak, B. J. Jansen, Anatomy of online hate: developing a taxonomy and machine learning models for identifying and classifying hate in online news media, in: Proceedings of the 12th International Conference on Web and Social Media, Stanford, USA, 2018, pp. 330–339.

[29] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, USA, 2016, pp. 855–864.

[30] J. Chen, S. Yan, K.-C. Wong, Verbal aggression detection on twitter comments: Convolutional neural network for short-text sentiment analysis, Neural Computing and Applications 32 (15) (2020) 1089-1098.

[31] S. Rajamakickam, P. Mishra, H. Yannakoudakis, E. Shutova, Joint modelling of emotion and abusive language detection, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 2020, pp. 4270–4279.

[32] T. Caselli, V. Basile, J. Mitrovic, M. Grauhter, Hatebert: Retraining bert for abusive language detection in english, arXiv preprint arXiv:2010.12472.

[33] J. Qian, M. ElSherief, E. Belding, W. Y. Wang, Leveraging intra-user and inter-user representation learning for automated hate speech detection, in: Proceedings of the 16th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, Louisiana, 2018, pp. 118–123.

[34] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 2017, pp. 1–14.

[35] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandli, A. Patel, Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages, in: Proceedings of the 11th Forum for Information Retrieval Evaluation, Kolkata, India, 2019, pp. 14–17.

[36] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, N. Kourtellis, Large scale crowdsourcing and characterization of twitter abusive behavior, in: Proceedings of the 12th International Conference on Web and Social Media, Stanford, USA, 2018, pp. 491–500.

[37] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (10) (2008) 10008.

[38] R. Lambiotte, J.-C. Delvenne, M. Barahona, Laplacian dynamics and multiscale modular structure in networks, arXiv preprint arXiv:0812.1770.

[39] R. Tang, S. Jiang, X. Chen, H. Wang, W. Wang, W. Wang, Interlayer link prediction in multiplex social networks: An iterative degree penalty algorithm, Knowledge-Based Systems 194 (2020) 105598.

[40] R. Tang, X. Chen, C. Wei, Q. Li, W. Wang, H. Wang, W. Wang, Interlayer link prediction based on multiple network structural attributes, Computer Networks 203 (2022) 108651.

[41] R. Tang, Z. Miao, S. Jiang, X. Chen, H. Wang, W. Wang, Interlayer link prediction in multiplex social networks based on multiple types of consistency between embedding vectors, IEEE Transactions on Cybernetics (2021) 1–14.

[42] C. Baziotis, N. Pelekis, C. Douleridis, Datastories at senseval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis, in: Proceedings of the 11th International Workshop on Semantic Evaluation, Vancouver, Canada, 2017, pp. 747–754.

[43] X. Ghorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Chia Laguna Resort, Italy, 2010, pp. 249–256.

[44] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations, San Diego, USA, 2015, pp. 1–15.

[45] C. Baziotis, N. Pelekis, C. Douleridis, Datastories at senseval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis, in: Proceedings of the 11th International Workshop on Semantic Evaluation, Vancouver, Canada, 2017, pp. 747–754.

[46] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, N. Kourtellis, Large scale crowdsourcing and characterization of twitter abusive behavior, in: Proceedings of the 12th International Conference on Web and Social Media, Stanford, USA, 2018, pp. 491–500.

[47] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, N. Kourtellis, Large scale crowdsourcing and characterization of twitter abusive behavior, in: Proceedings of the 12th International Conference on Web and Social Media, Stanford, USA, 2018, pp. 491–500.