A Social-Aware Deep Learning Approach for Hate-Speech Detection

George C. Apostolopoulos, Panagiotis Liakos(✉), and Alex Delis

University of Athens, 15703 Athens, Greece
g.c.apostolopoulos@gmail.com, {p.liakos,ad}@di.uoa.gr

Abstract. Despite considerable efforts to automatically identify hate-speech in online social networks, users still face an uphill battle with toxic posts that seek to sow hatred. In this paper, we initially observe that there is a great deal of social properties transcending both hateful passages and respective authors. We then exploit this observation by *i)* developing deep learning neural networks that classify online posts as either hate or non-hate based on their content, and *ii)* proposing an architecture that may invigorate any such text-based classifier with the use of additional social features. Our combined approach considerably enhances the classification accuracy of previously proposed state-of-the-art models and our evaluation reveals social attributes that are the most helpful in our classification effort. We also contribute the first publicly-available dataset for hate-speech detection that features social properties.

Keywords: Social features · Deep learning · Twitter · User profile

1 Introduction

With more than half of the world's population actively using social media, much of the communication among individuals is now taking place online [8]. Massive online interactions on *social networks* (*SN*s) have incentivized malicious players to exploit pertinent infrastructures in pursuit of illicit actions. The well-documented correlation between violent acts and *SN*s inflammatory speech have necessitated the need for censoring such content [14]. In the past, the problem of automatic hate speech detection has been predominantly approached as a supervised document classification task. Such efforts mainly entail: *i)* traditional approaches employing feature engineering [4], and *ii)*deep learning approaches that automatically learn features from raw data using neural networks [16].

Feature engineering approaches have relied in the use of *surface* features including *dictionaries* of insults and swear-words [10], *N-grams* [6], as well as *URL*s, mentions, hashtags and capitalization [3]. Besides simple *surface* features, existing methods have employed sentiment analysis to identify negative polarity [9], *part of speech (POS)* to detect the role of each word in the context of a sentence [2], and meta-information related to the author of a passage and her past activity [7,15]. Using the above features, approaches have deployed *SVM* and *Naive Bayes* classification algorithms to train models for hate-speech detection [3,7].

In [1], the *CNN* and *LSTM* network architectures are examined along the use of *random* and *GloVe* word embeddings [12] to obtain vector representations of terms. [1] shows that deep neural network architectures outperform traditional classifiers such as Logistic Regression, SVM and Gradient Boosted Decision Trees, offering greater classification accuracy. Although linear classifiers and deep learning models have certainly helped attain noteworthy accuracy in identifying hate-speech, these methods have strictly focused on the textual content posted by online users, without considering whether *social properties* can improve the performance of the proposed models.

In this paper, we advocate that using social properties is very beneficial to the task of identifying hate-speech in social media content. We focus on `Twitter` and apply deep learning methods that use *both* the post *content and* the *social properties* related to the user and the post itself. Our main contributions are:

– We propose a deep learning model that exploits the network structure, as well as individual social reactions, e.g., likes, to help identify hate-speech.
– We generate and make publicly available an annotated dataset of `Twitter` posts that do or do not feature hate-speech and is generated by the activity of different individuals. Previous available datasets are *biased* as they involve activity of a very limited number of users, and lead to training models that may penalize a particular writing style. This is the first hate-speech related dataset to include numerous social attributes associated with each post.
– We outperform state-of-the-art methods in terms of classification accuracy through our *combined approach* and demonstrate that network relationships and social activity do enhance the accuracy of hate-speech detection models. Moreover, our approach is orthogonal to text-based hate-speech detection as our architecture can complement classifiers focusing strictly on text.

The rest of the paper is organized as follows: in Sect. 2, we introduce our text-based models and discuss how we can enhance their effectiveness with a novel social-based deep learning approach. Section 3 describes the dataset we collected. Next, we present our experiments in Sect. 4, followed by the conclusion in Sect. 5.

2 Methodology

Our approach commences with a preprocessing step whose objective is to normalize the content of our dataset. This step first removes URLs, emojis, usernames and numbers. Moreover, we split hashtags on upper case letters, as oftentimes they are used to compose sentences and we need to come up with the individual words. Also, we split words appended with slashes, and we mark punctuation repetitions and elongated words. Finally, we transform all upper case letters to lower case for consistency. As an example, given the tweet:

> *@TheLeoTerrell: Good News!!! Leo 2.0 just received message from Team Trump. Going to be used in Campaign. Cannot wait to help*

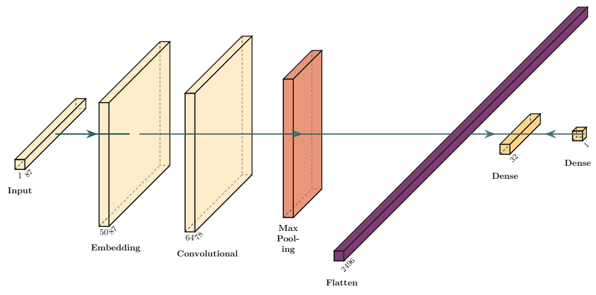


Fig. 1. Our convolutional neural network approach.

we come up with the following text which is ready for our vectorization process:

> *<user>: good news! leo <number> just received message from team trump. going to be used in campaign. cannot wait to help*

2.1 CNN Architecture

The first model we train is a CNN (Convolutional Neural Network), which is known in the literature to extract features effectively [11]. Figure 1 depicts the functional layout of the CNN in question whose components we outline below.

Input Layer: Our CNN model initially features an Input layer, to which we feed the vectors list. The maximum vector produced after the preprocessing of the tweets in our dataset has a length of 87 tokens.

Embedding Layer: The next layer of our architecture is an embedding layer, which allows words with similar meaning to have a similar representation. Word embeddings are one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems [5]. Individual words are represented as real-valued vectors in a predefined vector space. We use a pre-trained dimensional embedding called GloVe [12]. In our approach, we exploit a 50 dimensional pre-trained GloVe vector that is created using a total of 2 billion tweets, 27 billion tokens with an overall vocabulary of 1.2 million words. Our embedding layer produces a 87×50 representation.

Convolutional Layer: The output of the embedding layer is then fed into a convolutional layer that learns to extract salient features from the word embeddings. We use a $1D$ convolutional layer with 64 filters and a sliding window size of 10. These are the same initialization values used in the model of [1]. Our convolutional layer uses the rectified linear unit function for activation. This process convolves the input feature space into a 78×64 representation.

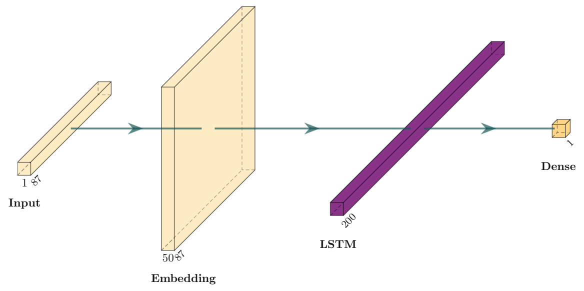


Fig. 2. Our LSTM network.

Pooling Layer: Next, we down-sample the incoming vectors using a pooling layer. Here, we use maximum pooling, which calculates the maximum, or largest, value in each patch of each feature map. This process halves the layer's input, producing a 39×64 representation.

Flatten and Dense Layers: The down-sampled vectors produced by the pooling layer are then flattened to a $1D$ array of size 39×64 with a Flatten layer. Next, follows a standard dense layer of 32 neurons. Finally, we classify the input as hate or not-hate, with the use of the sigmoid activation function, that is frequently used for binary classification.

2.2 LSTM Architecture

Our second model is based on an LSTM (Long Short Term Memory) network, and is shown in Fig. 2. The input and embedding layers are identical to our CNN approach. The vector length of our input layer is 87, defined by the length of the longest input vector in our dataset. The embedding layer is a 50 dimensional pre-trained GloVe vector, that produces a 87×50 output representation.

After the embedding layer, our model features an LSTM layer with a dropout probability. We use a bidirectional LSTM layer of 100 units to preserve information from both past and future, and better capture the context of sentences. Moreover, we set the dropout probability to be equal to 0.5. This leads to ignoring half of the input in each repetition, which is a well-known form of regularization that prevents neural networks from overfitting [13]. The size of the LSTM layer and the dropout probability have been selected after extensive experimentation and in accordance to earlier findings [13].

Finally, our LSTM based model features a dense layer that classifies the input with the use of the sigmoid activation function.

2.3 Social-Aware Deep Learning Network

Our CNN and LSTM networks perform text-based classification using the content of a user's post. Similar networks have been studied in the past and are

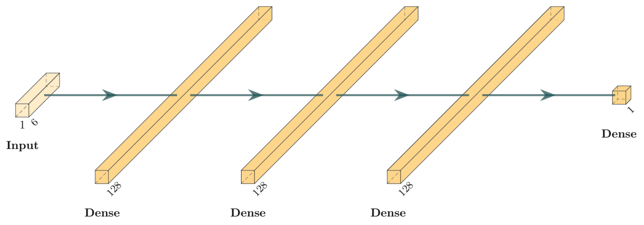


Fig. 3. Our social features based network

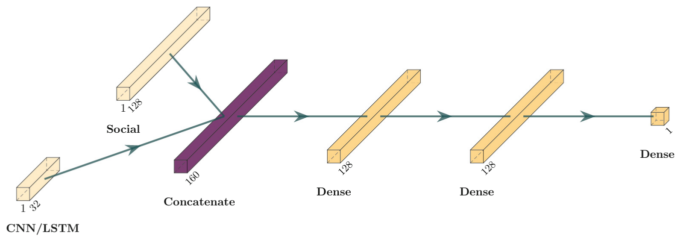


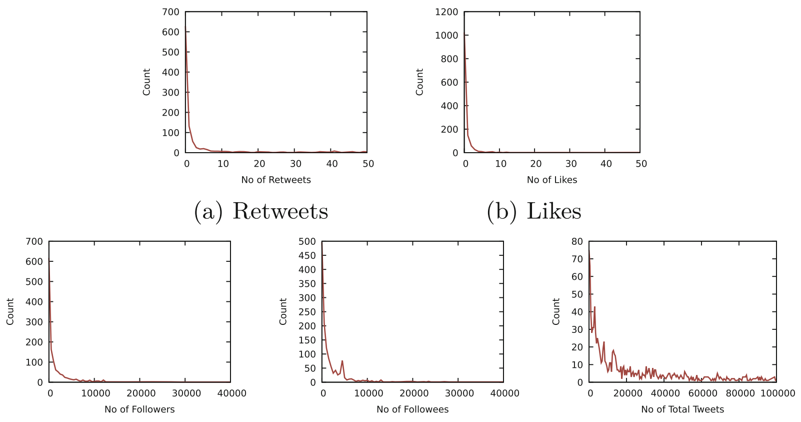
Fig. 4. Combining text-based models with our social features based networks.

shown to outperform traditional classifiers [1]. Here, we discuss how we can additionally exploit information related to the post. In particular, we enhance our deep learning approach with a second network that takes as an input additional social properties about the user and the post itself. Our working hypothesis is that this meta-information can help build more accurate models. Thus, we focus on the following six key features: *i)* user followers, *ii)* user followees, *iii)* user posts, e.g., tweets, *iv)* post shares, e.g., retweets, *v)* post likes, and *vi)* whether the post is a reply. Our goal is to uncover which of the above can enhance the performance of our text-based models. To this end, we build a neural network that comprises a series of consecutive dense non-linear layers and takes as input any combination of the above features; this network is depicted in Fig. 3.

The network of Fig. 3 is ultimately combined with our text-based models as portrayed in Fig. 4. Here, we deploy another network of hidden non-linear layers that takes as input the concatenated output of the network in Fig. 3 and any text-based models, such as those depicted in Figs. 1 and 2. This architecture allows for exploiting the social information through an agnostic –with regard to text classification– approach. Consequently, our contribution is orthogonal to previous approaches [1,16] that exclusively use the textual content of a post to detect hate-speech.

3 Dataset

Our dataset focuses on former US president, Donald J. Trump. The collection process took place just before the 2020 Presidential US elections, a time at



(a) Retweets (b) Likes

(c) Followers (d) Followees (e) Total Tweets

Fig. 5. Analysis of the social features of our dataset.

which media focused heavily on Trump, as a candidate. We followed a collection approach similar to that of [15,16]. In particular, we initially utilized the `Twitter` *API* through the `tweepy`[1] Python library to collect the tweets containing the word `Trump`. Naturally, most tweets did not feature hate speech. After a collection period of one month that ended 3 days before the elections, we had collected a total of 1,311 tweets, out of which 219 are labeled as hateful.

We also present a useful analysis of our social features in Fig. 5. We can see that most posts exhibit a small number of retweets or likes, with the former being more frequent than the latter. Moreover, our users include accounts who have neither followers nor followees as they are posting their very first tweets as well as people with significant activity and thousands of followers.

4 Experiments and Results

We implemented our models using Python `Keras` with the `TensorFlow`-backend and the `scikit-learn` library. Our implementation and dataset are publicly available.[2] For all models discussed in this section, we split the dataset into 80:20 to use 80% with cross-validation to tune learning epochs. We test the optimized model on the 20% held-out data and report average results of multiple executions using weighted precision, recall and *F1*-scores, as in [1,16]. The evaluation of our models answers the following key questions: (i) Does the use of social features improve the effectiveness of our models? (ii) What are the social features that prove to be most helpful?

We begin with an investigation of the potential of *social* features to enhance the performance of our models. There are a total of 63 combinations of features

[1] https://www.tweepy.org/.
[2] https://github.com/giorgos-apo/hate-speech-detection-using-user-attributes/.

Table 1. Results of our CNN model when enhanced with social features.

	User features	F1 Score	Recall	Precision
1	`user_followers, tweet_is_reply`	0.8342	0.8463	0.8387
2	`user_followers, user_total_tweets`	0.8327	0.8494	0.8404
3	`tweet_retweets, tweet_likes, tweet_is_reply`	0.8301	0.8517	0.8339

Table 2. Results of our LSTM model when enhanced with social features.

	User features	F1 Score	Recall	Precision
1	`tweet_likes, user_followers, user_total_tweets`	0.8405	0.8456	0.839
2	`user_followers, user_following`	0.8383	0.8471	0.8369
3	`user_followers, tweet_is_reply`	0.8318	0.841	0.8266

that we can utilize to enhance the performance of our models. We experiment with all and report in Table 1 the ones with the most significant gains, when combined with our CNN model. We obtain the best results when combining the CNN model with a network that takes as input the author's number of followers and whether the tweet is a reply or not. This combination improves the *F1*-score by almost 2 points.

Table 2 lists the combinations of social features that are most effective when combined with our LSTM network. We see that our combined approach manages again to increase the classification accuracy of this network even more than the case was with the CNN model. The most notable improvement is achieved when combining the number of user followers with the user's total number of tweets and the tweet's likes. This synthesis helps us increase the F1-score of the LSTM network by more than 4 points.

Overall, our combined approach shows significant improvement with regards to all precision, recall and accuracy for both our text-based deep-learning models. We note here that our architecture allows for any text-based classifier to be plugged into our model. We have experimented using a CNN and an LSTM network that are trained using the content of user posts. However, our contribution is orthogonal to text-based hate-speech detection and can be exploited by any such approach. More importantly, we show here that the use of meta-information regarding social properties of posts and their respective authors is very beneficial for the task of hate-speech detection. The results of Tables 1 and 2 quantify the importance of various contributing social features as they clearly point out that the number of user followers is very helpful when it comes to identify hate-speech content.

5 Conclusion

We present a novel approach that enhances classification accuracy of hate-speech detection models through the use of *social properties* related to content posted online. Our model exploits meta-information for each post we want to have classified as hate or non-hate, in addition to the actual text of the post. Our experimentation ascertains the importance of social properties in significantly improving the effectiveness of all text-based models we have deployed. Moreover, we investigate and quantify the importance of various social features with regard to detecting hate-speech. Our findings point out that the number of followers a user has is a very helpful property to consider when building hate-speech detection classification models. Combinations of social properties that include this feature improve our text-based classifiers significantly with regards to *F1*-score. Last but not least, we make available a new dataset, complementing existing ones by considering social features instead of exclusively focusing on text content.

References

1. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: WWW 2017 (Companion), pp. 759–760. Perth, Australia (2017)
2. Burnap, P., Williams, M.L.: Cyber hate speech on twitter: an application of machine classification and statistical modeling for policy and decision making. Policy Internet **7**(2), 223–242 (2015)
3. Davidson, T., Warmsley, D., Macy, M.W., Weber, I.: Automated hate speech detection and the problem of offensive language. In: ICWSM 2017, pp. 512–515 (2017)
4. Fortuna, P., Nunes, S.: A survey on automatic detection of hate speech in text. ACM Comput. Surv. **51**(4), 1–30 (2018)
5. Deep Learning for Natural Language Processing. Apress, Berkeley (2018). https://doi.org/10.1007/978-1-4842-3685-7_5
6. Greevy, E., Smeaton, A.F.: Classifying racist texts using a support vector machine. In: SIGIR 2004, pp. 468–469. Sheffield, United Kingdom (2004)
7. He, J., Liu, H.: Bi-labeled LDA: inferring interest tags for non-famous users in social network. **5**, 27–47 (2020)
8. Liakos, P., Papakonstantinopoulou, K.: On the impact of social cost in opinion dynamics. In: Proceeding of the Tenth International Conference on Web and Social Media, Cologne, Germany, pp. 631–634 (2016)
9. Liu, S., Forss, T.: Combining N-gram based similarity analysis with sentiment analysis in web content classification. In: IC3K 2014, p. 530–537. Rome, Italy (2014)
10. Liu, S., Forss, T.: New classification models for detecting hate and violence web content. In: IC3K 2015, pp. 487–495 (2015)
11. Ordóñez, F.J., Roggen, D.: Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. Sensors **16**(1), 115 (2016)
12. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP 2014. pp. 1532–1543. Doha, Qatar (2014)
13. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
14. Vosoughi, S., Roy, D., Aral, S.: The spread of true and false news online. Science **359**(6380), 1146–1151 (2018)

15. Waseem, Z., Hovy, D.: Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In: Proceedings of the NAACL Student Research Workshop, pp. 88–93. ACL, San Diego, CA, June 2016
16. Zhang, Z., Robinson, D., Tepper, J.: Detecting hate speech on Twitter using a convolution-GRU based deep neural network. In: ESWC 2018, pp. 745–760 (2018)