JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Deep Learning Approach for Hate Speech Spreaders Detection using Statistical and Contextualized Embeddings

[1]Dr. T. Raghunadha Reddy, [2]M. Shashi Preetham, [3]K. Sree Vasini, [4]A. Rajesh
[1]Associate Professor, [2, 3, 4] Student
[1, 2, 3, 4] Department of CSE, Matrusri Engineering College, Hyderabad, India

Abstract : As the social network usage increases, hate speech is also increasing. An automated method to detect hate speech spreaders is needed. There have been many methods but each has its own limitations. Some of them are restricted to language, some need high computational power, and some does not consider the context. So, we tried various statistical and contextual embeddings paired with standard deep learning classifiers. This experiment carried out with the PAN competition 2021 dataset of hate speech spreaders detection task. For English language dataset, the Ensemble model using CNN obtained best training accuracy of 0.70 for hate speech spreaders detection and best testing accuracy of 0.73 for hate speech spreaders detection when compared with other models. For Spanish language dataset, the Ensemble model using CNN obtained best training accuracy of 0.90 for hate speech spreaders detection and the Trained Embeddings + CNN obtained best testing accuracy of 0.80 for hate speech spreaders detection when compared with other models.

***IndexTerms* - Hate Speech Spreaders, Word Embeddings, CNN, DistilBERT, TFIDF, Bi-LSTM.**

I. INTRODUCTION

Hate Speech is any kind of post on social media that spreads negative feelings, bully with offensive comments, harass based on race, color, community or nationality. It has been gradually increasing over the last decade. It was an overhead for humans to look at each tweet and classify if an author was a hate speech spreader. Therefore, an automated detection method was an increasing need. The hate speech could be sarcastic and in any language. Several researchers proposed approaches based on machine learning algorithms such as Support Vector Machine (SVM), Naive-Bayes, etc. However, to capture the context for more accurate classification, deep learning techniques such as BERT, LSTMs are used by the researchers. The word embeddings have to be improvised for betterment of accuracy and performance of models.

In this work, the experiment carried out with different models such as DistilBERT + Bi-LSTM, Word2Vec with CNN, Word2Vec embedding + TF-IDF weights + CNN, Word2Vec + TFIDF + LR, Word2Vec + TFIDF + SVM, DistilBERT with TF-IDF concatenated embeddings + Bi-LSTM, DistilBERT, TF-IDF concatenated embeddings + ANN, Trained Embeddings + CNN, Trained char-level embeddings + CNN, and Ensemble model using CNN by training at author level and tweet level for hate speech spreaders detection. Among these models, the Ensemble model using CNN attained best training accuracies for English dataset of hate speech spreaders detection. For testing accuracy, Trained Embeddings + CNN attained best accuracy for Spanish language dataset and Ensemble model using CNN attained best accuracy for English language dataset.

This work is organized in 7 sections. Section 2 describes the existing works proposed by the researchers for hate speech spreaders detection. The dataset characteristics are presented in section 3. The methodology followed in the proposed models is explained in section 4. The section 5 describes the models proposed in this work. The experimental results of proposed models for hate speech spreaders detection are presented in section 6. The section 7 concludes this work.

II. RELATED WORKS

The works in [1] are based on the embeddings generated by using RoBERTa and other representations using TF-IDF that are given to the classifier Linear SVM. They also tried the ensemble model and different models are developed for different languages. They achieved an accuracy of 0.67 in English and 0.80 in Spanish.

The authors in [2] fine-tuned BERT and produced embeddings even for emojis and other symbols. They concatenated all these embeddings into a single sample and gave it to Logistic Regression Classifier. They achieved the best accuracy in English, i.e., 0.75. [1] and [2] are on tweet level.

In [3], they have used BERT and TF-IDF but TF-IDF embeddings with Support Vector Machine gave better results, i.e., an accuracy of 0.67 in English and 0.81 in Spanish. They have also tried BERT embeddings with CNN and got an accuracy of 0.66 in English.

The authors in [4] tried various machine learning models like SVM, Naive Bayes, KNN, Logistic Regression and different deep learning techniques such as LSTM, Bi-LSTM, BERT but satisfied with the performance of Multinomial Naive Bayes.

Fake News Spreaders Identification is similar to our project. Those works could be useful in exploring different approaches. In [5], they have focussed more on TF-IDF of char and word n-grams and linear SVM as classifier. They experimented with various n-gram ranges and consideration of emoji as feature. In [6], an ensemble model is built that consists of models that work on n-grams and other on statistical features derived from social posts. These statistical features may consider the count of emojis, hashtags, retweets, length of tweets, etc. The works like [5], [6] depict that n-grams paired with TF-IDF weights are effective on author profiling tasks. Many previous models used Linear SVM as the classifier.

III. DATASET CHARACTERISTICS

In this experiment, the dataset is taken from PAN-2021 Hate Speech Spreaders Detection task [7]. The dataset consists of 300 author's data in XML format in two languages such as English and Spanish. In each language, Training data consists of 200 authors and test data consists of 100 authors. Each author has 200 tweets. The characteristics of dataset are presented in Table 1.

Table 1: The dataset description

Features / Classes	Class 0	Class 1
Training Profiles	100	100
Testing Profiles	50	50
Number of unique tweets in each profile	200	200
Unique words count	20280	19298
Total Emojis count	8465	7201
Unique Emojis Count	531	540
Uppercased words count	44316	42135
Uppercased phrases count	1026	1243
URL's count	8556	6759
Hashtags count	3644	3290
@mentions count	17250	17585
Retweets count	7731	6159

Thus, based on the tweets of each author, the model has to classify if he is a Hate Speech Spreader or not.

IV. METHODOLOGY

This section discusses the procedure and techniques that are used throughout the process of developing the solution.

4.1 Pre-processing of text

Removal of XML tags

The tweets are in the XML format. The first step is to remove the XML tags and retrieve the tweets from each file.

Removal of unwanted symbols, characters and contractions

Our first assumption was that emojis would help in classification. But the emojis are not abundant in the data to be trained as a feature. The hashtags are used in both hate speech tweets and non-hate speech tweets, so it may not give good results because of its irrelevant presence in the text. The URLs and hyperlinks are removed.

No stop word is removed in English. In Spanish, only negation stop words are preserved, thinking of the information they add to the surrounding context. Contractions like "isn't" are expanded as "is not", so that they are uniform throughout the data.

Casing of the letters: Preserving the case

This helps us to attain more information, e.g., 'US' is a country and 'us' refers to people.

We have used pretrained 'bert-base-multilingual-cased' (considers the case and applicable to various languages) and freeze its pretrained layers and added extra ANN layers for fine-tuning. This experiment was done on 'tweet level' (if it is hate speech tweet or not) on English training dataset. The major problem was it was computationally very expensive and it had been very difficult to fine-tune the BERT and train it on this dataset. The accuracy was quite low. We thought the cause of this problem was it was pretrained on a different data, where it included formal words unlike in this dataset where there is presence of new informal words.

Thus, preservation of case makes the task of learning sophisticated, because the no. of features(words) to learn increases drastically, i.e., 'This' and 'this' will be considered different. This may not be that significant for the current dataset for the classification.

Lower casing all the text

This makes the training simpler in terms of number of features and uniformity. Therefore, we proceeded with this.

Lemmatization

We chose lemmatization over stemming, because it seemed to produce the words in standard form.

4.2 Building Vocabulary

After pre-processing and tokenization, the words with at least 2 times occurred in a dataset are added to the vocabulary. The single letter words are not considered.

This vocabulary is used as a reference. In training or testing, while building embeddings the words that are not in vocabulary are omitted.

4.3 Building Model

In this work, the experiment performed in two ways such as training at author level and training at tweet level.

4.3.1 Training on tweet level

This may seem equivalent to sentiment analysis, where each the tweets of all authors is considered as a training sample. Each tweet is given label of author (as in the dataset, the labels are assigned to author not the tweets).

The thing to note here is, this may seem simpler to implement but it is not semantically correct according to our project and also it is not good to explicitly assign the labels for tweets the labels of their author. It is because, a hate speech spreader may not contain all his tweets to be hate speech.

Some of the previous approaches followed tweet level process and defined a threshold for the number of tweets for an author to be a 'Hate Speech Spreader'.

4.3.2 Training on Author level:

All the tweets of an author are concatenated together to form a sample. Thus, in each language, there are 200 samples with respect to 200 authors in training phase.

This approach seemed to be appropriate. Firstly, we have done experiments in tweet level. After working on the tweet level, we shifted towards author level.

The main factors that affect the performance of the model are Word Embeddings and Classifiers that are used in the experiment.

There are many popular methods to generate word embeddings. We experimented with the following techniques.

DistilBERT: Compared to BERT, DistilBERT was computationally compatible. We imported 'distilbert-base-uncased' model and its respective tokenizer from 'transformers' library.

Word2Vec: It is very popular method introduced in 2013 by Google. It is pretrained on a large corpus. GloVe, FastText are as well-known as Word2Vec. The way they are built is different. Each has its own pros and cons.

TF-IDF: This method generates embeddings considering occurrence of the word in the document with respect to occurrence in entire corpus. These do not comprise information of surrounding words.

Embedding layer: The models such as Word2Vec, GloVe, FastText face a problem of out-of-vocabulary, as they are pre-trained and are limited to a specific set of words. FastText is based on n-grams, so it may handle few new words, but those embeddings might not be relevant. To overcome this issue, we choose to build our own vocabulary and generate embeddings for each word using Embedding layer while iterating through each sample.

The classifiers that are used in this work are

Bi-LSTM: It is most used Natural Language Processing model after transformers and learns sequences efficiently, captures the information for a word in backward and forward directions.

ANN: Artificial Neural Networks have also given good results.

CNN: These are typically used for image classification tasks, because of their ability to capture the local context clearly and generalize it using pooling layers. But 1D CNN Layers can be used for sequences such as text.

V. PROPOSED MODELS

The different models that are developed in this work are

i. DistilBERT + Bi-LSTM: The embeddings generated by DistilBERT are provided to Bi-LSTM layer with 64 LSTM units followed by a GlobalMaxPooling1D layer and other dense layers.

ii. Word2Vec with CNN: Word2Vec embeddings are given to CNN model. If a word is not in vocabulary, it generates a new random embedding for it. It is on author level. Embedding size is 100.

The models iii, iv, v, vi are on tweet level. We have not tested all these following models on test data. Therefore, we have only the training/validation data accuracy of these, but not test accuracy. Few of them are tested and found their test accuracy was very low.

iii. Word2Vec with CNN [Tweet level]: This is similar to model ii, but ignores new words and trained on tweet level.

iv. Word2Vec, TF-IDF + CNN, LR, SVM: The Word2Vec embeddings are multiplied with TF-IDF weights. These are trained on tweet level. Training accuracy was good, but the test accuracy was bad.

What we thought was when the embeddings generated by Word2Vec are multiplied by TF-IDF weights, their meaning computed in the embeddings is getting altered.

v. DistilBERT, TF-IDF concatenated embeddings + Bi-LSTM: The embeddings of TF-IDF are concatenated horizontally to embeddings of DistilBERT and given to Bi-LSTM. We thought of multiplying embeddings with TF-IDF would disturb the meaning, therefore we concatenated. Our little intuition was LSTM has the ability to learn the sequences, so it would be able to derive the relationship between both the DistilBERT embeddings and TF-IDF embeddings.

vi. DistilBERT, TF-IDF concatenated embeddings + ANN: It is same as the above model v, but without LSTM, with normal dense layers.

We have not tried individual TF-IDF embeddings, because they are not robust and contextual. Most of the models are tried on the English dataset. If it was satisfying, we moved to check on Spanish dataset.

vii. Trained Embeddings + CNN: Two individual models were developed for two languages with the same architecture. Figure 1 depicts the architecture. It is on author level.

After the success of the model achieving better accuracy, we have tried to make it character-level model, having the same architecture. This was done considering the performance of models using n-grams and FastText.

viii. Trained char-level embeddings + CNN: It is similar to model vii, but here, each character is treated as a separate feature, and the model learns patterns in sequences of characters.

ix. Ensemble model using CNN: Considering the good scores of model vii, in the sense of betterment, we created an ensemble model that consists of 5 CNN models that are built as per model vii specifications. Voting criteria is used to decide the label. Fig. 1 shows the architecture of CNN model using trained embeddings.

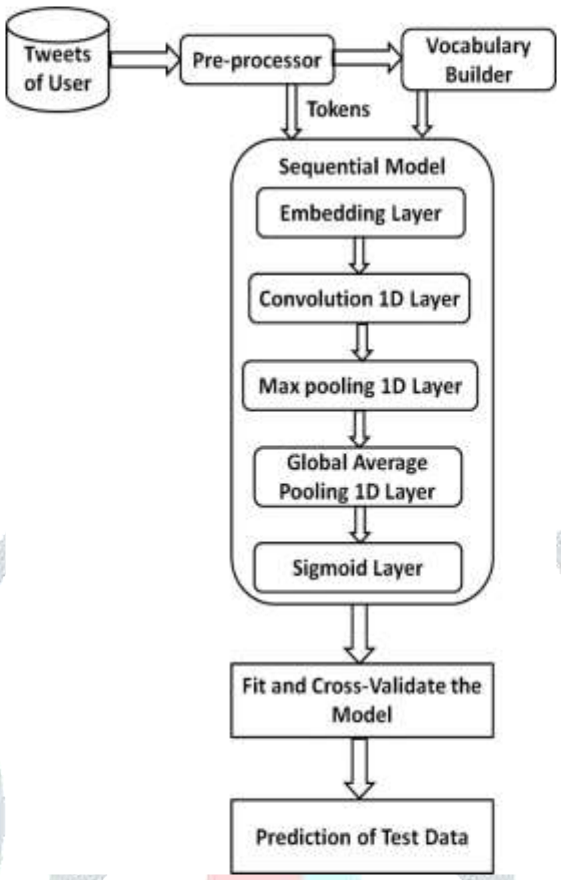

Figure 1: Architecture of CNN model using Trained Embeddings

In Fig. 1, the model had been run for 30 epochs and 5-fold cross-validation. We experimented with Average Pooling 1D layer and Max Pooling 1D layer after Convolution 1D layer, with various hyperparameters. The following layers with provided parameters worked best for us are specified in Table 2.

Table 2: The Hyperparameters used in different layers

Layer	Hyperparameters
Embedding Layer	Embedding size: 100
Convolution 1D Layer	No. of Kernels: 36, Kernel size: 24
Max Pooling 1D layer	Pool size: 3

VI. Experimental Results

The accuracies of the proposed models that are trained on author level for hate speech spreaders detection are presented in Table 3.

Table 3: The accuracies of hate speech spreaders detection when models trained on author level

S. No.	Model Name	Language	Training Accuracy	Testing Accuracy
1	DistilBERT + Bi-LSTM	English	0.69	0.66
2	Word2Vec + CNN	English	0.51	0.5
3	Trained Embeddings + CNN	English	0.65	0.72
4	Trained Embeddings + CNN	Spanish	0.85	**0.80**
5	Trained char-level embeddings + CNN	English	0.57	0.65
6	Trained char-level embeddings + CNN	Spanish	0.585	0.5
7	Ensemble model using CNN	English	**0.70**	**0.73**
8	Ensemble model using CNN	Spanish	**0.90**	0.77

For English language dataset, the Ensemble model using CNN obtained best training accuracy of 0.70 for hate speech spreaders detection and best testing accuracy of 0.73 for hate speech spreaders detection when compared with other models. For Spanish language dataset, the Ensemble model using CNN obtained best training accuracy of 0.90 for hate speech spreaders detection and the Trained Embeddings + CNN obtained best testing accuracy of 0.80 for hate speech spreaders detection when compared with other models.

The accuracies of the proposed models that are trained on tweet level for hate speech spreaders detection are presented in Table 4.

Table 4: The accuracies of hate speech spreaders detection when models trained on tweet level

S. No.	Model Name	Language	Training/Validation data Accuracy
1	Word2Vec + CNN	English	**0.75**
2	Word2Vec + LR	English	0.67
3	Word2Vec, TF-IDF + CNN	English	0.61
4	Word2Vec, TF-IDF + LR	English	0.66
5	Word2Vec, TF-IDF + SVM	English	0.66
6	DistilBERT, TF-IDF concatenated embeddings + Bi-LSTM	English	0.50
7	DistilBERT, TF-IDF concatenated embeddings + ANN	English	0.64

The combination of Word2Vec and CNN attained best training or validation accuracy of 0.75 for hate speech spreaders detection on English dataset when the training performed on tweet level.

We observed that the models that are trained on author level performed well and quite suitable for the task of hate speech spreaders detection.

We analysed that CNN has performed well in this task, it prioritizes the capturing the individual entities than the order they occur. It may be the way the words present in the given dataset, i.e., this task needs attention of capturing the words that make hate speech than their order, because the tweets are posted in an informal way and therefore, may not possess any characteristic of ordering or sequences.

VII. Conclusions

Hate Speech Spreaders Detection has become a necessity following the unwanted incidents in the society. The need of automated version paved way to the modern deep learning techniques. The traditional machine learning models may provide accuracy but may not learn perfectly the correlation in sequential data. We have tried BERT models and also merged the idea of TF-IDF, n-grams. We generated embeddings through pretrained models like Word2Vec, but the major issue was the out-of-vocabulary which certainly should be considered, because new words are common in these kind of problem statements like hate speech detection. Among classifiers SVM, LSTM, Logistic Regression, CNN outperformed with Max Pooling and Global Average Pooling Layers. The embeddings were trained on our training dataset and therefore are task-specific, increasing the scope of accuracy. The same approach was followed in both languages. For English language dataset, the Ensemble model using CNN obtained best training accuracy of 0.70 for hate speech spreaders detection and best testing accuracy of 0.73 for hate speech spreaders detection when compared with other models. For Spanish language dataset, the Ensemble model using CNN obtained best training accuracy of 0.90 for hate speech spreaders detection and the Trained Embeddings + CNN obtained best testing accuracy of 0.80 for hate speech spreaders detection when compared with other models.

References

[1] Hamed Babaei Giglou, Taher Rahgooy, Jafar Razmara, Mostafa Rahgouy and Zahra Rahgooy, Profiling Haters on Twitter using Statistical and Contextualized Embeddings Notebook for PAN at CLEF 2021.

[2] David Duki´c, Ana Sovi´c Krži´c, Detection of Hate Speech Spreaders with BERT Notebook for PAN at CLEF 2021.

[3] Kumar Gourav Das, Buddhadeb Garai, Srijan Das and Braja Gopal Patra, Profiling Hate Speech Spreaders on Twitter Notebook for PAN at CLEF 2021.

[4] Rakshita Jain, Devanshi Goel, Prashant Sahu1, Abhinav Kumar and Jyoti Prakash Singh, Profiling Hate Speech Spreaders on Twitter.

[5] Juan Pizarro, Using N-grams to detect Fake News Spreaders on Twitter Notebook for PAN at CLEF 2020.

[6] Jakab Buda, Flora Bolonyai, An Ensemble Model Using N-grams and Statistical Features to Identify Fake News Spreaders on Twitter Notebook for PAN at CLEF 2020.

[7] https://pan.webis.de/clef21/pan21-web/author-profiling.html