# Dynamic Graph Neural Networks on Hate Speech Detection

Serkan Karakulak
NYU Center for Data Science
serkan@nyu.edu

Raghav Jajodia
NYU Center for Data Science
rj1408@nyu.edu

## ABSTRACT

Hate speech detection on social networks is a critical task for ensuring healthy and sustainable public conversation around the world. Social media platforms today are held accountable for the toxic content on their platforms, and machine learning assumes a critical role in detecting malevolent users at scale. The complexity of natural language makes this task hard to solve by purely from text. There has been some promising work on exploiting information from the topology of the social network to infer the toxicity of the user. Our work compares various approaches to this problem on a semi-supervised learning task where we have limited number of labeled user, and we build on the current Graph Neural Network methods by utilizing not only the topology of the graph, but also the evolution of the topology over time. Our results show that in the case where there are limited number of labeled users, algorithms that exploit the graph topology outperforms content-base approaches that use users' features and tweets. Additionally, we show that utilizing the dynamic evolution of the graph improves the performance of hate-speech prediction.

## KEYWORDS

social networks, toxicity detection, graph representation learning

## 1 INTRODUCTION

Detecting hate-speech and toxicity on social networks has become an increasingly critical task over the recent years. We have witnessed some of the online social platforms becoming fundamental mediums where public conversation takes place. This, however, came with the expense of conventional trust mechanism that conventional media platforms provide. One of the ways this has manifested itself is that the ability for hateful content to be published and propagated has increased significantly [3]. This has various other social implication as well; Oksanen et al. [14] reports 67% of 15 to 18 year old social media users had been exposed to cyber hate on Facebook and YouTube, with 21% of them becoming victims of such material. Hence, detecting hate speech and stopping hate crimes on online platforms has become a crucial task, and many social platforms have been criticised in recent years for not doing enough to stop spread of such toxic online content.

As manual moderation of online content would not scale to the number of content that is shared on online platforms these days, machine learning seems to be a key tool for detecting toxic users on large social network platforms. Initial works to accomplish this task were based mostly on pure natural language processing methods, however this in itself had some limitations due to the complexity of natural language. It is worth noting that hate speech is observed very sparsely and depends largely on temporal, social and historical context [20]. Ribeiro et al. [18] illustrate this by giving the following tweet as example:

*"Timesup, yall getting w should have happened long ago"*

This tweet was in reply to another tweet that mentioned the holocaust. Without having this context, it is very hard to categorise this tweet as hateful using only the textual features. Also, consider the following tweet:

*"Hope one of those bitches falls over and breaks her leg"*

It is clear that this tweet contains offensive language. However classifying this as hate speech would require considering reviewing the context around this tweet. This can be a harmless commentary for a scene in a popular television show, or it can a tweet attacking to a minority group.

Aggregating textual features at the user level, and then combining this with user level features seems to be one possible strategy to address the shortcoming of inspecting each tweet separately. However we can do even better, and utilize information from users' social connections and enrich the data we use for our classification algorithm. Additionally, Graph Neural Networks [2] [10] [11] [23] offer a way of using an efficient variant of convolutional neural networks that can operate directly on graphs and propagate relevant information from nodes' neighbors. Using Graph Neural Networks (GNNs), we can combine text features and user features with neighbor information and the topology of the graph, hence would utilize more information to evaluate context of the content that are shared on social media platforms. One additional dimension we can utilize is the dynamicity of the social networks. Connections of the users change over time, and evolution of the topology can potentially allow us to exploit temporal patterns within the user graph as well. Exsiting works that study the escalation, duration, diffusion and de-escalation of hate speech online report clear temporal patterns [22], hence it seems as another source of information which will provide detectable and reliable signals of hate-speech on social networks. In this work we have evaluated all these options on Twitter dataset. Our code for all the experiments is available **here**

## 2 RELATED WORK

Extensive work has been done on detecting hate speech in social network settings. These methods primarily rely on textual features of the tweets, features of the users, and network based features to solve this problem. However, to the best of our knowledge no work has been done to utilize temporal features to detect hateful users on online media. Content based features mainly include the natural language text features from tweets, posts or websites represented as n-grams/BoWs [13], Glove Embeddings [15], and FastText embeddings [1]. On the other hand, user based features refer to the user's activity on social media, for example like-count and tweet-count in the case of Twitter network [7]. In addition, Network based features correspond to the users' neighborhood in the social network, for example social graph centrality, number of followers and the user-retweet graph. [18].

Most of the previous works model this problem as classification task by predicting the hateful label given the relevant features. Various studies have worked on this problem using supervised methods including Logistic Regression [5], Support Vector Machines [13], Gradient Boosted Decision Trees [1], Deep Neural Networks [1] and Graph Neural Networks [18].

Our work extends on the *Characterizing and Detecting Hateful Users on Twitter Dataset* paper of Ribeiro et al.[18], which successfully demonstrates that GNNs outperform other methods by utilizing the users' social-network topology. Ribeiro et al. 2018 uses GraphSAGE [9] model to learn hateful representations on the static graph of user retweet network.

We conduct our experiments on the dataset that Ribeiro et al. have introduced. Our experiments are summarized below:

- Set baselines using textual features alone by utilizing NLP methods to classify users based on their tweets
- Exploring the effect of adding user features and network features (including the full social graph itself) and comparing the results
- Experimented with different Graph Neural Network models to measure the effectiveness of graph topology features
- Experimented with Dynamic Graph Neural Network architecture to exploit the dynamic evolution of the network topology

## 3 DATA

### 3.1 Twitter Graph

Twitter graph $\mathcal{G}$ (fig 1) is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of nodes $\mathcal{V}$ consist of two types of nodes; $\mathcal{V} = \{\mathcal{U}, \mathcal{T}\}$, where $\mathcal{U}$ are the user nodes, and $\mathcal{T}$ are the tweets. Edge-types are defined by the interactions that we have in the dataset, which are users' own tweets, retweets, replies and quotes. In other words; $\mathcal{E} = \{\mathcal{E}_o, \mathcal{E}_{rt}, \mathcal{E}_{rp}, \mathcal{E}_q\}$, where the types of edges represent users' own tweets, retweets, replies and quotes respectively. In this representation, there is no edge between two users, and information can only be propagated through users' tweets. For the rest of the report, we would use the term *heterogeneous graph* to denote the case where we have multiple type of edges in our input.

### 3.2 User-Retweet subgraph

This twitter graph is too big for some of the algorithms that we use in our experiments. This is due to the fact that there are 22 million tweets in the dataset. For the static GCN experiments discussed in 5.4, we create a new graph by keeping user nodes only. Additionally, we remove all the edges and create an edge $e_{i,j}$ if user $i$ retweets user $j$. These simplifications to create homogenous graph follows from the study of Ribeiro et al. [18]. This study shows that *"Hateful users are 71 times more likely to retweet other hateful users than a normal user"*, indicating that *retweet* relation between users could be a strong indicator of hateful influence flow.

Number of nodes $\|\mathcal{U}\| = 100386$
Number of edges $\|\mathcal{E}_{rt}\| = 4289572$
Number of annotated hateful users (label 1) = 544
Number of annotated normal users(label 0) = 4427

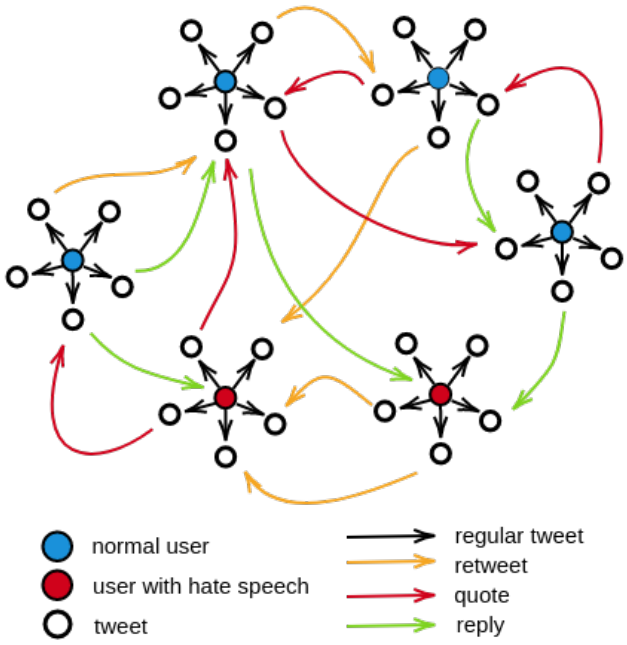



**Figure 1: Twitter graph.**

### 3.3 Definitions and Data Collection

Data is collected by Ribeiro et al. [18] for the study that was done in the *Characterizing and Detecting Hateful Users on Twitter Dataset* paper. Since the data and the data collection process is a fundamental part in assesment of this study, in our report we have directly quoted much of the data preparation section of the original paper.

**Hateful Users:** *"hateful user"* and *"hate speech"* are defined according to Twitter's guidelines. For the purposes of this study, *"hate speech"* is any type of content that 'promotes violence against or directly attack or threaten 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.

**Offensive Language:** Ribeiro et al. 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 [4].

Compared to the studies [21][4] that use a lexicon-based data collection, which involves sampling tweets that contain specific words, Ribeiro et al. employ a procedure that relies on lexicon indirectly and collects users rather than tweets.

First, they use *Direct Unbiased Random Walk* algorithm [17], which estimates out-degrees distribution efficiently by performing random jumps in an undirected graph it constructs online. With this strategy, they have collected a sample of Twitter graph with 100,386 users and 2,286,592 retweet edges along with the 200 most recent tweets (which can be replies and quotes) for each users.

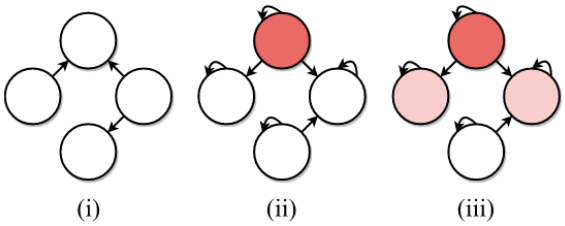



**Figure 2: Toy example of the diffusion process.**

*(i)* The process begins with the sampled retweet graph $\mathcal{G}$; *(ii)* We revert the direction of the edges (the way influence flows), add self loops to every node, and mark the users who employed words in the lexicon; *(iii)* We iteratively update the belief of other nodes.

As this graph is too large to be annotated entirely, a subsample is selected to be annotated. Choosing users uniformly at random, would risk having a very insignificant percentage of hate speech in the subsample. On the other hand, choosing users that have tweets containing obvious hatespeech features, such as offensive racial slurs, we will impose a bias in the selected subsample and potentially distort the complexity of the detection problem. Hence they:

- Create a lexicon of words that are mostly used in the context of hate speech. They use 23 words such '*holohoax*', '*racial treason*' and '*white genocide*', handpicked from Hatebase.org, and *ADL Hate Symbols Database*.
- Run a diffusion process on the graph based on DeGroot's Learning Model [8], assigning aninitial belief $p_0^i = 1$ to each user $u_i$ who employed the words in the lexicon; This prevents the sample from being excessively small or biased towards some vocabulary
- Divide the users in 4 strata according to their associated beliefs after the diffusion process, and perform a stratified sampling, obtaining up to 1500 user per strata.
- Manually annotate the sampled data using CrowdFlower

This process ensures that such users are annotated who didn't employ any of the words in the lexicon, but yet have a high potential to be hateful due to homophily. In total there are 4,972 twitter users who are annotated as hateful or not using CrowdFlower. The annotators were given the definition of hateful conduct according to Twitter's guidelines, mentioned above in the report.

Annotators were asked to consider the entire profile (limiting the tweets to the ones collected) rather than individual publications or isolate words and were given examples of terms and codewords in ADL's hate symbol database. Each user profile was independently annotated by 3 annotators, and, if there was disagreement, up to 5 annotators. In the end, 544 hateful users and 4,427 normal ones were identified by the annotators.

## 3.4 Features

Each tweet is represented by average of 300 dimensional GloVe word embeddings of the tweet tokens [16]. We also use 768 dimensional sentence level representations produced by BERT [6] for some of the experiments.

Users are represented as 320 dimensional feature. This includes 300 dimensional GloVe features created by averaging GloVe embeddings of all the tokens in a tweet, and subsequently averaging embeddings of all the tweets by a user. Rest 20 features are based on user attributes (such as number of statuses, followers, followees, favorites) and network features (such as betweenness, eigenvector centrality and the in/out degree of each node).

## 3.5 Data Statistics

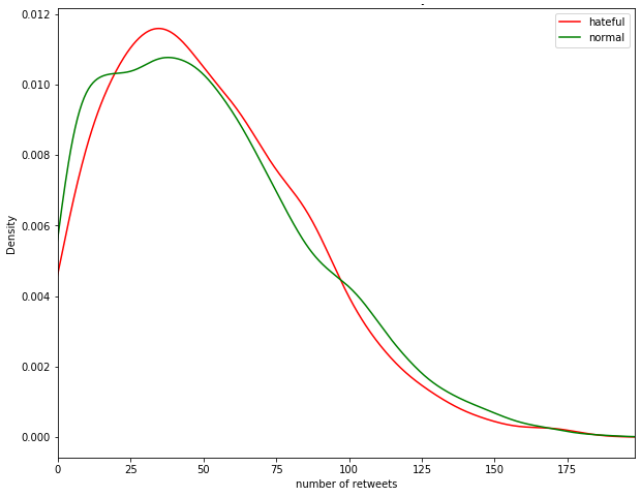



**Figure 3: Distribution of Retweets by a user.**

Edges in social networks potentially contain rich contextual information about users' interests, views, demographics, etc. and hence would be a good information source to utilize for many prediction tasks that we may have for users on social networks. To confirm that there are detectable patterns in the topology of the graph, we have studied some basic statistics that relates to the edges.

We have done the analyses for edges formed by *retweets*, *quotes* and *replies* as well and observed that even though the distribution of total numbers of interactions look similar (Fig: 3), normal users interact more with normal users (Fig: 4) and hateful users interact more with hateful users (Fig: 5). Additionally, according to Ribeiro et al. [18], *"hateful users are 71 times more likely to retweet other hateful users than normal users"*, which shows that there is a clear modularity of hateful users in user-retweet graph 3.2. Also we see that the hateful user are more active, and form connections more rapidly. Studies [22] report clear temporal patterns in the escalation, duration, diffusion and de-escalation of hate speech. This also hints at why utilizing dynamic evolution of the graph can potentially give rich information and provide shows clear patterns that can be captured by out algorithms.

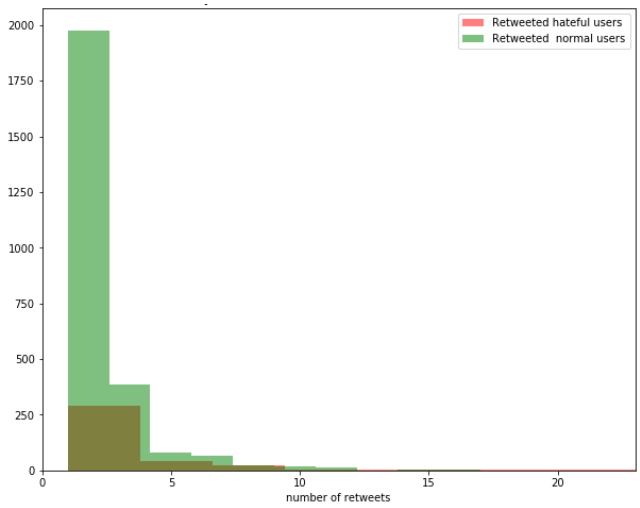



**Figure 4: How many times normal users retweet normal/hateful users.**

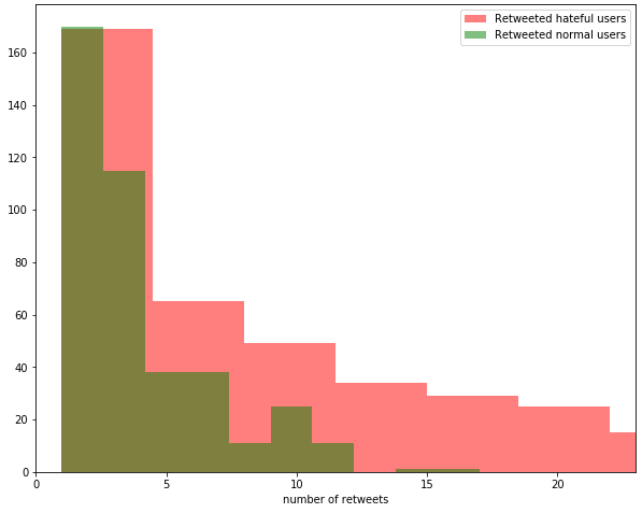



**Figure 5: How many times hateful users retweet normal/hateful users.**

# 4 MODEL DETAILS

Let $x \in \mathbb{R}^{inp}$ denote the input features for users, and $y \in \{0, 1\}$ denote the hateful label. This problem can be formulated as learning the parameters $\theta$ in the following equation

$$P_\theta(y = 1 \mid x) = \frac{1}{1 + e^{-f_\theta(x)}} = p_\theta(x) \tag{1}$$

To learn parameter $\theta$ we minimize binary Cross entropy,

$$\mathbb{E}_{x,y}[y \log(p_\theta(x) + (1 - y) \log(1 - p_\theta(x)))] \tag{2}$$

## 4.1 Class Balancing

Since the number of positive instances are much less as compared to number of negative instances, we adjust the effect of class imabalance by using weighted Binary Cross Entropy loss. This is important to avoid learning label statistics, and instead using input features to predict the label. We use the following loss -

$$\mathbb{E}_{x,y}[qy \log(p_\theta(x) + (1 - y) \log(1 - p_\theta(x)))] \tag{3}$$

where q is the balancing coefficient = $\frac{\#\text{ of negative examples}}{\#\text{ of positive examples}} \approx 8.13$. Note that Ribeiro et al [18] also uses the same balancing on this dataset.

We describe different models $f_\theta(x)$ that we use subsequently in our experiments -

## 4.2 Logistics Regression

$$f_\theta(x) = w^\top x, \quad w \in \mathbb{R}^{320}$$

## 4.3 Multi Layer Perceptron

2 layer perceptron with hidden dimension 256 -

$$f_\theta(x) = W_2\,[\sigma(W_1 x + b_1)] + b_2,$$

where $W_1 \in \mathbb{R}^{256x320}, b_1 \in \mathbb{R}^{256}, W_2 \in \mathbb{R}^{1x256}, b_2 \in \mathbb{R}, \sigma$ is activation function

## 4.4 Classification from Tweets

We use modified version of MLP (4.3) as $f_\theta(u_i)$ for the task in 5.3.1. Let $z_{i,j}$ be the sentence level embedding of $j$th tweet of user $i$. We can either produce $z_{i,j}$ by using sentence level embeddings produced by BERT, or by averaging the GloVe word embeddings of the words that the tweet contains.
$u_i$ is the user representation created by aggregating tweets related to user i. We try two types of aggregation

- We can sample a set of $n$ tweets $S_i$ "directly tweeted" by user $i$, where $n \sim \text{Unif}[a, b]$. Then,

$$u_i = \frac{1}{n} \cdot \sum_{j \in S_i} z_{i,j}$$
$$f_\theta(u_i) = \text{MLP}(u_i)$$

- We can also use different weights for different tweet types $\mathcal{R} = \{\text{tweet, retweet, reply, quote}\}$. Then $S_{ik}$ denotes the subset of $S_i$ which are connected to node i with relation '$k$'. Then,

$$u_i = \frac{1}{n} \cdot \left( \sum_{k \in \mathcal{R}} \sum_{j \in S_{ik}} W_k z_{i,j} \right)$$
$$f_\theta(u_i) = \text{MLP}(u_i)$$

## 4.5 Graph Convolution Networks

Problem definition in GCNs [12] is as follows: Given a homogeneous graph with node feature $x_i$ of $i$th node, representation $h_i^{(l+1)}$ of $i$th

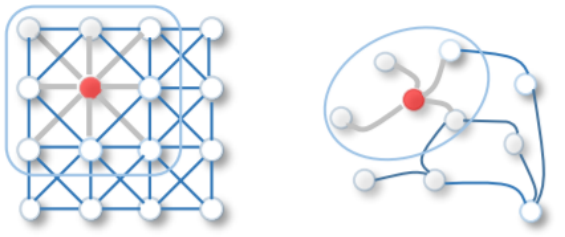
node at layer (l+1) -

$$h_i^{(l+1)} = \sigma\left(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} h_j^{(l)} W^{(l)}\right)$$

$\mathcal{N}(i)$ is the neighbor set of node i,
$c_{ij} = \sqrt{|\mathcal{N}(i)|}\sqrt{|\mathcal{N}(j)|}$, $\sigma$ is an activation function , $h_i^1 = x_i$

$$f_\theta(x_i) = w^\top h_i^L$$

where L = 3 is number of GCN layers and
$w \in \mathbb{R}^{256}$ is hidden dimension of GCN
Intuitively, L-layer GCN is equivalent to aggregating features of L hop neighborhood of a node i as shown in the figure 6



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.

(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

**Figure 6: Image Convolution and Graph Convolution**

## 4.6 GraphSAGE

GraphSAGE[9] is another Graph Neural Network to model static graphs, similar to GCNs. Note that we have different learnable weight for current node features when aggregating neighbor features with current node features, which is not true for GCNs.

$$\begin{aligned} h_{\mathcal{N}(i)}^{(l+1)} &= \text{aggregate}\left(\{h_j^l, \forall j \in \mathcal{N}(i)\}\right) \\ h_i^{(l+1)} &= \sigma\left(W \cdot \text{concat}(h_i^l, h_{\mathcal{N}(i)}^{l+1} + b)\right) \end{aligned} \tag{4}$$

Here 'aggregate' is a function to aggregate neighbor node features, which could be either {'mean', 'gcn', 'lstm', 'pool'}. Rest of the notation remains same as GCN model above.

## 4.7 Relational GCN [19]

Given a heterogeneous graph with node feature $x_i$ of ith node, representation $h_i^{(l+1)}$ of ith node at layer (l+1) -

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} W_r^{(l)} h_j^{(l)}\right)$$

where $\mathcal{R}$ is set of different relations in the graph,
$\mathcal{N}_r(i)$ is the neighbor set of node i with relation r, $h_i^1 = x_i$

$$f_\theta(x_i) = w^\top h_i^L$$

where L = 3 is number of GCN layers and
$w \in \mathbb{R}^{256}$ is hidden dimension of Relational GCN

## 4.8 Relational GCN with edge features

We propose the following model to incorporate edge features in Relational GCN -

$$h_i^{(l+1)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} W_r^{(l)} h_j^{(l)} + V_r^{(l)} u_{ij}\right)$$

Here $u_{ij} \in R^{|edgefeatures|}$ is the feature vector of edge between node i and node j, connected with relation r, and $V_r^{(l)}$ is learnable parameter of the model. Rest of the notation remains exactly same as Relational GCN

## 4.9 Dynamic GNN

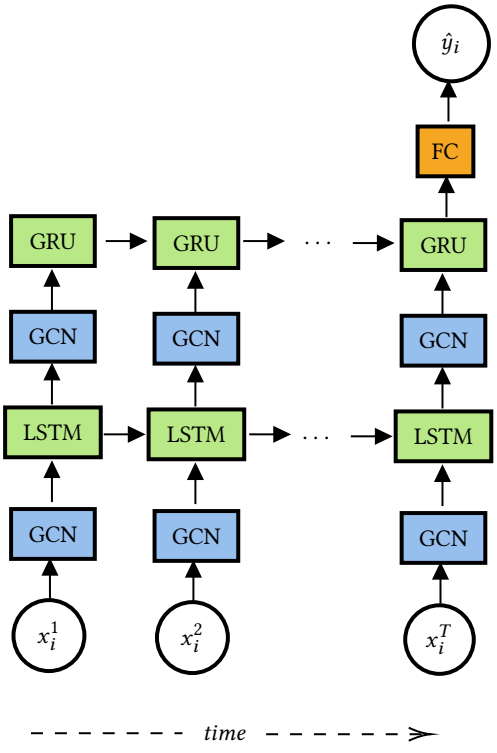



**Figure 7: Dynamic GNN**

In Dynamic variant of the Graph Neural Networks, we have combined GCNs with RNNs and introduced hidden layers that carry information over time. Let $x_i^t$ be the node features of user $i$ at time $t$. In our dataset users' features do not change over time, but their neighborhood does. Let the Graph Convolution operation $G^l(h_i^{t,l}) = h_i^{t,l+1}$ be computed as follows:

$$h_{\mathcal{N}(t,i)}^{(t,l+1)} = \text{aggregate}\left(\{h_j^{t,l}, \forall j \in \mathcal{N}(t,i)\}\right)$$
$$h_i^{(t,l+1)} = \sigma\left(W \cdot \text{concat}(h_i^{t,l}, h_{\mathcal{N}(t,i)}^{t,l+1} + b)\right)$$

Also,

$$h_i^{(t,0)} = x_i^t$$

Let $c_{t,i}$ be the hidden state of GRU at timestep $t$. Also let $l_{t,i}$ and $o_{t,i}$ be long-term and short term hidden cells of an LSTM respectively. We update the hidden state of the LSTM as follows:

$$l^{t,i}, o^{t,i} := \text{LSTM}((l^{t-1,i}, o^{t-1,i}), h_i^{t,1})$$

Next, we run one more iteration of GCN to create an embedding that has information from its 2-hop neighborhood, and also has information containing temporal information of that user.

$$h_i^{t,2} = G^2(o^{t,i})$$

Using $h_i^{t,2}$ we update our previous user embedding using GRU.

$$c_{t,i} = \text{GRU}(c^{t-1,i}, h_i^{t,2}))$$

At every $n$th timestep, we can use $c_{t,i}$ to predict the label of the user, propagate our error over to the previous $n$ timesteps, and do one step of truncated backpropagation.

$$f_\theta(x_i) = w^\top c_{t,i}$$

# 5 EXPERIMENTS AND RESULTS

## 5.1 Evaluation Metrics

We choose AUC (Area under operating curve) as the evaluation metric for all the models. This was decided considering that AUC metric is tolerant to class imbalance to some degree, and this works well empirically for this problem.

## 5.2 Feature definition

Unless mentioned, we will be using the same 320 features for all the below experiments as Ribeiro et al [18]. These features are comprised of 300 dimensional Glove features of past retweets, user features like status/follower count and network features like retweet graph 'betweenness'/eigenvectors, which has been already discussed in section 3.4

## 5.3 Baseline

*5.3.1 Baseline with Textual Features.* In this experiment, we set a baseline using just the textual features. Inputs of this model is users' tweets in addition to the tweets they have interacted with including {quotes, retweets, replies and tweets}. We first produce sentence level embeddings using BERT, and then follow the prediction model that was described in section 4.4. Aggregation type 2 gives the best AUC of 0.884.

*5.3.2 Baseline with Logistic Regression and MLP.* We set the baseline results of the features by learning Logistic Regression and MLP (section 4.2 and 4.3). These models do not utilize topology of user-retweet graph, and hence serve as good baselines to compare with Graph based models. AUC obtained for logistic regression and MLP are 0.8646 and 0.8994 respectively. This shows that combining information from tweets with the user features gives better results then using textual features alone.

## 5.4 Static Graph Experiments

*5.4.1 Experiment with GraphSAGE.* Original retweets in the user-retweet graph 3.2 are directional, which means that if *user a retweets b*, this does not necessarily means that *user b retweets user a*. Hence adjacency matrix of the original graph is not symmetric. However, we make the edges bidirectional to allow hateful message flow in both the directions, thus making adjacency matrix symmetric. We then learn GraphSAGE model (section 4.6) on this user-retweet graph, using the same node features as baseline model. We observe that this model gets AUC of 0.9189 and consistently outperforms the baseline model, indicating that graph topology provides contains useful information for this task, and utilizing graph topology gives better results.

*5.4.2 Performance Difference with Limited Features.* To verify the importance of topology, we experiment by dropping input features from 320 dimensions to 10, 20 and 40 dimensions and we train MLP and GraphSAGE with these features. The results show clear gap between MLP and GraphSAGE AUC (summerized in table 1). With just 10 dimensions, MLP gets 0.7924 AUC vs. 0.8946 AUC for graphSAGE, indicating that graph topology contains rich information to solve this problem, which would become even more valuable when there is not enough user features available for the prediction task.

| AUC with Feature dimensions | | | | |
|---|---|---|---|---|
| Models | **10 dims** | 20 dims | 40 dims | 320 dims |
| Log-Reg | 0.779 ± 0.000 | 0.835 ± 0.000 | 0.878 ± 0.000 | 0.890 ± 0.000 |
| MLP | **0.792** ± 0.002 | 0.853 ± 0.001 | 0.883 ± 0.001 | 0.901 ± 0.004 |
| GraphSAGE | **0.894** ± 0.001 | 0.901 ± 0.000 | 0.911 ± 0.000 | 0.921 ± 0.001 |

**Table 1: Table shows variation of AUC with feature dimensions and models. Features used are NLP + User attributes + Graph Centrality measures**

*5.4.3 Relational GCN on Heterogeneous Graph .* We also experimented with original directional adjacency matrix (*retweeted-by* relation) to learn GraphSAGE model described above.Note that if there is an edge *'a retweeted-by b'*, then this means that hateful message flows from node a to node b in the graph, which characterizes the natural hate flow amongst the users. Surprisingly, this model gets 0.9097 AUC, which does not improve the bidirectional graph results. Similarly we train graphSAGE on *'retweets'* relation, and observe 0.8979 AUC. However, this verifies our intuition of better message flow in *'a retweeted-by b'* relation than *'a retweets b'* relation.

We also train Relational GCN heterogenous graph in section 4.7 learning separate weights for 'a retweets b' and 'a retweeted-by b' relation. This model gets approximately same results as bidirectional model case. Results presented in fig 9

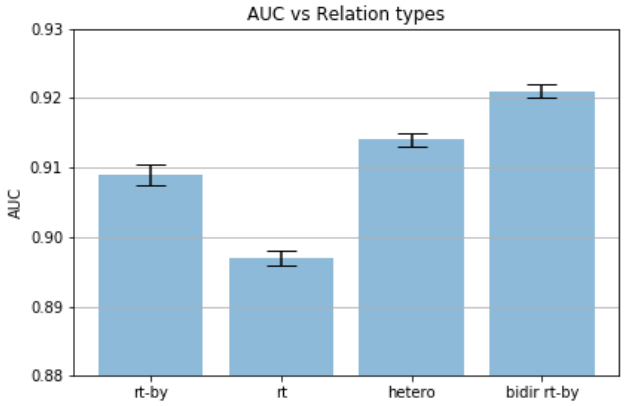



**Figure 8: AUC of graphs with different types of relations**

*(i) rt-by: retweeted-by (ii) rt: retweets (iii) hetero: separate relations for rt-by and rt (iv) bidir rt-by: symmetric retweeted by*

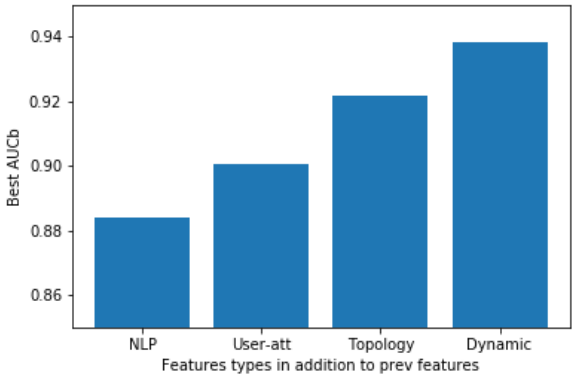



**Figure 9: Best AUC for additional feature types**

*(i) NLP only, obtained with MLP (ii) NLP and User-attributes, obtained with MLP (iii) NLP, User-att and Network feats, obtained with GraphSAGE (iv) Prev feats and topology timeseries, obtained with Dynamic GNN*

## 5.5 Dynamic Graph Experiments

*5.5.1 Adding Time Nodes to the Static Graph.* We discretized the time space in T=100 timestamps and we created a time node in the user-retweet graph corresponding to every timestamp. User node $u_i$ and the time node $u^t$ is connected if the user $u_i$ retweets at time $t$. The motivation to use this graph for modelling dynamic information was that the new edges would pass the message amongst users retweeting in the same timestamp. We ran the GraphSAGE algorithm on this graph, and the experiment gave an AUC of 0.906, which is less then the AUC that we had using GraphSAGE on the user-retweet graph. The reason is that adding time nodes completely changes the graph topology, and creates shortcut connections between two very different users tweeting at same time who are otherwise far apart from each other in the original graph.

*5.5.2 Unrolling Graph Over Timestamps.* In this experiment we used the Dynamic GNN model that we proposed in section 4.9. We have used truncated backpropagation, and backpropagated the errors backwards to the last 4 timesteps due to memory constraints. We have also considered all the edge types that were described in (1), and learned different message passing function for each of these edge types. In addition we also use temporal aggregated features like retweet count as edge features. This gives AUC of 0.9384, which is much better than static graph models. This shows us that utilizing the dynamic evolution of the graph topology and using temporal features allows us to detect hateful users with more accuracy.

## 6 CONCLUSION AND FUTURE WORK

We would like to discuss some challenges faced along this task. In section 5.4.2, we showed that graph topology alone gets really good results (0.89 AUC). Even though we have managed to increase the performance using a dynamic variant of the GCNs, the baselines that we had was already quite high which did not leave too much room for improvements. We would have prefered to benchmark our approach on multiple datasets which contained interesting graph structure that evolved over time, however we have found that there was not many open datasets that contained interesting graphs that evolved over time. Another challenge is that dynamic GNN proposed here is memory intensive and its difficult to train this model on large timestamps. We needed to do truncated backpropagation at every four timesteps, which may have limited model's capability to learn long-term dependencies.

Based on these identified challenges, some directions of the future work could be

- Formulate sampling methods which would avoid fitting the full training graph in memory. This includes training the model in minibatches of subgraphs, which remains a open research question, or we can sample over the edges
- Understand the basic dynamics of the proposed dynamic GNN model, and come up with qualitative failures and success cases of capturing temporal information
- Include temporal aggregated information as edge features in the static GNN that operated on user-retweet graph. Some examples of these features could be # of retweets, retweet frequency etc.
- Identify a challenging task for static graphs that has complex topology evolution, and measure the performance difference to dynamic GNNs. This may also be a toy dataset where we can create temporal pattern that can not be captured by static graphs, and observe how successful dynamic GNN had performed in detecting that

As a conclusion, we presented different experiments to gain insights about Graph Based model and Hate Speech Detection problem. Our conclusions and contributions can be summarized below as

- We show that performance on Hate Speech detection task increases if we utilize User-based and network related features in addition to NLP features.
- We verify the effectiveness of graph topology in Hate Speech detection task, especially in the setting of inadequate user

features. There is a clear modularity of hateful users in user-retweet graph, which serves as important feature for this problem.
- We compare the performance with directional edges in both directions. For user-retweet graph *a retweeted-by b* relation is better than *a retweets b* relation for passing hatefulness message in Graph. This coincides with the natural intuition of hatefulness influence, and we conclude that modelling heterogenous edges provides better results than directional homogenous edges.
- We also find that adding artifact nodes for timestamps is **not** an effective method to model timeseries graphs.
- Finally we propose a new model to learn dynamic graph and we show that dynamic topology information and temporally aggregated features could improve the model performance

## 7 ACKNOWLEDGEMENTS

We would like to thank Prof Carlos Fernandez-Granda and Prof Joan Bruna for their guidance and ideas.
We would also like to acknowledge Lei Chen, Pau Batlle and Min Jae Song for helping us with implementation, brainstorming ideas and results, and designing the experiments.

## REFERENCES

[1] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. *CoRR* abs/1706.00188 (2017). arXiv:1706.00188 http://arxiv.org/abs/1706.00188
[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. arXiv:cs.LG/1312.6203
[3] Pete Burnap and Matthew L. Williams. 2015. Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. *Policy & Internet* 7, 2 (June 2015), 223–242. https://doi.org/10.1002
[4] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. arXiv:cs.CL/1703.04009
[5] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. *CoRR* abs/1703.04009 (2017). arXiv:1703.04009 http://arxiv.org/abs/1703.04009
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:cs.CL/1810.04805
[7] Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A Unified Deep Learning Architecture for Abuse Detection. *CoRR* abs/1802.00385 (2018). arXiv:1802.00385 http://arxiv.org/abs/1802.00385
[8] Benjamin Golub and Matthew O. Jackson. 2010. Naïve Learning in Social Networks and the Wisdom of Crowds. *American Economic Journal: Microeconomics* 2, 1 (February 2010), 112–49. https://doi.org/10.1257/mic.2.1.112
[9] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *CoRR* abs/1706.02216 (2017). arXiv:1706.02216 http://arxiv.org/abs/1706.02216
[10] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. arXiv:cs.LG/1506.05163
[11] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:cs.LG/1609.02907
[12] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016). arXiv:1609.02907 http://arxiv.org/abs/1609.02907
[13] Rijul Magu, Kshitij Joshi, and Jiebo Luo. 2017. Detecting the Hate Code on Social Media. *CoRR* abs/1703.05443 (2017). arXiv:1703.05443 http://arxiv.org/abs/1703.05443
[14] Atte Oksanen, James Hawdon, Emma Holkeri, Matti Näsi, and Pekka Räsänen. 2014. *Soul of Society* (2nd. ed.). Emerald, Chicago, Chapter Exposure to online hate among young social media users. https://doi.org/10.1108/S1537-466120140000018021
[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162
[16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162
[17] B. Ribeiro, Pinghui Wang, F. Murai, and D. Towsley. 2012. Sampling directed graphs with random walks. In *2012 Proceedings IEEE INFOCOM*. 1692–1700. https://doi.org/10.1109/INFCOM.2012.6195540
[18] Manoel Horta Ribeiro, Pedro H. Calais, Yuri A. Santos, Virgílio A. F. Almeida, and Wagner Meira Jr. 2018. Characterizing and Detecting Hateful Users on Twitter. *CoRR* abs/1803.08977 (2018). arXiv:1803.08977 http://arxiv.org/abs/1803.08977
[19] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. *Lecture Notes in Computer Science* (2018), 593–607. https://doi.org/10.1007/978-3-319-93417-4_38
[20] Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. Association for Computational Linguistics, Valencia, Spain, 1–10. https://doi.org/10.18653/v1/W17-1101
[21] Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics, San Diego, California, 88–93. https://doi.org/10.18653/v1/N16-2013
[22] Matthew L. Williams and Pete Burnap. 2015. Cyberhate on Social Media in the aftermath of Woolwich: A Case Study in Computational Criminology and Big Data. *The British Journal of Criminology* 56, 2 (06 2015), 211–238. https://doi.org/10.1093/bjc/azv059 arXiv:http://oup.prod.sis.lan/bjc/article-pdf/56/2/211/7451319/azv059.pdf
[23] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. arXiv:cs.LG/1901.00596