Detection and Analysis of Hateful Users on Twitter

A dissertation report submitted in fulfilment of the requirements for the degree of Bachelor of Technology

by

Uditi Arora (2016UCP1415)

Arnav Loonker (2016UCP1016)

Rishabh Kalakoti (2016UCP1098)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

April 2020

Certificate

We,

UDITI ARORA (2016UCP1415)

ARNAV LOONKER (2016UCP1016)

RISHABH KALAKOTI (2016UCP1098)

Declare that this thesis titled, “Detection and Analysis of Hateful User on Twitter” and the work presented in it are our own. I confirm that:

- This project work was done wholly or mainly while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT).

- Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely our own work.

- We have acknowledged all main sources of help.

Signed: ______________________________

Date:

Dr. Mahipal Jadeja

Assistant Professor

April 2020 Department of Computer Science and Engineering

Malaviya National Institute of Technology Jaipur

Abstract

Nowadays, hate speeches on social media have become a severe problem that do unpredictable harm to people, especially young students and professionals. To solve this problem it is crucial to correctly find hateful users who deliver hateful speeches and can prove to be harmful for the online society. Most current approaches to characterize and detect hate speech focus on content posted in Online Social Networks. They face shortcomings to collect and annotate hateful speech due to the incompleteness and noisiness of OSN text and the subjectivity of hate speech. These limitations are often aided with constraints that oversimplify the problem, such as considering only tweets containing hate-related words. In this work we partially address these issues by shifting the focus towards users. This works presents a user-centric view of hate speech, paving the way for better detection methods and understanding. We use a Twitter dataset of 100, 386 users along a sample of 4, 972 manually annotated nodes.We observe that hateful users are densely connected, and thus formulate the hate speech detection problem as a task of semi-supervised learning over a graph, exploiting the network of connections on Twitter. We examine the difference between user activity patterns, the content disseminated between hateful and normal users, and network centrality measurements in the sampled graph. From observations and previous research it has been shown that twitter retweet graph is a small-world model. We used NetworkX to visualise the graph and preprocess data. We examine the difference between user activity patterns, the content disseminated between hateful and normal users, and network centrality measurements in the sampled graph. We start from graph networks and implement and evaluate semi-supervised GraphSAGE. We also explore different versions of Graph Convolutional Networks (GCNs).

Acknowledgements

Foremost, we would like to thank our guide, Dr. Mahipal Jadeja, for his continuous guidance and support. He has held our hand throughout the process of this project and we couldn’t have asked for anything more. We would also like to thank our HoD, Dr. Pilli Emmanuel Shubhakar, and the department of Computer Science and Engineering for giving us the opportunity of doing this project.

Uditi Arora	Arnav Loonker	Rishabh Kalakoti
(2016UCP1415)	(2016UCP1016)	(2016UCP1098)

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

The importance of understanding hate speech in Online Social Networks (OSNs) is manifold. The trade-off between banning such behavior from platforms and not censoring dissenting opinions is a major societal issue. This scenario has motivated work that aims to understand and detect hateful content by creating representations for tweets or comments in an OSN and then classifying them as hateful or not, often drawing insights on the nature of hateful speech. However, in OSNs, the meaning of such content is often not self-contained, referring, for instance, to some event which just happened and the texts are packed with informal language, spelling errors, special characters and sarcasm . Besides that, hate speech itself is highly subjective, reliant on temporal, social and historical context, and occurs sparsely.

These problems, although observed, remain unaddressed. Consider the tweet:

Timesup, yall getting w should have happened long ago

Which was in reply to another tweet that mentioned the holocaust. Although the tweet, whose author's profile contained white-supremacy imagery, incited violence, it is hard to conceive how this could be detected as hateful with only textual features. Furthermore, the lack of hate-related words makes it difficult for this kind of tweet to be sampled.

The data in posts, tweets or messages are not the only signals we may use to study hate speech in OSNs. Most often, these signals are linked to a profile representing a person or organization. Characterizing and detecting hateful users shares much of the benefits of detecting hateful content and presents plenty of opportunities to explore a richer feature space. Furthermore, in a practical hate speech guideline enforcement process, containing humans in the loop, it is natural that user profiles will be checked, rather than isolated tweets. The case can be made that this wider context is sometimes needed to define hate speech, such as in the example, where the abuse was made clear by the neo-nazi signs in the user's profile. Analyzing hate on a user-level rather than content-level enables our characterization to explore not only content, but also dimensions such as the user's activity and connections. Moreover, it allows us to use the very structure of Twitter's network in the task of detecting hateful users.

Analyzing hateful users rather than content is also attractive because other dimensions may be explored, such as the user's activity and connections in the network. For example, in Twitter, it is possible to see the number of tweets, followers, and favorites a user has. It is also possible to extract influence links among users who retweet each other, analyzing them in a larger network of influences. This allows us to use network-based metrics, such as betweenness centrality and also to analyze the neighborhood of such users. Noticeably, although several studies characterize hateful speech in text, no study that the authors are aware of focuses on the dimension of hateful users in OSNs.

Our results show that hateful users tweet more and within smaller intervals, and favorite other tweets significantly more than the normal ones. They also are more negative according to lexicon based sentiment analysis and use more swear words. Hateful users have follow more people per day than normal ones, and use vocabulary related to categories such as hate, anger, shame, violence and

terrorism less frequently. Also, the median hateful user has higher network centrality according to several metrics, contradicting the "lone wolf" behavior often associated with the practice.
We evaluated several different graph based node classification algorithms. First we used the GraphSAGE semi-supervised classifier. We also used variations of GCNs, out of which BGCN gave the best results.

Chapter 2 - Important Terms and Concepts

2.1 Terms used in graphs plotted

2.1.1 Box Plots

A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.[1]

2.1.2 KDE Plots

KDE Plot described as **Kernel Density Estimate** is used for visualizing the Probability Density of a continuous variable. It depicts the probability density at different values in a continuous variable. We can also plot a single graph for multiple samples which helps in more efficient data visualization.[2]

2.1.3 Network Centrality.

In graph theory and network analysis, indicators of centrality identify the most important vertices within a graph. Applications include identifying the most influential person in a social network, key infrastructure nodes in the Internet or urban networks, and super-spreaders of disease.[3]

2.1.4 Eigenvectors

In linear algebra, an eigenvector or characteristic vector of a linear transformation is a nonzero vector that changes at most by a scalar factor when that linear transformation is applied to it. The corresponding eigenvalue is the factor by which the eigenvector is scaled.[4]

2.1.5 Network Connectivity

Network connectivity describes the extensive process of connecting various parts of a network to one another, for example, through the use of routers, switches and gateways, and how that process works.[5]

2.2 Terms Related to Dataset

2.2.1 Hateful Users

We define "hateful user" and "hate speech" according to Twitter's guidelines. For the purposes of this paper, "hate speech" is any type of content that 'promotes violence against or directly attacks or threatens other people on the basis of race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or disease." (Twitter 2017) On the other hand, "hateful user” is a user that, according to annotators, endorses such type of content.[6]

2.2.2 Retweet Graph

The retweet graph G is a directed graph G = (V, E) where each node u ∈ V represents a user in Twitter, and each edge (u1, u2) ∈ E implies that the user u1 has retweeted u2. Previous work suggests that retweets are better than followers to judge users’ influence. As influence flows in the opposite direction of retweets, we invert the graph’s edges.[6]

2.2.3 Offensive Language

We employ Waseem et al. definition of explicit abusive language, which defines it as language that is unambiguous in its potential to be abusive, for example language that contains racial or homophobic slurs. The use of this kind of language doesn’t imply hate speech, although there is a clear correlation.[6]

Chapter 3 - Literature Survey

3.1 Belief Propagation

In general, people on social media are closely connected with people of similar interests and backgrounds. Based on this assumption, the belief propagation algorithm was implemented. It is a baseline model which is structure based. It gave an accuracy of 0.8282. However, the labelled dataset which was used was highly unbalanced. 90% of the data were normal users. Due to this there is a need to incorporate node attributes as we cannot get much information about hateful users from such a small percentage of data.[7]

3.2 GraphSAGE

Implementing a semi-supervised classifier using GraphSAGE operators for hateful user node classification with Pytorch geometric. GraphSAGE is a general inductive framework that makes use of node features and graph information to efficiently generate node embeddings for previously unseen data. With these generated embeddings, a binary classification was conducted on each node for hateful user detection. The main steps of the embedding generation process included aggregation from neighbors with sampling, update and normalization with multi-layer perceptron plus skip connection. GraphSAGE-0 performed even worse than the baseline model of belief propagation with an accuracy of 0.8028. 25 neighbours of a labeled node were sampled for GraphSAGE. Using this variation took the accuracy up to 0.8489. Across all models, GraphSAGE with 25 neighbour sampling performs best among all models. [7]

3.3 Graph Attention

Experimenting with GAT(Graph Attention) operator with only one layer and one head, making use of an attention mechanism which benefits from biasing on more relevant parts of the input. Experiment settings were similar to GraphSAGE. 5-folds cross validation for model training and evaluation was applied for the GAT classifier. It performs slightly worse than GraphSAGE as the accuracy drops to 0.8081. The performance decreased because the experiments were conducted using one head, and it's probably not powerful enough to learn many useful features. [7]

3.4 LoNGAE

Local Neighborhood Graph Autoencoder is used for graph link prediction. LoNGAE performs encoding and decoding transformations for link prediction. The autoencoder is learned with symmetrically shared parameters. In order to use the model for node classification, a feature set had to be added manually with a Softmax activation layer. The input was both labeled data and sampled unlabeled data. The model was using labeled and sampled data. Different portions of unlabeled 1-hop data for training. The results show that the accuracy was 0.8480. It was also observed that the performance goes down as soon as the number of unlabeled nodes go above 1000. This is probably because with more random unlabeled data, this learning algorithm fails to capture certain feature relationships and handle some of the uncertainty during training and thus gives worse performance.[7]

3.5 GraphMix

GraphMix is a regularization technique for graph neural network based semi-supervised node classification. It trains a graph neural network and a fully-connected network via parameter sharing, interpolation-based regularization and self- predicted targets. It gave an accuracy of 0.884. It was observed with increasing time the performance dropped because the network became too big.[7]

Chapter 4 -Related Work

A lot of analysis on Twitter has been conducted on the Twitter follower graph, which connects users with edges representing a user follows or is followed by another user. Label propagation over lexical links and Twitter follower graph for polarity classification has been applied. Conclusions drawn in these works gave inspiration and reference to graph analysis. (Bild et al., 2015[8]) conducted the first study on the Twitter retweet graph, where each node represents a Twitter user and each edge represents a retweet. They showed that the retweet graph is a small-world model and conducted spammer detection on it using a decision tree classifier. (Ribeiro et al., 2017[9]) created a new retweet dataset and characterized users with attributes like sentiment score on tweets and number of followers, and came up with several findings. For example, median hateful users have higher network centrality than the median normal users. (Yang et al., 2012[10]) proposed a variant of the HITS algorithm and used it for valuable post classification. (Tran, 2018[11]) presented a new autoencoder architecture that can learn a joint representation of local graph structure and node features for multi-task link and semi-supervised node classification. (Verma et al., 2019[12]) proposed a regularization method for graph neural network based semi-supervised node prediction, in which a fully connected network is trained jointly with a graph neural network via parameter sharing, interpolation-based regularization and self-predicted targets. (Yipeng et al., 2019[7]) used SNAP and NetworkX to analyse the graph structure. They implemented and evaluated different graph-based node classification algorithms. They also delivered in-depth graph and model analysis on hateful user prediction tasks. Since our dataset contains a large amount of unlabeled data, we experiment with the above semi-supervised learning algorithms for in-depth exploration and learning of retweet graph dataset. We also try to compare performances of different models using Matplotlib python library.

Chapter 5 - Proposed Method

To solve this problem it is crucial to correctly find hateful users who deliver hateful speeches and can prove to be harmful for the online society. Most current approaches to characterize and detect hate speech focus on content posted in Online Social Networks. They face shortcomings to collect and annotate hateful speech due to the incompleteness and noisiness of OSN text and the subjectivity of hate speech. These limitations are often aided with constraints that oversimplify the problem, such as considering only tweets containing hate-related words. In this work we partially address these issues by shifting the focus towards users. This work presents a user-centric view of hate speech, paving the way for better detection methods and understanding. We use a Twitter dataset of 100, 386 users along a sample of 4, 972 manually annotated nodes. We observe that hateful users are densely connected, and thus formulate the hate speech detection problem as a task of semi-supervised learning over a graph, exploiting the network of connections on Twitter. We start by analysing the data in the dataset. We use Matplotlib and NetworkX to plot graphs and draw insights from them. We then moved on to using semi-supervised graph network models which showed a huge improvement in performance. We also compare performance of different models in order to have a better insight about which model works best.

Chapter 6 - Experimental Results

6.1 Results of Experimental Data Analysis

Through experimental data analysis, we try to analyse how hateful and normal users differ with respect to their activity, vocabulary and network centrality. We also try to compare the neighbours of hateful and normal users. It is assumed that neighbours are likely to show similarities. Thus, we compare proximity of pairs and inspect its effect on the results. This is done in order to increase the robustness of the analysis.

6.1.1 Number of users in each group

Table 1. Number of users in each group

Type of User	Number of Users
Hateful Users	544
Normal Users	4427
Hateful Neighbours	3471
Normal Neighbours	33564
Banned Users	668
Active Users	99718

As can be seen, the data is highly biased. The number of hateful users is very less as compared to normal users. which is also the case in real time situations but this creates an issue when training models. This leads to a drop in precision and recall values even when the accuracy values are high. The solution is to train models which learn neighbourhood characteristics.

6.1.2 Date of creation of accounts of hateful users

As can be seen in the figure below, hateful users' accounts were created later than normal users. A possible reason behind this can be that hateful users are banned more often due to Twitter's guidelines infringement. This resonates with existing methods for detecting fake accounts. It also observed that even 1-neighbours of hateful users have newer accounts which supports common neighbour behaviour assumption. This gives important insights into the way hateful users behave. It also shows that banning accounts helps only to an extent as hateful users tend to make new fake accounts. The constant ranting is likely to continue even after banning the users. In today's scenario, trolls work the same way.

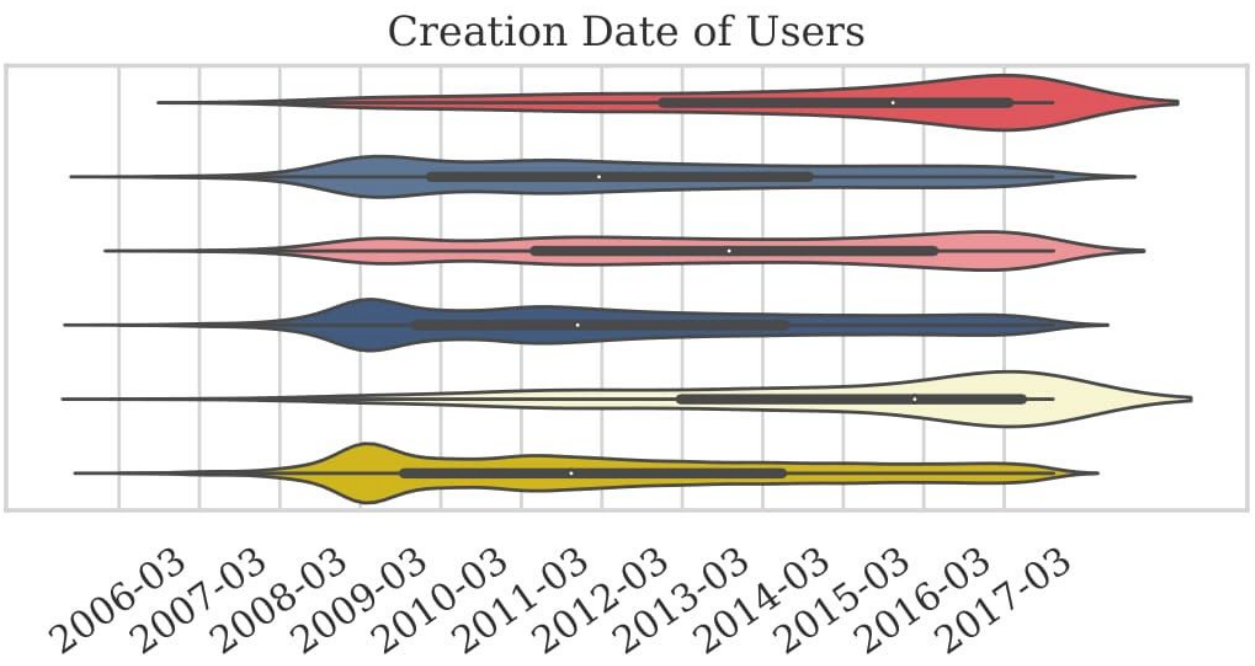


Figure 6.1 : Creation date of users - Top to Bottom labels: 1. Hateful Users, 2. Normal Users, 3. Hateful Neighbours, 4. Suspended Users, 5. Active Users

Kernel Density Estimation of creation dates of user accounts. The small white dot indicates the median and the thicker bar indicates the first and third quartiles.

6.1.3 Nature of Hateful Users

Through the analysis of metrics like number of followers, tweets, followees and favourite tweets. These metrics are normalised by dividing them by the number of days since account creation in the following figure. The results show that hateful users are more active. It can also be said that they are "power" users in the sense that they tweet more, in shorter intervals, favourite more tweets by other people and follow other users more. The analysis of neighbours yields very similar results with the exception of the number of favourite tweets metric.

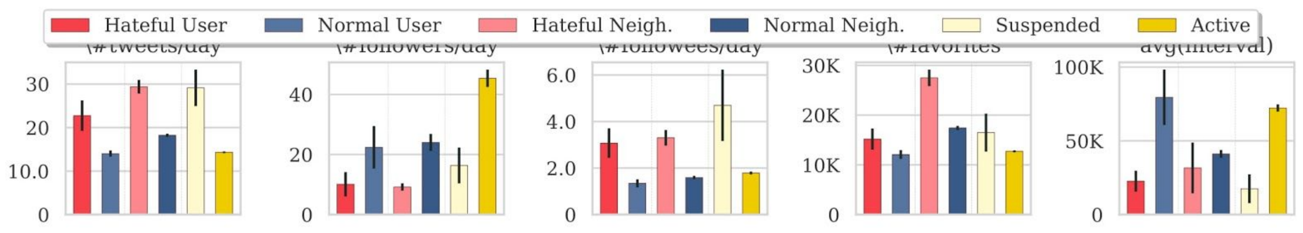


Figure 6.2 : Nature of users - Metrics from left to right - 1. #tweets/day, 2. #followers/day, 3. #followees/day, 4. #favourites, 5. avg(interval)

Average values for several activity-related statistics for hateful users, normal users, users in the neighborhood of those, and suspended/active users.

6.1.4. Comparison of hateful users with spammers

In order to carry this analysis, the metrics used are the ones which are generally used to detect spammers like URLs per tweet, hashtags per tweet and followers per followee. We inferred that hateful users, on an average, use less hashtags and less URLs. They same holds for their 1-neighbourhood. It was also found that average number of followers per followee are more for normal users. From this, it can be said that hateful users don't have refined content. They don't put efforts in reaching out and are only interested in expressing their anger online. This also suggests that hateful and suspended users do not use systematic and programmatic methods to deliver content. Their content is not engaging as well. **Thus, hateful users are not spammers.** This is all represented through the following figure:

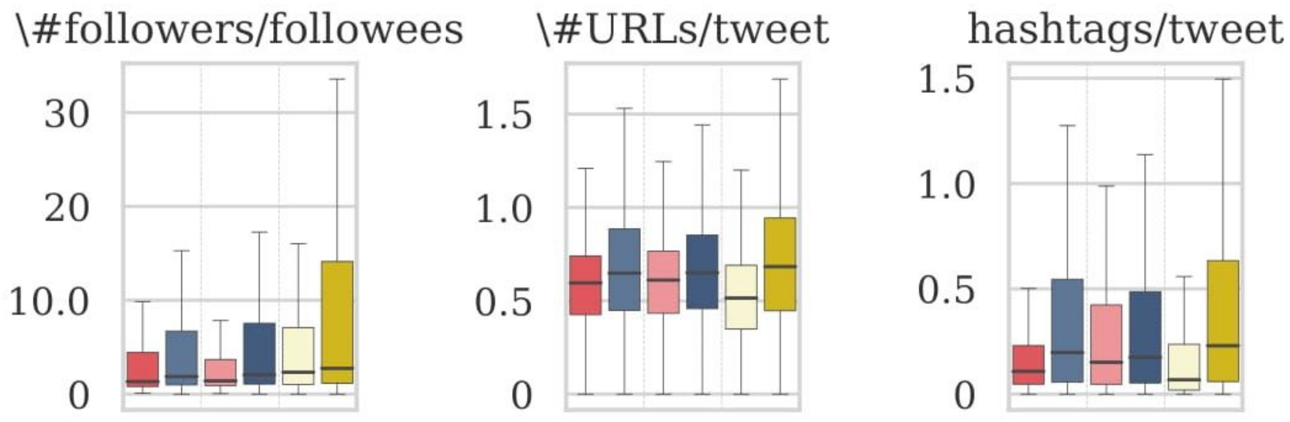


Figure 6.3 : Hateful users v/s spammers - From left to right labels - 1. Hateful Users, 2. Normal Users, 3. Hateful Neighbours, 4. Suspended Users, 5. Active Users

Boxplots for the distribution of metrics that indicate spammers. Hateful users have slightly less followers per followee, less URLs per tweet, and less hashtags per tweet.

6.1.5 Centrality of Hateful Users

We analyze different measures of centrality for users, as depicted in the following figure. The median hateful user is more central in all measures when compared to their normal counterparts. This is a counter-intuitive finding, as hateful crimes have long been associated with "lone wolves", and antisocial people. We observe similar results when comparing the median eigenvector centrality of the neighbors of hateful and normal users, as well as suspended and active users. In the latter pair, suspended users also have a higher median out degree. When analyzing the average for such statistics, we observe the average eigenvector centrality is higher for the opposite sides of the previous comparisons. This happens because some very influential users distort the value: for example, the 970 most central users according to the metric are normal. Notice that despite this, hateful and suspended users have higher average out degree than normal and active users respectively.

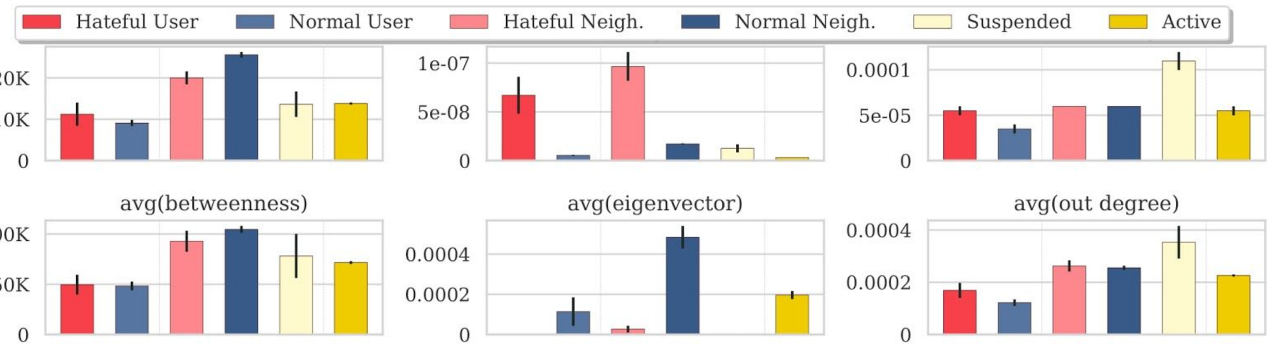


Figure 6.4 : Centrality of Hateful Users - Left to right, top to bottom metrics - 1. median(betweenness), 2. median(eigenvector), 3. median(out degree), 4. avg(betweenness), 5. avg(eigenvector), 6. avg(out degree)

Network centrality metrics for hateful and normal users, their neighborhood, and suspended/non-suspended users calculated on the sampled retweet graph.

6.1.6 Vocabulary of Hateful Users

We characterize users w.r.t. their content with Empath, as depicted in the following figure. Hateful users use less words related to hate, anger, shame and terrorism, violence, and sadness when compared to normal users. A question this raises is how sampling tweets based exclusively in a hate-related lexicon biases the sample of content to be annotated to a very specific type of "hate-spreading" user, and reinforces the claims that sarcasm, code-words and very specific slang plays a significant role in defining such user. Categories of words more used by hateful users include positive emotions, negative emotions, suffering, work, love and swearing (with p-values < 0.001), suggesting the use of emotional vocabulary. When we compare the neighborhood of hateful and normal users and suspended vs active users, we obtain very similar results. This also brings to our notice that just NLP analysis of tweets is ineffective in categorising hateful users.

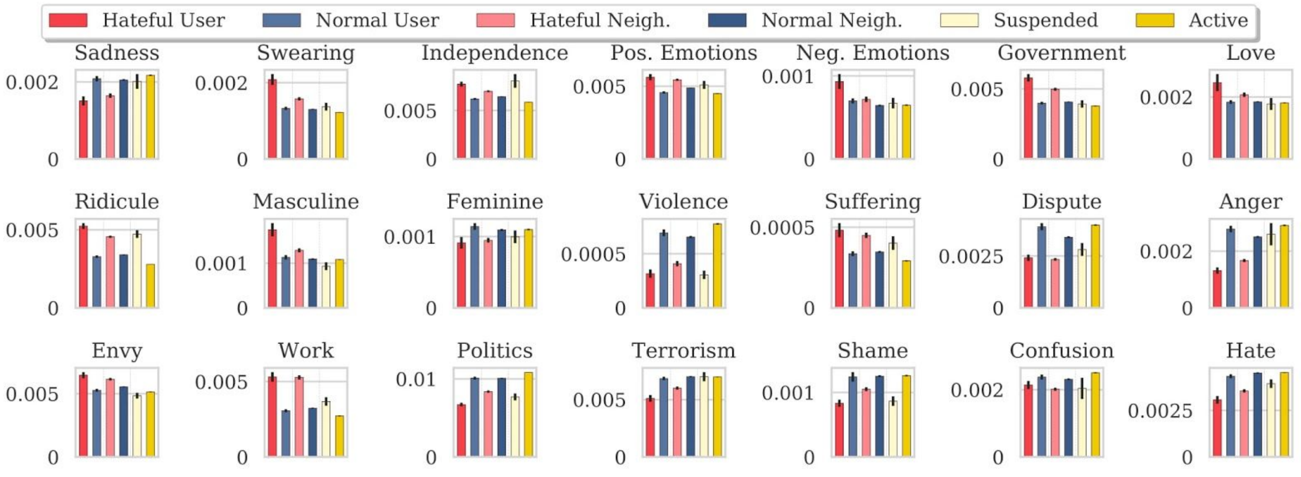


Figure 6.5 : Vocabulary of hateful users

Average values for the relative occurrence of several categories in Empath. Notice that not all Empath categories were analyzed and that the to-be-analyzed categories were chosen before-hand to avoid spurious correlations. Error bars represent 95% confidence intervals.

6.1.7 Sentiment of Hateful Users

We find that sentences written by hateful and suspended users are more negative, and are less subjective. The neighbors of hateful users in the retweet graph are also more negative, however not less subjective. We also analyze the distribution of profanity per tweet in hateful and non-hateful users. We find that suspended users, hateful users and their neighbors employ more profane words per tweet, also confirming the results from the analysis with Empath. It can be shown through the following figure:

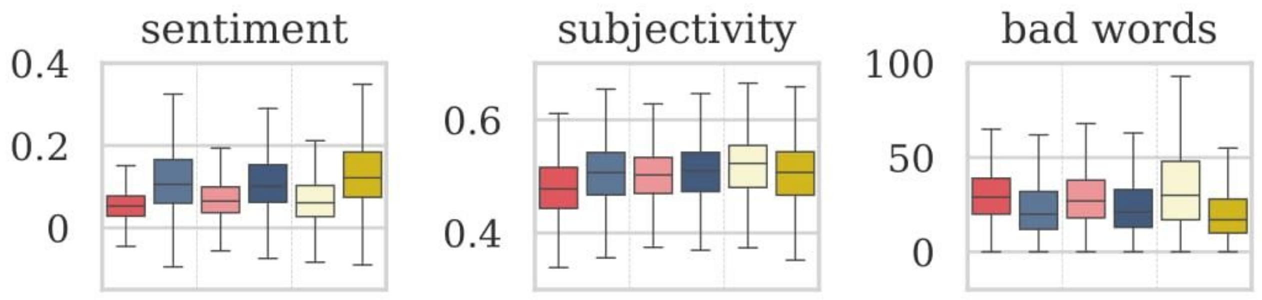


Figure 6.6 : Sentiment of hateful users - From left to right labels - 1. Hateful Users, 2. Normal Users, 3. Hateful Neighbours, 4. Suspended Users, 5. Active Users

Box plots for the distribution of sentiment and subjectivity and bad-words usage. Suspended users, hateful users and their neighborhood are more negative, and use more bad words than their counterparts.

6.1.8 Connectivity of Hateful Users

We find that 41% of the retweets of hateful users are to other hateful users, which means that they are 71 times more likely to retweet another hateful user, considering the occurrence of hateful users in the graph. We observe a similar phenomenon with suspended users, which have 7% of their retweets redirected towards other suspended users. As suspended users correspond to only 0.68% of the users sampled, this means they are approximately 11 times more likely to retweet other suspended users. The high density of connections among hateful and suspended users suggest a strong modularity. We exploit this, along with activity and network centrality attributes to robustly detect these users. Connectivity can be shown in the following figure:

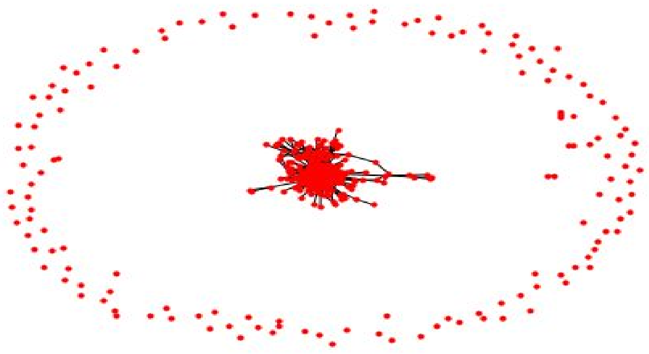


Figure 6.7 : Plot of hateful users and how they are connected. Made using NetworkX library.

6.2 Models Used For Training

6.2.1 GraphSAGE

We implemented a semi-supervised classifier using GraphSAGE (graph sampling and aggregation) operators for hateful user node classification with Pytorch geometric. The main purpose is to reproduce the results reported in (Yipeng et al., 2019[7]). GraphSAGE is a general inductive framework that makes use of node features and graph information to efficiently generate node embeddings for previously unseen data. With these generated embeddings, we conducted a binary classification on each node for hateful user detection. The main steps of the embedding generation process include aggregation from neighbors with sampling, update and normalization with multi-layer perceptron plus skip connection.

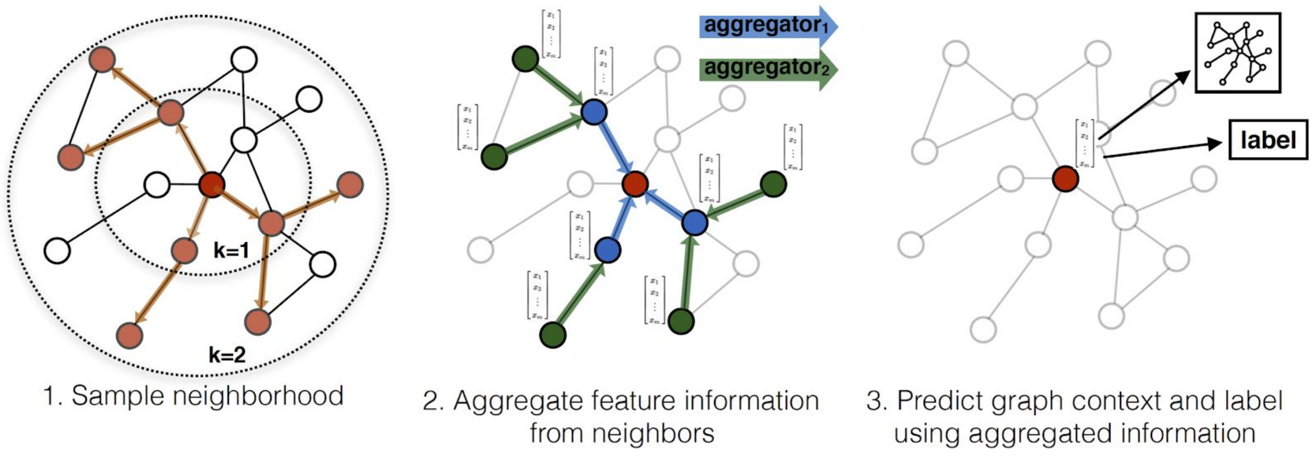

k=1
k=2
1. Sample neighborhood
aggregator1
aggregator2
2. Aggregate feature information from neighbors
label
3. Predict graph context and label using aggregated information

Figure 6.8 : Steps in GraphSAGE [13]

Low-dimensional vector embeddings of nodes in large graphs have numerous applications in machine learning (e.g., node classification, clustering, link prediction). However, most embedding frameworks are inherently transductive and can only generate embeddings for a single fixed graph. These transductive approaches do not efficiently generalize to unseen nodes (e.g., in evolving graphs), and these approaches cannot learn to generalize across different graphs. In contrast, GraphSAGE is an inductive framework that leverages node attribute information to efficiently generate representations on previously unseen data. To run GraphSAGE, it needs to train on an example graph or set of graphs. After training, GraphSAGE can be used to generate node embeddings for previously unseen nodes or entirely new input graphs, as long as these graphs have the same attribute schema as the training data.[13]

After preprocessing our data for the model and training using GraphSAGE, the epoch-wise distribution of loss looks as below:

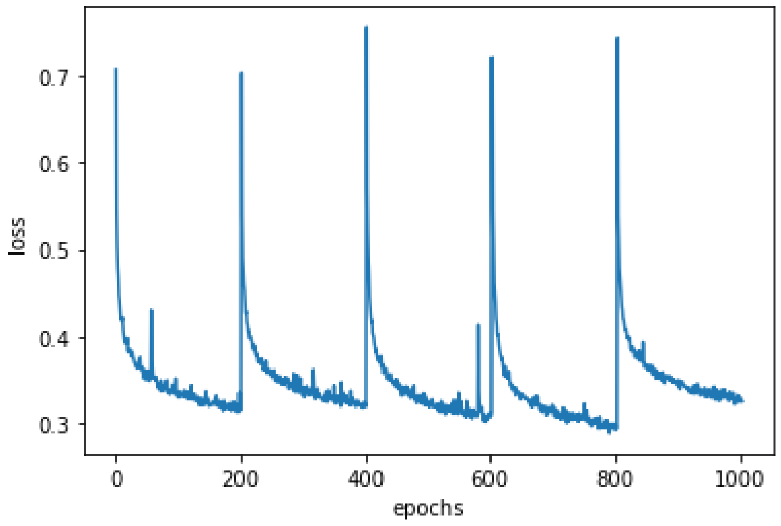


Figure 6.9 : GraphSAGE loss v/s epochs

6.2.2 Graph Convolutional Network(GCN)

GCNs are a very powerful neural network architecture for machine learning on graphs. In fact, they are so powerful that even a randomly initiated 2-layer GCN can produce useful feature representations of nodes in networks. The figure below illustrates a 2-dimensional representation of each node in a network produced by such a GCN. Notice that the relative nearness of nodes in the network is preserved in the 2-dimensional representation even without any training.

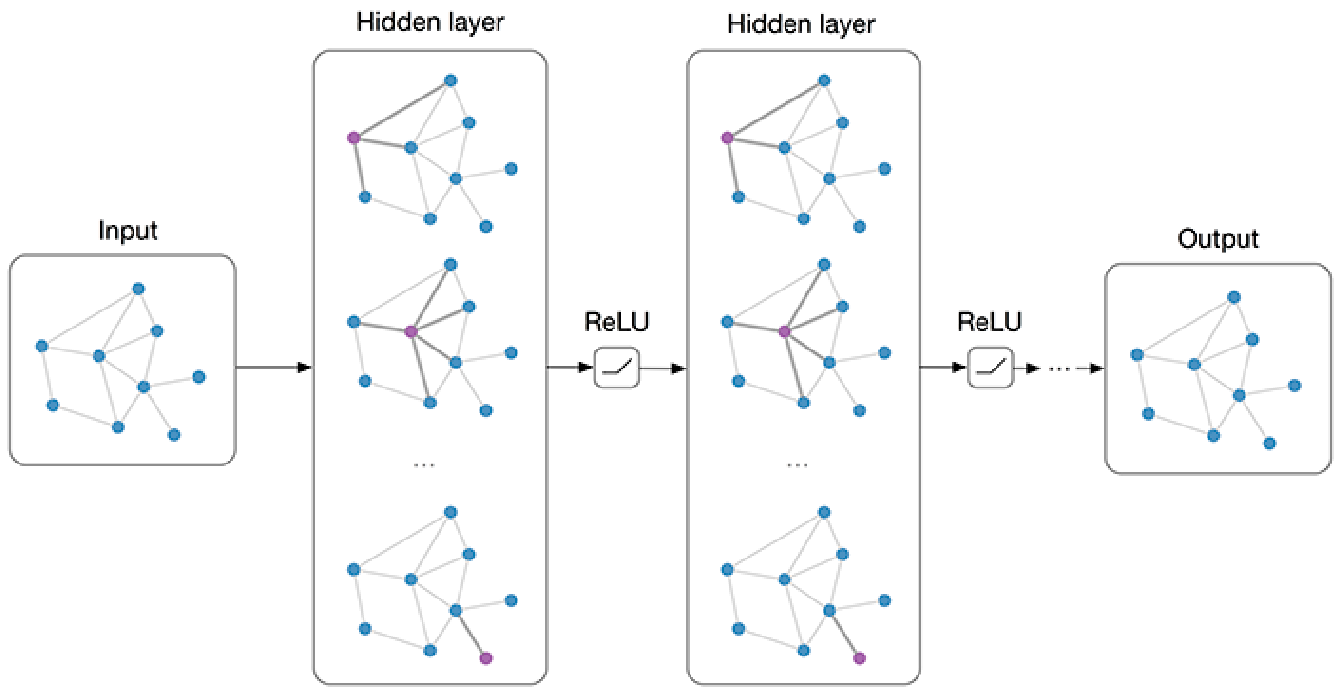


Figure 6.10 : Basic GCN architecture [14]

More formally, a graph convolutional network (GCN) is a neural network that operates on graphs. Given a graph G = (V, E), a GCN takes as input

1. an input feature matrix $N \times F^0$ feature matrix, X, where N is the number of nodes and F^0 is the number of input features for each node, and
2. an $N \times N$ matrix representation of the graph structure such as the adjacency matrix A of G.

A hidden layer in the GCN can thus be written as $H^i = f(H^{i-1}, A))$ where $H^0 = X$ and f is a propagation. Each layer H^i corresponds to an $N \times F^i$ feature matrix where each row is a feature representation of a node. At each layer, these features are aggregated to form the next layer's features using the propagation rule f. In this way, features become increasingly more abstract at each consecutive layer. In this framework, variants of GCN differ only in the choice of propagation rule f (1). [15]

After preprocessing our data for the model and training using GCN, the epoch-wise distribution of accuracy and loss looks as below:

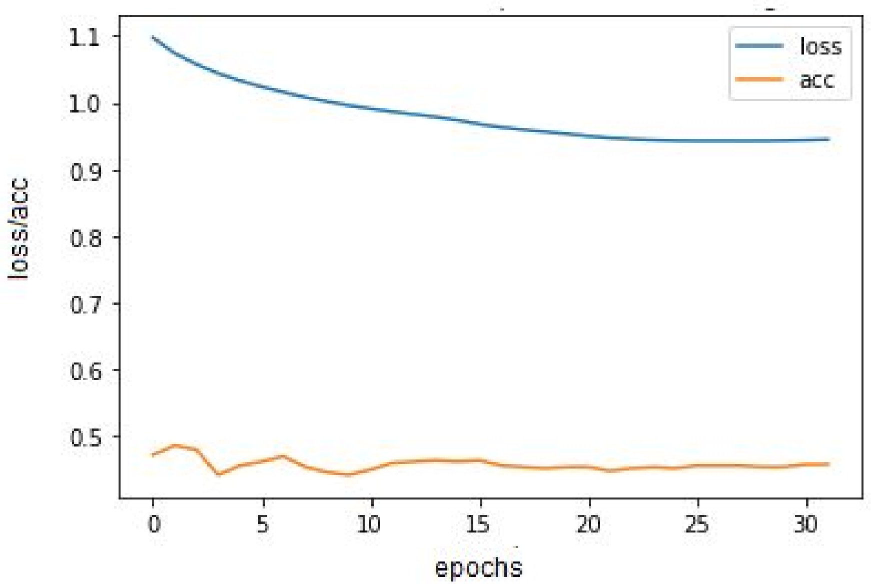


Fig 6.11 : GCN accuracy and loss v/s epochs

6.2.3 Hierarchical Graph Convolutional Networks(H-GCN)

Graph convolutional networks (GCNs) have been successfully applied in node classification tasks of network mining. However, most of these models based on neighborhood aggregation are usually shallow and lack the "graph pooling" mechanism, which prevents the model from obtaining adequate global information. In order to increase the receptive field, a novel deep Hierarchical Graph Convolutional Network (H-GCN) for semi-supervised node classification was proposed.

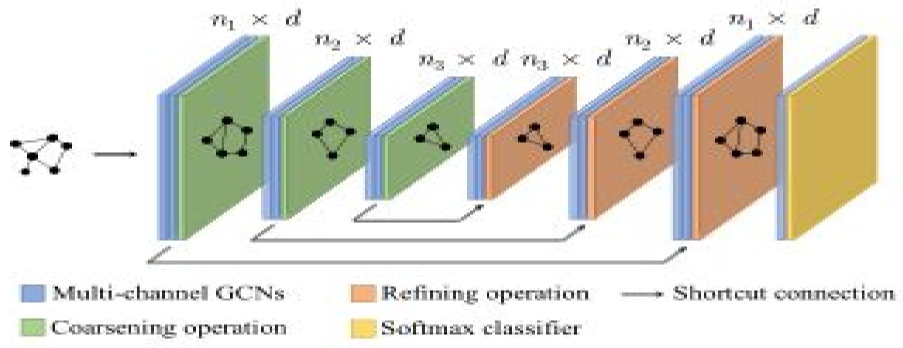


Figure 6.12 : HGCN architecture [16]

H-GCN first repeatedly aggregates structurally similar nodes to hyper-nodes and then refines the

coarsened graph to the original to restore the representation for each node. Instead of merely aggregating one- or two-hop neighborhood information, the proposed coarsening procedure enlarges the receptive field for each node, hence more global information can be captured. [16]
After preprocessing our data for the model and training using HGCN, the epoch-wise distribution of accuracy and loss looks as below:

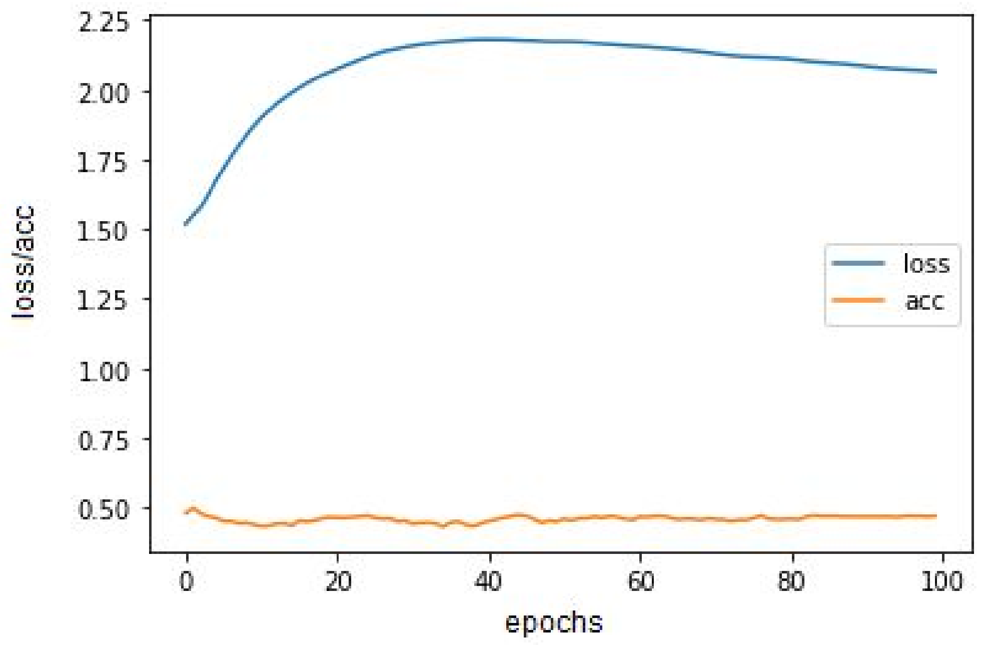


Figure 6.13 : HGCN loss and accuracy v/s epochs

6.2.4 Bayesian Graph Convolutional Neural Networks(BGCN)

Recently, techniques for applying convolutional neural networks to graph-structured data have emerged. Graph convolutional neural networks (GCNNs) have been used to address node and graph classification and matrix completion. Although the performance has been impressive, the current implementations have limited capability to incorporate uncertainty in the graph structure. Almost all GCNNs process a graph as though it is a ground-truth depiction of the relationship between nodes, but often the graphs employed in applications are themselves derived from noisy data or modelling assumptions. Spurious edges may be included; other edges may be missing between nodes that have very strong relationships.

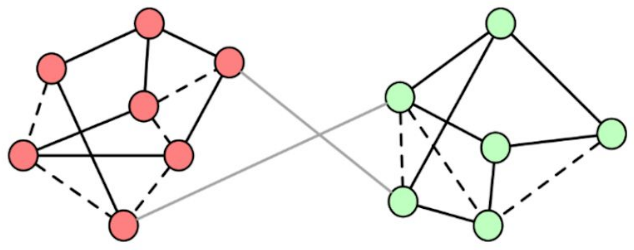
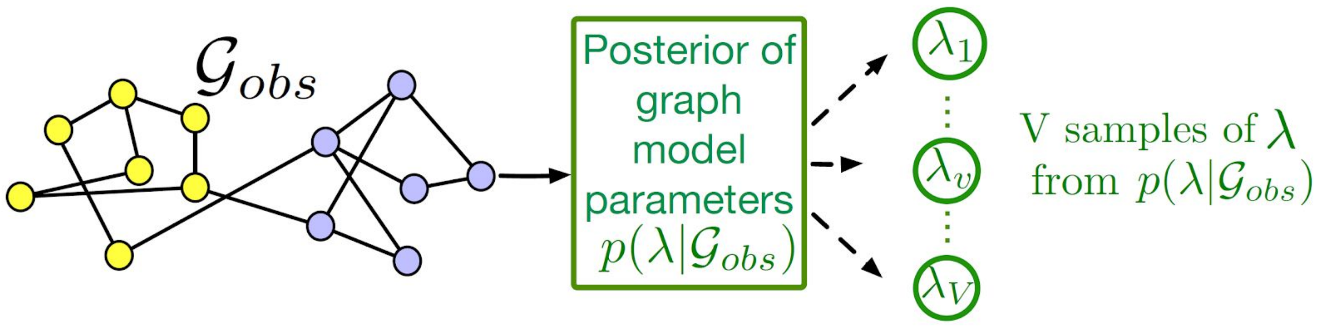


Figure 6.14 : Graph edges in non-bayesian approach [17]

Thus, a Bayesian approach is adopted in which we view the observed graph as a realization from a parametric family of random graphs. We then target inference of the joint posterior of the random graph parameters and the node (or graph) labels. We present the Bayesian GCNN framework and develop an iterative learning procedure for the case of assortative mixed-membership stochastic block models. [17]

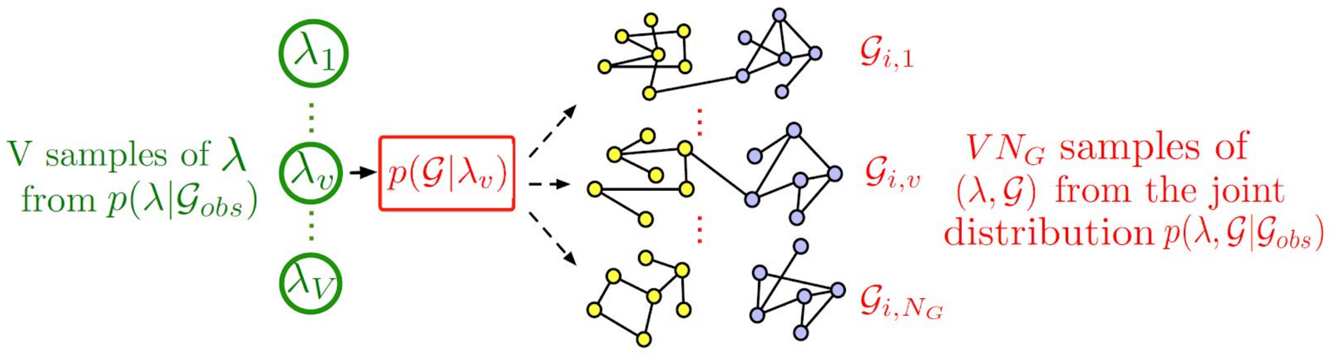
Steps of a BGCN depicted in a visual form

Step 1:


$\mathcal{G}_{obs}$
Posterior of graph model parameters $p(\lambda|\mathcal{G}_{obs})$
λ_1
λ_v
λ_V
V samples of λ from $p(\lambda|\mathcal{G}_{obs})$

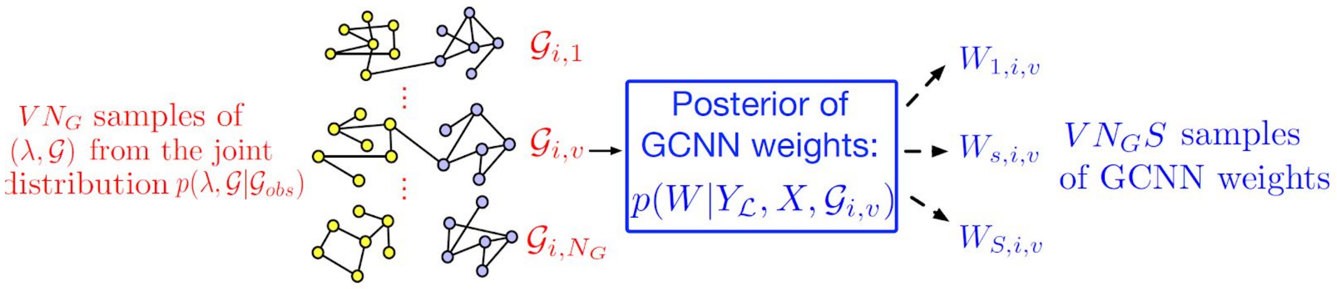
$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G}_{obs}) = \int p(\mathbf{Z}|W,\mathcal{G},\mathbf{X})p(W|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G})p(\mathcal{G}|\lambda)p(\lambda|\mathcal{G}_{obs})\,dW\,d\mathcal{G}\,d\lambda\,.$$

Step 2 :


V samples of λ from $p(\lambda|\mathcal{G}_{obs})$
λ_1
λ_v
λ_V
$p(\mathcal{G}|\lambda_v)$
$\mathcal{G}_{i,1}$
$\mathcal{G}_{i,v}$
$\mathcal{G}_{i,N_G}$
VN_G samples of $(\lambda,\mathcal{G})$ from the joint distribution $p(\lambda,\mathcal{G}|\mathcal{G}_{obs})$

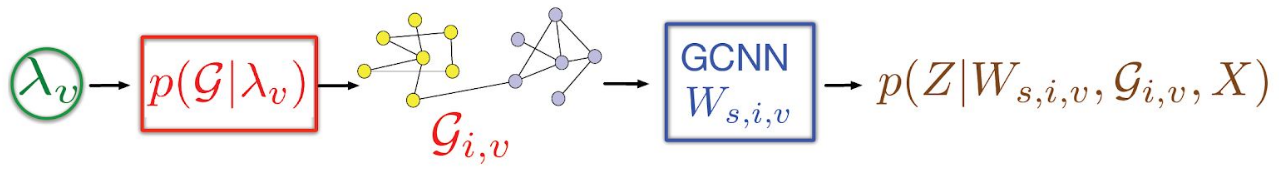
$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G}_{obs}) = \int p(\mathbf{Z}|W,\mathcal{G},\mathbf{X})p(W|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G})p(\mathcal{G}|\lambda)p(\lambda|\mathcal{G}_{obs})\,dW\,d\mathcal{G}\,d\lambda\,.$$

Step 3 :


VN_G samples of $(\lambda,\mathcal{G})$ from the joint distribution $p(\lambda,\mathcal{G}|\mathcal{G}_{obs})$
$\mathcal{G}_{i,1}$
$\mathcal{G}_{i,v}$
$\mathcal{G}_{i,N_G}$
Posterior of GCNN weights: $p(W|Y_{\mathcal{L}},X,\mathcal{G}_{i,v})$
$W_{1,i,v}$
$W_{s,i,v}$
$W_{S,i,v}$
VN_GS samples of GCNN weights

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G}_{obs}) = \int p(\mathbf{Z}|W,\mathcal{G},\mathbf{X})p(W|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G})p(\mathcal{G}|\lambda)p(\lambda|\mathcal{G}_{obs})\,dW\,d\mathcal{G}\,d\lambda\,.$$

Step 4:


λ_v
$p(\mathcal{G}|\lambda_v)$
$\mathcal{G}_{i,v}$
GCNN $W_{s,i,v}$
$p(Z|W_{s,i,v},\mathcal{G}_{i,v},X)$

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G}_{obs}) = \int p(\mathbf{Z}|W,\mathcal{G},\mathbf{X})p(W|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G})p(\mathcal{G}|\lambda)p(\lambda|\mathcal{G}_{obs})\,dW\,d\mathcal{G}\,d\lambda\,,$$
$$\approx \frac{1}{V}\sum_{v=1}^{V}\frac{1}{N_GS}\sum_{i=1}^{N_G}\sum_{s=1}^{S}p(\mathbf{Z}|W_{s,i,v},\mathcal{G}_{i,v},\mathbf{X})\,.$$

[17]

After preprocessing our data for the model and training using GCN, the epoch-wise distribution of accuracy and loss looks as below:

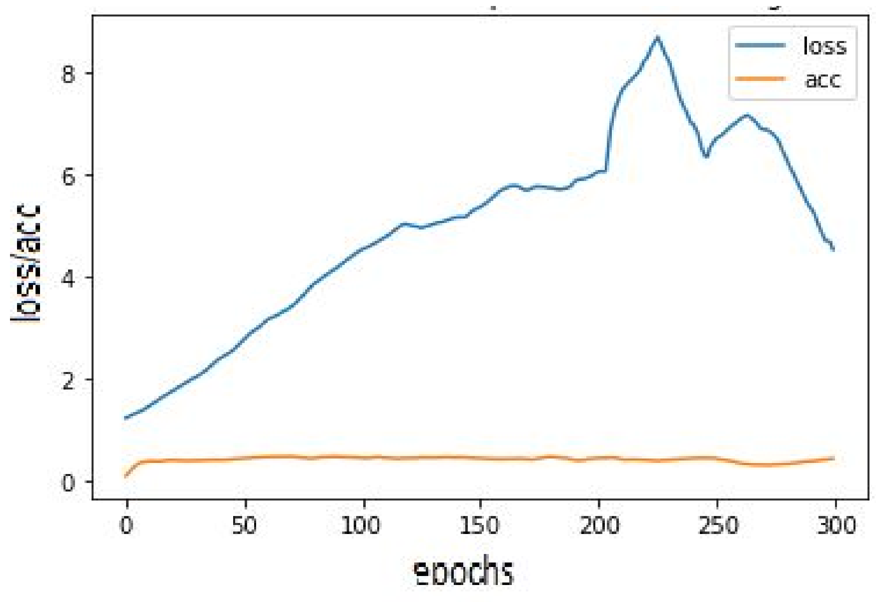


Figure 6.15 BGCN loss and accuracy vs epochs

Results from various models look as below:

Table 2. Model results

Model	Loss	Accuracy
GCN	0.94714	0.61051
HGCN	2.14972	0.55496
BGCN	**0.31735**	**0.93333**
GraphSAGE	0.3411	0.8489

Chapter 7 - Conclusion

We applied several models to perform node classification tasks on Twitter retweet graph, aiming to classify hateful users. The main challenge is that the dataset only has a small amount of labeled data (about 5% of the whole dataset). We used several semi-supervised learning methods based on graph structures, including GraphSAGE and GCNs. We can conclude that the graph structure and information of neighbors are helpful to classification accuracy.

References

[1] https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

[2] https://www.geeksforgeeks.org/kde-plot-visualization-with-pandas-and-seaborn/

[3] https://en.wikipedia.org/wiki/Centrality

[4] https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

[5] https://www.techopedia.com/definition/12937/network-connectivity

[6] https://arxiv.org/pdf/1803.08977.pdf

[7] http://web.stanford.edu/class/cs224w/project/26424341.pdf

[8] David R Bild, Yue Liu, Robert P Dick, Z Morley Mao, and Dan S Wallach. 2015. Aggregate characterization of user behavior in twitter and analysis of the retweet graph. ACM Transactions on Internet Technology (TOIT), 15(1):4.

[9] Manoel Horta Ribeiro, Pedro H Calais, Yuri A Santos, Virgilio AF Almeida, and Wagner Meira Jr. 2017. " like sheep among wolves": Characterizing hateful users on twitter. arXiv preprint arXiv:1801.00317.

[10] Min-Chul Yang, Jung-Tae Lee, Seung-Wook Lee, and Hae-Chang Rim. 2012. Finding interesting posts on twitter based on retweet graph analysis. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, pages 1073–1074. ACM.

[11] Phi Vu Tran. 2018. Learning to make predictions on graphs with autoencoders. In 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), pages 237–245. IEEE.

[12] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. 2018. Manifold mixup: Better representations by interpolating hidden states. arXiv preprint arXiv:1806.05236.

[13] https://towardsdatascience.com/an-intuitive-explanation-of-graphsage-6df9437ee64f

[14] https://tkipf.github.io/graph-convolutional-networks/

[15]
https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780

[16] http://snap.stanford.edu/hgcn/

[17] https://github.com/huawei-noah/BGCN