Detecting Hateful Users on Twitter with Graph Machine Learning

Francesco Salvi, Giacomo Orsi
Department of Computer Science, EPFL, Switzerland

Abstract—**Hateful speech on online platforms is a prominent and widespread issue that can lead to real-world violence and discrimination, making its mitigation crucial for creating safe online environments. In this project, we approach this problem by implementing automated systems that detect hateful users with the help of Graph Machine Learning. We focus on a retweet graph collected in October 2017, with a sample of users being labeled as hateful or normal. We study the network by analyzing its degree and feature distributions, their correlations with labels, and the occurrence of triadic motifs, and we compare the graph with theoretical random models. Subsequently, we train various predictive models on the task of node prediction, including logistic regression, label propagation, GraphSAGE, and Graph Attention Networks (GAT). Our results indicate that GraphSAGE performs the best in terms of F1 score, outperforming all baselines and demonstrating its effectiveness in generalizing to unseen nodes and running on large graphs. Overall, our findings highlight the potential of Graph Machine Learning for fighting hateful speech on social media platforms.**

I. Introduction

Hateful speech is a prominent issue that concerns all online platforms, which due to the anonymity and accessibility afforded by the internet provide a breeding ground for the spread of violence and extremism. Without moderation, hateful content can incite real-world discrimination, harassment, and bullying, with severe psychological and emotional impacts on the victims. Despite implementing measures to combat hate speech, social media platforms still struggle to effectively curb its spread due to the lack of resources needed to carry out adequate oversight for the sheer volume of content generated daily.

In this project, we make an attempt to mitigate this problem, by implementing automated systems that detect hateful users with the help of Graph Machine Learning. We work with the dataset introduced in Ribeiro et al., 2018, containing a sample of Twitter's retweet graph made of 100 386 users and 2 286 592 edges, along with features extracted from the 200 most recent tweets of each user. The data were collected during a single week of October 2017, using an algorithm that unbiasedly estimates the out-degree distribution of nodes. A sample of 4972 nodes was then annotated as hateful or normal, resulting in a total of 544 hateful users and 4427 normal users.

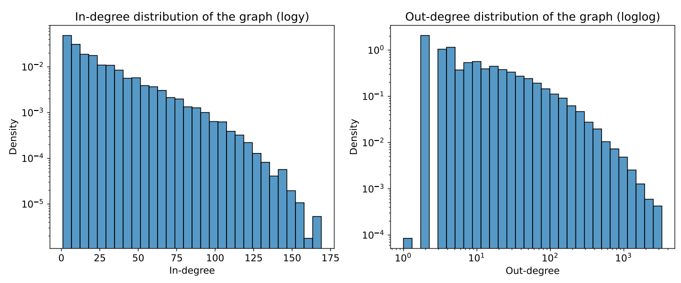
We start by exploring the data in Section II, analyzing the degree distributions, the distribution of features, and their correlation with the labels. We also extract additional insights through the use of motifs, studying the occurrence of triads in the network. In Section III, we then move to the main predictive part, casting the problem as an instance of binary node classification. We explore simple baselines and more sophisticated models, including logistic regression, label propagation, GraphSAGE, and Graph Attention Networks (GAT). We evaluate the models using metrics such as accuracy, precision, recall, and F1 score. Finally, we conclude in Section V by summarizing our findings and discussing possible future directions.

II. Exploration

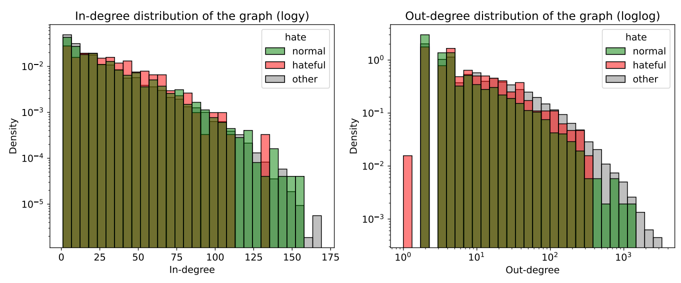
A. Initial analyses

As a preliminary step of analysis, we study the degree distribution of the network and the distribution of its features, looking for interesting patterns that we can leverage later on for the exploitation part.

Figure 1 shows the degree distribution of the network, separately by in-degree and out-degree. In our graph, a bit counterintuitively, a directed edge $A \rightarrow B$ means that user B retweeted a post of user A. We observe from the total distribution (cf. Figure 1a) that the in-degree trend approaches an exponential decay, while the out-degree is closer to a power-law distribution. Both those behaviors reproduce what is normally found in social media. In fact, it is reasonable to expect that the number of retweets received by a user, expressed by their out-degree, spans different orders of magnitude, because the



(a) Total



(b) By label

Fig. 1: Degree distribution of the retweet graph. In-degrees are plotted on a log-linear scale, while out-degrees are plotted on a log-log scale.

network includes prominent users such as celebrities that are retweeted by a very large number of people. On the other side, the number of retweets carried by individual users only grows to a certain extent, and hence decreases rapidly without a long tail. By analyzing the distribution conditional on the label (cf. Figure 1b), we find that, somewhat surprisingly, hateful users seem to lie more at the center of the distribution, avoiding very low and very high values.

To investigate more in-depth the characteristics of different types of users, we then analyze the distribution of node features conditional on the labels, which are reported in Figure 2. Again, we observe patterns that are commonly found on social networks, with most variables following normal or lognormal trends. In particular, the number of followers, followees, favorites (the predecessor of the *heart* button on Twitter), mentions, and lists of a user all follow a lognormal distribution; while the betweenness and eigenvector centralities, average sentiment, subjectivity, and number of tweets follow a normal distribution, the latter cut at 200 because that is the maximum number of tweets considered for each user. Finally, the number of quotes and the average number of bad words per tweet (i.e. *baddies*) follow an exponential distribution with a negative exponent. When taking into account the labels, we discover some interesting patterns: hateful users tend to express more negative sentiments,

Model	#edges	clustering coeff.
Original	2287k	0.057
Erdős–Rényi	2273k	0.002
Configuration	2287k	0.002
Scale-free	2006k	0.253

Table 1: Clustering coefficients for the random network models fitted to the original graph.

have a higher number of bad words per tweet, and write more subjective texts. Additionally, they seem to have higher centrality values and more favorites than the rest of the users, but generally appear in fewer lists.

B. *Random network models*

To gain a better understanding of the theoretical properties of our graph, we compare it with several random network models. We experiment with the following random models:

- **Erdős–Rényi** with $p = 2.27 \times 10^{-4}$, chosen by imposing the expected number of edges to be the actual number of edges in the retweet graph.
- Directed **Configuration Model**, fitted jointly to the in-degree and out-degree distributions.
- A **Scale-free** model. Since we have to deal with a directed graph, we cannot use standard small-world benchmarks such as the Watts-Strogatz and Barabasi-Albert models, and our options are rather limited. We use therefore the method introduced in Bollobás et al., 2003, adapting preferential attachment to the directed case. We choose as parameters $\alpha = 0.01$, $\beta = 0.95$, and $\gamma = 0.04$, in order to maximize the similarity in terms of number of edges and degree distributions.

The resulting degree distributions are shown in Figure 3. As expected by design, the configuration model completely mimics the distribution of the original graph, both for in-degrees and out-degrees. Erdős–Rényi, instead, completely fails to approximate the original graph, because its binomial distribution is not a good fit. Finally, the Scale-free model does reasonably well for the out-degree distribution, but is also completely off for in-degrees. Its generative model, in fact, assumes a power law for both distributions, while in our case the in-degree distribution is exponential.

Additionally, Table 1 reports the clustering coefficients of the generated networks. Again, we find that none of the random networks is a good fit for the retweet graph. The original network, in fact, has a much higher clustering coefficient than the Erdős–Rényi and the Configuration models, which add edges completely at

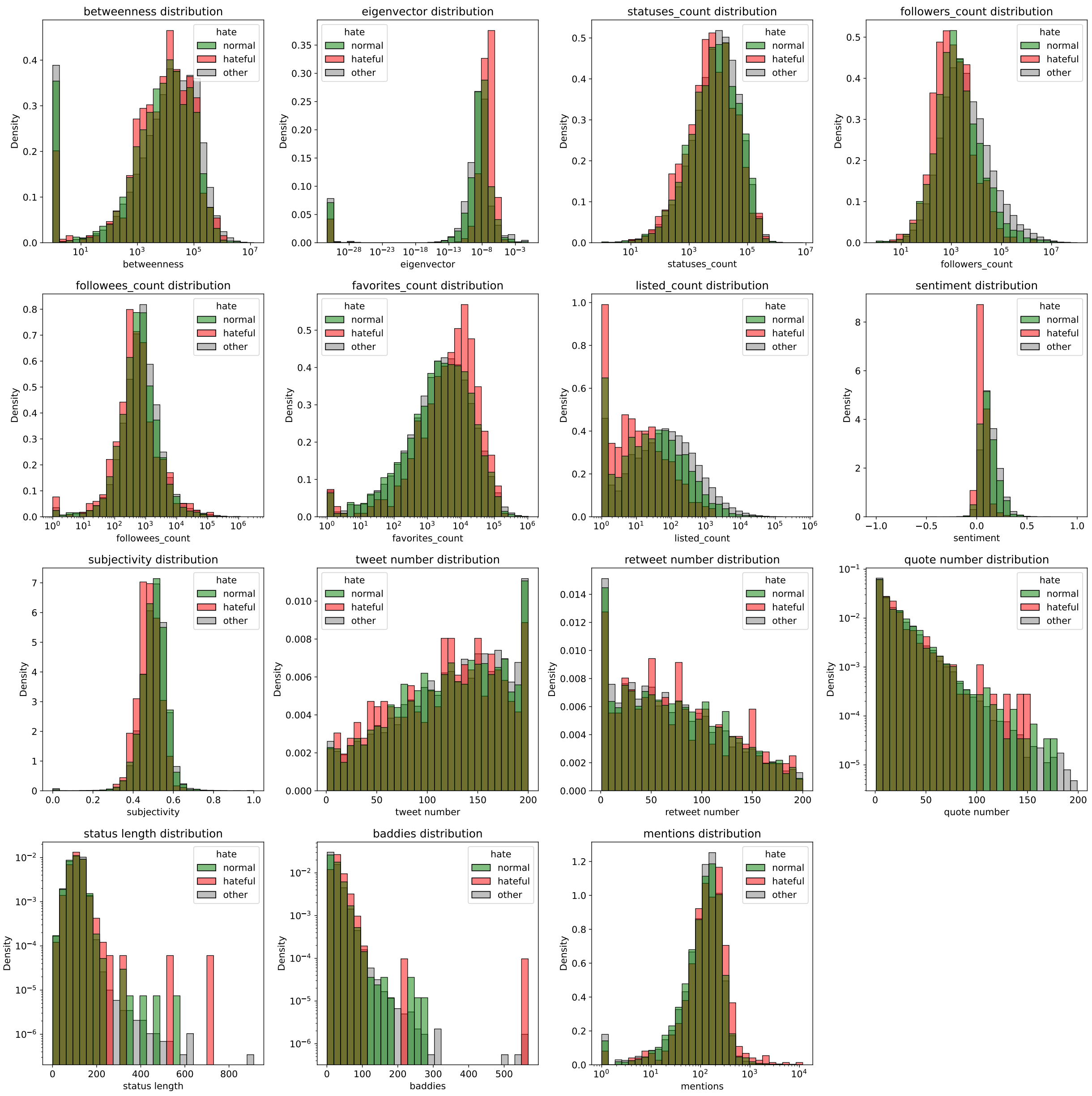


Fig. 2: Distribution of node features in the retweet graph.

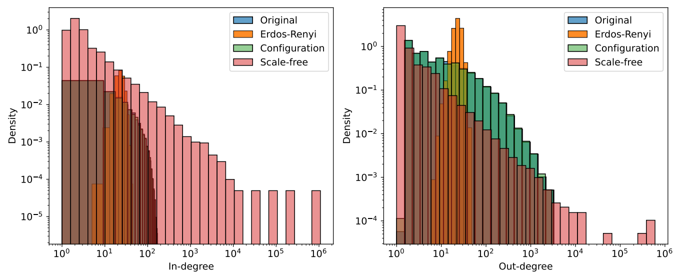


Fig. 3: Degree distributions of random network models fitted to the original graph.

random and therefore do not exhibit the typical clustering pattern of real-world networks. However, the clustering coefficient of the retweet graph is also much lower than the one of the Scale-free graph, which is built using preferential attachment. Interestingly, the fact that only one of the degree distributions follows a power law makes the original graph lie somewhat in the middle between random lattices and scale-free networks, without completely fitting any of the two. Out of the models that we discussed, the best one seems to be the Configuration model, because despite having a very different clustering coefficient it fits well both degree distributions.

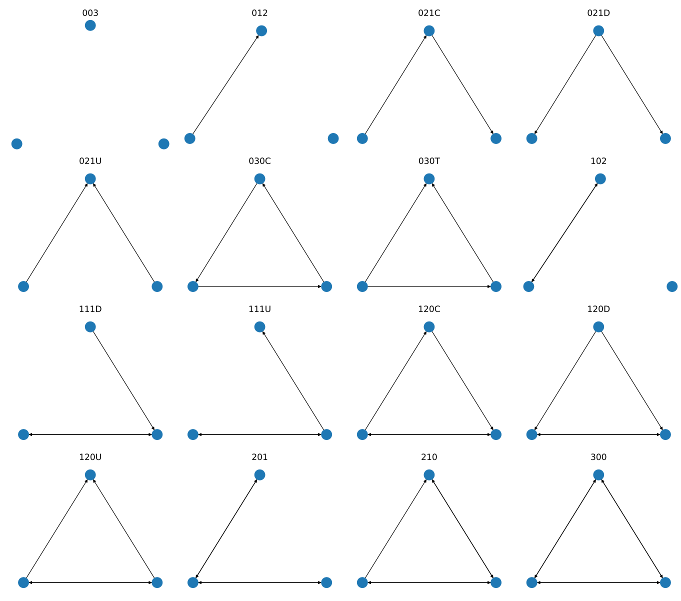


Fig. 4: Topology of possible motifs (triads).

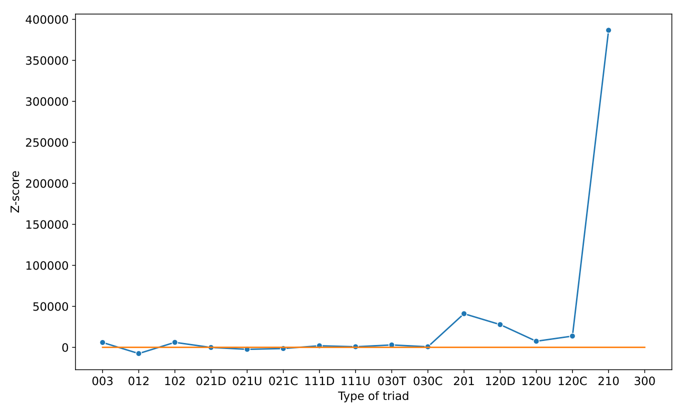


Fig. 5: Z-score per motif, with respect to the null model, averaged over 8 simulations. A score for the triad 300 could not be computed because it did not appear even once over the 8 simulations of the null model, as opposed to the original graph where it appears 108 573 times.

C. *Motifs*

After finding that our network does not entirely reproduce the scale-free behavior in terms of clustering coefficient, as discussed in Section II-B, we decided to take a deeper dive into investigating its connectivity properties. To do that, we focus on triadic motifs, i.e. local interconnection patterns that characterize sets of three nodes. Studying such fundamental structures, in fact, can help us uncover organizational mechanisms and recurrent schemes, ultimately understanding better the peculiarities of the graph. A topology of all the possible triadic motifs, or triads, is reported for clarity in Figure 4. We count the occurrence of each such motif in the whole graph, looping through each possible triangle. To assess the statistical significance of the counts, we repeat the same procedure using a Directed Configuration Model, the random model that we found to have the best fit to our graph, performing this step 8 times to get a distribution of counts. Then, we compute a Z-score for each motif type by subtracting the average counts in the random graphs from the counts in the original network, and dividing the result by the standard deviation of the random counts.

The results of this process are visualized in Figure 5. We observe an incontrovertible predominance of ”closed” triads, i.e. the ones where arrows go in both directions between nodes. Note that a score could not be computed for the triad 300 because it did not appear even once over the 8 simulations of the configuration model, as opposed to the original graph where it appears 108 573 times, being therefore very significant. Remarkable spikes are observed also for the triads 201 and 210, again showing a high degree of reciprocity for the retweet relation. On the opposite, the motif 012 is less frequent than in the null model, confirming that single-oriented edges are unlikely to happen. Those findings reproduce behaviors that are normally found on social networks, where users retweeting each other are likely to be connected and hence retweeted by the counterpart, and which are commonly characterized by a strong triadic closure.

III. Prediction models

In this section, we describe different approaches to build a predictive model for the node label, with the goal of automatically detecting hateful users. As described in Section II, the graph contains 100 386 users, of which only 4972 (4.95%) are labeled by human evaluators: 544 (10.94%) are labeled as *hateful* and 4427 (89.06%) as *normal*.

The small portion of labeled users and the high imbalance between the two classes make the problem of predicting node labels particularly challenging, but also very interesting. In fact, a graph with many unlabeled nodes is a common scenario for many real-world applications, and the same is true for unbalanced classification.

In addition to the features analyzed in Section II, we also consider the following features provided by the authors of the original dataset:

1) Empath categories: Empath (Fast et al., 2016) is a tool that classifies text into a set of lexical categories. We use the 200 pre-computed Empath categories as features.

2) Glove embeddings: Glove (Pennington et al., 2014) is a technique to obtain word embeddings. We use 300-dimensional Glove embeddings as features, averaging for each user the embeddings of all their tweets.
3) Node centrality features: we add degree, betweenness, closeness, and eigenvector centrality as features.

A. Methods

1) Baseline – Logistic Regression: We use logistic regression as a baseline model. In this case, the model is not aware of the neighbors of each node, with information about their connectivity being only brought by the centrality features.

2) Label propagation: Label propagation (Zhu & Ghahramanih, 2002) is a semi-supervised learning algorithm that leverages the network structure to propagate labels from labeled nodes to unlabeled nodes. This approach assumes that nodes connected in the graph are likely to share similar labels. We initialize the labels of the labeled nodes and iteratively update the labels of the unlabeled nodes, based on the labels of their neighboring nodes. This iterative process continues until convergence.

3) GraphSAGE: GraphSAGE (Graph Sample and Aggregate), introduced by Hamilton et al., 2017, is a graph representation learning algorithm that learns node embeddings by sampling and aggregating information from the neighborhood of each node. It is an inductive framework, meaning that it can generalize to unseen nodes. The algorithm is composed of two steps: sampling and aggregation. In the sampling step, we sample for each node v a fixed-size set of neighbors $\mathcal{N}(v)$, and we aggregate the features of the sampled neighbors to produce a representation of v. In the aggregation step, we then use a neural network to learn a function that maps the resulting representations into a new representation for each node. The algorithm is trained to minimize the loss between the predicted labels and the true labels of the labeled nodes. The learned function can then be used to predict the labels of the unlabeled nodes.

4) Graph Attention Networks (GAT): GATs (Veličković et al., 2018) leverage the attention mechanism to assign importance weights to the neighboring nodes during the aggregation step. This allows the model to focus on more informative nodes and capture complex relationships in the graph.

IV. Results

A. Data

We perform our experiments on the dataset using PyTorch Geometric (Fey and Lenssen, 2019). Nodes are partitioned in a 70-30 train-test split. Unlabeled nodes are always kept in the graph, and their features are used by our GNN models to produce the embeddings of all nodes, but their labels are ignored during training.

As the input graph is directed, we investigate our methods both using the original graph and an undirected version of it. We report in the following all the results, even though all the methods perform better on the undirected graph.

B. Evaluation metrics

Since the dataset is highly imbalanced, we use the F1 score as the main comprehensive measure of model performance, while reporting for completeness also accuracy, precision, and recall. Note that in a hypothetical real-world scenario where Twitter would be interested in identifying hateful users with our framework, optimizing over precision or recall might be more relevant than using the F1 score. For instance, if Twitter plans on employing human evaluators to review users identified as hateful by the model, it might be better to optimize recall, to make sure that they do not miss any hateful users. On the other hand, if Twitter plans on automatically banning users detected as hateful, it might be better to optimize for precision, to avoid banning normal users by mistake.

C. Model parameters and reproducibility

We have experimented with different parameters for each model, and we report in this section the configurations that produced the best results.

- Logistic Regression: we use the default parameters of the `sklearn` implementation.
- Label propagation: we use $\alpha = 0.8$
- GraphSAGE: we use 128 hidden channels, two layers, a dropout probability of 0.4, and a learning rate of 0.01, over 10 epochs. Each node is sampled with 25 neighbors in the first layer and 10 neighbors in the second layer. We use aggregate by performing averages and we use ReLU as activation function.
- GAT: we use 128 hidden channels, two layers, 4 attention heads, a dropout probability of 0.4, and a learning rate of 0.01, over 10 epochs.

All the GNN models are optimized using Adam (Kingma & Ba, 2015), with standard parameters. We enforce the same random seed for all the experiments, to ensure reproducibility.

D. Results

Table 2 shows the results obtained on the test set using the directed graph, where an edge $A \to B$ means that user B retweeted a post of user A. Table 3 shows the results obtained on the test set using the undirected graph, where an edge $A \rightarrow B$ means that either user B retweeted user A or user A retweeted user B. Logistic Regression is reported on both tables as a baseline, even though it does not take into account the edges of the graph and thus is agnostic to the directionality.

	Accuracy	Precision	Recall	F1
Logistic	0.9023	0.5507	0.4967	0.5223
Label Propagation	0.9037	0.5714	0.4183	0.4830
GraphSAGE	0.9177	0.6184	0.6144	**0.6164**
GAT	0.9191	0.7262	0.3987	0.5148

Table 2: Results obtained on the directed graph

	Accuracy	Precision	Recall	F1
Logistic	0.9023	0.5507	0.4967	0.5223
Label Propagation	0.9184	0.6391	0.5556	0.5944
GraphSAGE	0.9100	0.5648	0.7124	**0.6301**
GAT	0.9226	0.7416	0.4314	0.5455

Table 3: Results obtained on the undirected graph

Interestingly, we observe that the label propagation algorithm performs much better in the undirected graph (F1: 0.5944) than in the directed graph (F1: 0.4830).

GraphSAGE outperforms all the other models in terms of F1 score. The GAT model has the best accuracy and precision, but the worst recall. This means that the GAT model is more conservative in predicting the label of a node, but it can be employed in scenarios where we primarily care about avoiding the misclassification of normal users as hateful. For our task, however, GraphSAGE is a particularly good fit, because it has been shown to be effective in generalizing to unseen nodes and it is suitable to run on very large graphs. This dataset is a tiny subset of the whole Twitter graph, which is instead composed of billions of nodes and edges.

In addition, we observed that GATs are very sensitive to changes in the hyper-parameters. A slight change in the number of layers, the number of attention heads, or the dropout probability can lead to significant variations in the performance of the model. On the other hand, GraphSAGE is more robust to a change in the hyper-parameters. This is probably due to the fact that GraphSAGE is a simpler model than GAT, and therefore it is less prone to overfitting. Furthermore, we observed *oversmoothing* in the GAT model when we increased the number of layers. This means that the model produced node embeddings that were very similar to each other, and therefore was not able to distinguish between different nodes.

Figure 6 shows the embeddings of the nodes obtained with GraphSAGE, projected in a 2D space using t-SNE (van der Maaten & Hinton, 2008). We can see that the embeddings of the nodes with the same label are clustered together, which means that GraphSAGE is able to learn a representation of the nodes that is useful for the classification task.

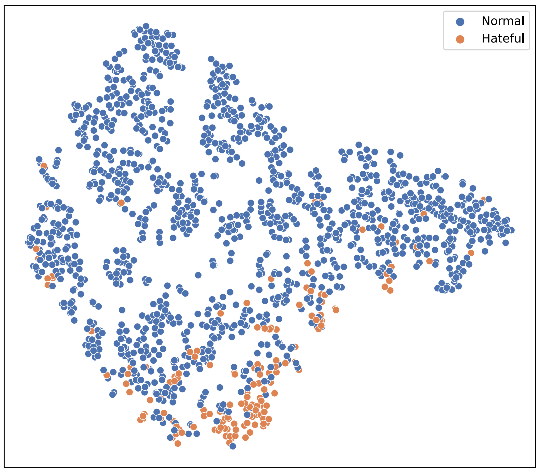


Fig. 6: Embeddings of the nodes obtained with GraphSAGE, projected in 2D using t-SNE.

V. Conclusion

In this report we explored the problem of identifying hateful users on Twitter. We used a dataset of tweets and retweets and we built a graph where nodes are users and edges are retweets. We used this graph to train Graph Neural Networks to predict whether a user is hateful or not. We compared the performance of the GNNs with simple baselines, both using or not the graph structure to make predictions. We found that GraphSAGE performs the best, outperforming all the other models in terms of F1 score.

References

Ribeiro, M., Calais, P., Santos, Y., Almeida, V., & Meira Jr., W. (2018). Characterizing and detecting hateful users on twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, *12*(1). https://doi.org/10.1609/icwsm.v12i1.15057

Bollobás, B., Borgs, C., Chayes, J., & Riordan, O. (2003). Directed scale-free graphs. *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 132–139.

Fast, E., Chen, B., & Bernstein, M. S. (2016). Empath: Understanding topic signals in large-scale text. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 4647–4657. https://doi.org/10.1145/2858036.2858535

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Zhu, X., & Ghahramanih, Z. (2002). Learning from labeled and unlabeled data with label propagation.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. *Proceedings of the 6th International Conference on Learning Representations*.

Fey, M., & Lenssen, J. E. (2019). Pytorch geometric: A fast graph representation learning library and beyond. *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. http://arxiv.org/abs/1412.6980

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(86), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html