Classification Of Hateful Speech Against Women And Immigrants As Targeted And Aggressive

BITS Pilani Hyderabad Campus

CS F429
Natural Language Processing

Done By:
Group 32

Abhinava Madda
f20180844@hyderabad.bits-pilani.ac.in
Sangitha Chakraborthy
f20180876@hyderabad.bits-pilani.ac.in
Shagun Mishra
f20180767@hyderabad.bits-pilani.ac.in
Vyshnavi S K
f20180619@hyderabad.bits-pilani.ac.in

ABSTRACT

Hate speech is defined as any statement that disparages a person or a group based on a trait such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other qualities. Given the vast amount of user-generated content on the Internet, particularly on social media, the problem of recognizing, and thus potentially countering, Hate Speech spread is becoming increasingly important, for example in the fight against misogyny and xenophobia. Twitter's job is to facilitate public debate, which necessitates the representation of a diverse range of viewpoints. Yet, it does not advance viciousness against or straightforwardly assault or undermine others based on race, nationality, public cause, rank, sexual direction, age, inability, or genuine illness. Hate speech can be harmful to an individual or a community. Now, due to the increase in social media usage, hate speech is very commonly used on these platforms. So, it is not possible to identify hate speeches manually. So, we develop an automated hate speech detection model targeted towards women and immigrants in the wake of situations throughout the world. We use various text pre-processing methods, text embeddings, and feature extraction using TF-IDF, CBOW, and GloVE and perform 3 binary classification tasks using Random Forest and Support Vector Machine algorithms. We detect the presence of hate speech, and if it’s found to be present we further classify them into targeted vs non-targeted and aggressive vs non-aggressive.

INTRODUCTION

In today's world, we are constantly witnessing a troubling rise in xenophobia, racism, and intolerance around the world, including increased anti-Semitism, anti-Muslim hostility, and persecution of Christians. Bigotry is being used to spread through social media and other means of communication. White supremacist and neo-Nazi movements are on the rise. With incendiary speech that stigmatizes and dehumanizes minorities, migrants, refugees, women, and any other so-called "other," public discourse is being weaponized for political benefit. This isn't a one-off occurrence or the outbursts of a small group of people on the periphery of civilization. Hate is becoming more mainstream - in both liberal democracies and authoritarian regimes. The pillars of our shared humanity are eroded with each breach of the norm. Hate speech puts democratic values, societal stability, and peace at risk.

As more and more communication takes place in digital form, the full range of public conversations is moving online — in groups and broadcasts, in text and video, even with emoji. These debates show the breadth of human experience: some are educational and informative, some are amusing and entertaining, and still, others are political or religious in nature. Some can also be hateful and ugly. The majority of competent communication platforms and systems are now actively attempting to limit hateful content online. And this is exactly what the objective of our project is too.

Before getting into the depth of how hate speech can be monitored, classified, and controlled, it is empirical to define hate speech. Paula Fortuna and Sergia Nunes define "Hate speech is language that attacks or diminishes, that incites violence or hate against groups, based on specific characteristics such as physical appearance, religion, descent, national or ethnic origin, sexual orientation, gender identity or other, and it can occur with different linguistic styles, even in subtle forms or when humor is used ". Hate speech has long been a problem, but the scale, customization, and velocity of today's hate speech make it a distinctively modern problem. Hate speech has one major flaw that makes it difficult to categorize: subjectivity. Hate speech has no legal meaning and is not punishable under the law, with the exception of the First Amendment. As a result, it's difficult to say what constitutes hate speech.

The rapid rise of social media platforms such as Twitter and community forums has revolutionized communication and content production, but it is also increasingly being used to spread hate speech and organize hate-related acts. The objective of our project is to address the hateful tweets aimed against women and immigrants specifically. The reasons are as follows.

Firstly, Xenophobia is on the steep rise these days and there is a lot of unrest and hatred towards certain communities caused by the refugee crisis and political changes that occurred in the last

few years. Several governments and policymakers are currently trying to address it, making especially interesting the development of tools for the identification and monitoring of such kind of hate. Then we address misogynist comments towards women, because we still live in a patriarchal world and casual sexist comments in the workplace and other places are not okay. Specifically, with the rise in Black Lives Movement(BLM), Asian Hate, immigrants, Me too movement, there is a need to curb the same and report these offensive tweets, comments as smoothly and early as possible and thus automation of hate speech classification with the help of Natural Language Processing and Machine Learning techniques comes into play here.

RELATED WORKS

The following are the previous works that have been done in the area of hate speech recognition and classification.

Davidson et al.(2017)[1] performed a multi-class classification with logistic regression, naive Bayes, decision trees, random forests, and linear SVMs amidst a crowd-sourced hate speech lexiconto collected tweets containing hate speech keywords into three categories: those containing hate speech, only offensive language, and those with neither and provide a comparative comparison. They further used Porter stemmer to create bigram, unigram, and trigram features, each weighted by its TF-IDF and used NLTK To construct POS tags and used modified Flesch-Kincaid Grade Level and Flesch Reading Ease scores, to measure the quality of the tweets. Park and Fung(2017)[2] worked on Wassem datasets and proposed a classification method using a CNN over Word2Vec word embeddings, showing also classification results on racism and sexism hate sub-classes.

Kamble and Joshi(2018)[3] compared CNN - 1D, LSTM, and Bidirectional LSTM, using domain-specific embeddings creating a 300 Dimension Vector for each word, of which CNN-1D gives the best results at an F1-score of 0.8085. Similar supervised classification and lexical baselining approaches were used by Malmasi et al(2017)[4], which uses character n-grams, word n-grams, and word skip grams. They were able to attain an accuracy of 78% by using three labels, separating hate speech from offensive language.

Bencheng et al.(2021)[5] propose in their paper, an approach to detect offensive language in a public tweet dataset using a Bi-LSTM model with empty embeddings and pre-trained GloVe embeddings, and compared to pre-trained BERT, DistilBERT, and GPT-2 language models. Hashtags and emojis in the raw data are handled efficiently during pre-processing. An accuracy of 92% is observed for fine-tuned Bi-LSTM model with the optimal hyperparameters, when evaluated on the test data, and performs better than the transfer learning models.

Kumar et al.(2021)[6] deal with a binary classification problem in their paper. The proposed model uses TF-IDF and BERT embeddings to define the word vectors, since they performed better than word2vec, GloVe, etc. For both English and Spanish, an SVM was used as a classifier with the TF-IDF embeddings and for English, a pre-trained model BERTweet with a CNN was evaluated. Preprocessing involved the removal of noisy words, tokenization, and stemming(Porter stemmer), and all emojis were converted into text. The SVM model gives an accuracy of 67% and 81% for the English and Spanish datasets respectively, and the CNN achieved 66% accuracy.

Rahul et al.(2021)[7] also perform a multi-class classification of ‘Hinglish’ tweets into hate-inducing, abusive, and non-offensive with the help of character-level embeddings to deal with the variations in transliteration and grammatical liberties of mixed-language code and recommend 3 out of 12 models they experimented upon as to be robust and efficient: Only GRU, Only GRU with Attention, Bidirectional LSTM + GRU: with Attention after LSTM Layer.

Bhavesh Pariyani et al.(2021)[8] came up with machine learning algorithms to classify if a text is hate speech or not; performed on the Twitter dataset. They have used supervised classification methods such as logistic regression, support vector machines, and random forest on the hate speech dataset. In order to extract features from the tweets, they have used Tf-IDF and bag of words methods. In conclusion to their research, they have found that without data preprocessing, a Random forest with a bag of words gives the best performance. After preprocessing, SVM with Tf-IDF gives the best performance. However, Tf-IDF is more preferred than a bag of words because in the latter one, only frequency of words are considered and that information is used for vector generation.

Ashwin Geet d’Sa et al.(2021)[9] performed hate speech classification on a Twitter dataset. The tweets have been classified into three categories- hate, offensive, and neither classes. The paper proposes a solution based on deep neural networks and word embedding representation. Two embedding representations have been considered here- fastText and BERT. For the classification part, the following approaches have been discussed -a)word embeddings as input to SVM and DNN based classifiers, b) fine-tuning the BERT model. For the DNN based classifier, CNN, Bi-LSTM, and CRNN have been compared. According to the results, BERT fine-tuning result was found to be better.

The above literature review shows that pervasive works on the same topic have been done throughout the world on different datasets, using different techniques, there are barely any few works that specifically focus on one aspect of hate speech detection. Our work proposes and narrows down hate speech addressed towards tackling misogynism and xenophobia. Furthermore, these previous works just perform a broad open-ended categorization of hate and offensive speech. Our work also works towards addressing whether the hate speech is targeted towards a particular individual and if it’s aggressive or violent. Thus, the novelty of our work improves upon the existing work in multi-folds.

METHODOLOGY

The methodology involves a threefold process:
Part I: Text Preprocessing
Part II: Text embedding using TF-IDF, Bag of Words, and GloVE
Part III: Binary Classification using Random Forest, and Support Vector Machine

Part I: Text Pre-Processing

One of the most important steps in any natural language processing task is the text pre-processing step. The machine learning model does not understand what humans understand naturally. The data thus needs to be reduced and brought down to a simpler level. We need to clean the data to make it ready to feed into the model. Since we are using the Twitter dataset, the raw data which has been scraped off Twitter needs to be cleaned as there are a lot of unnecessary or unrequired words which may affect model performance. The following are the pre-processing steps that have been used in our model.

1. **Lowercasing:** The goal is to transform the input text to the same casing format as the output text, so that 'text,' Text,' and 'TEXT' are all treated the same. This is very useful for text featurization techniques like frequency and tfidf, as it allows you to mix similar words, reducing duplication and obtaining accurate counts and tfidf values. Most current vectorizers and tokenizers, such as sklearn TfidfVectorizer and Keras Tokenizer, do the lower casing by default. As a result, depending on our use case, we'll need to set them to false as needed.
2. **Removal of Punctuations:** The removal of punctuations from text data is another common text preparation approach. This is yet another text standardization method that will allow 'hurray' and 'hurray!' to be treated in the same way. Depending on the use case, we must additionally carefully select the list of punctuations to omit. For example, the string. punctuation in python contains the following punctuation symbols **!"#$%&\'()*+,-./:;<=>?@[\\]^_{|}~`**
3. **Removal of StopWords:** Stopwords are frequently used words in a language, such as 'the,' 'a,' and so on. Most of the time, they can be eliminated from the text because they don't provide useful information for downstream analysis. We can safely utilize these stopword lists because they have previously been produced for several languages. Take, for example, the nltk package's stopword list for the English language.
4. **Removal of usernames:** Tweets usually are targeted or contain tagged persons using their usernames such as “@ABCD”. While the removal of punctuation removes the tags, the usernames in tweets are otherwise useless in our dataset. Therefore we need to remove them too. We use regex pattern matching to remove the usernames.

5. **Removal Of URLS:** Since we are doing a Twitter analysis, then there is a good chance that the tweet will have some URL in it. Probably we might need to remove them for our further analysis.
6. **Removal of HTML Tags:** One other common preprocessing technique that will come in handy in multiple places is the removal of HTML tags. This is especially useful if we scrap the data from different websites. We use regex to scrape these off.
7. **Lemmatization:** Lemmatization is similar to stemming in that it reduces inflected words to their word stems, but it differs in that it ensures that the root word (also known as lemma) is a language word. We use the WordNetLemmatizer in NLTK to lemmatize our sentences.
8. **Removal of Digits and Emoticons.**

Part II: Text Embedding

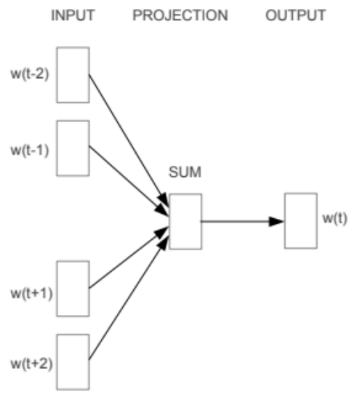
Word embedding, in short means the numerical representation of words. We wish to quantify the semantics. We want to represent words in such a manner that it captures their meaning in a way humans do. Not the exact meaning of the word but a contextual one. This is why word embeddings are required and below described are word embeddings/vectorization techniques we have used in our model.

1. **TF-IDF**: TF-IDF stands for “Term Frequency — Inverse Document Frequency”. This is a technique to quantify words in a set of documents. We usually assign each word a score to indicate its prominence in the document and corpus. It works by growing in proportion to the number of times a word appears in a document but is offset by the number of papers containing the word. As a result, words like 'this', 'what', and 'if', which appear frequently in all documents, rank low since they don't mean much to that document in particular. For each word in a document, the TF-IDF is calculated by multiplying two metrics:
    a. The term frequency of a word in a document. The simplest method for calculating this frequency is to simply count the number of times a word appears in a document. The frequency can then be adjusted based on the length of the document or the raw frequency of the most frequently used word in the document.
    b. The inverse document frequency of the word across a set of documents. This refers to how common or uncommon a word is within the entire document set. The closer a term is to zero, the more common it is. The logarithm may be determined by taking the total number of documents, dividing it by the number of documents that contain a word, and then multiplying by the total number of documents.

$$\text{tf-idf}(t, d) = \text{tf}(t, d) * \log(N/(\text{df} + 1))$$

So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1. We use TF-IDF in our model as a process of vectorization or text embedding so that once you've transformed words into numbers, in a way that's machine learning algorithms can understand, the TF-IDF score can be fed to algorithms such as Random Forest and Support Vector Machines.TF-IDF enables us to gives us a way to associate each word in a document with a number that represents how relevant each word is in that document. Then, documents with similar, relevant words(hate speech in our case) will have similar vectors, which is what we are looking for in our machine learning algorithm.

2. **Continuous Bag of Words(CBOW):** Based on the source context words(surrounding words), the CBOW model architecture attempts to forecast the current target word (the center word). While the Word2Vec family of models is unsupervised, this simply means that you may feed it a corpus and it will create dense word embeddings from it without any further labels or information. However, once you have this corpus, you will still need to use a supervised classification algorithm to find these embeddings. However, we will accomplish so without any further information, from within the corpus itself. This CBOW architecture may now be modeled as a deep learning classification model, in which we take the context words as input, X, and try to predict the target word, Y. In fact, this architecture is simpler to construct than the skip-gram model, in which we attempt to predict a large number of context words given a source-target pair..CBOW is several times faster to train than the skip-gram, with slightly better accuracy for the frequent words, and works better on smaller datasets. Therefore we use the same to predict how frequent are specific hate speech words in our corpus than using the skip-gram model for embedding.



CBOW

3. **GloVe Embedding:** GloVe stands for “Global Vectors. GloVe is an unsupervised learning technique that generates word vector representations. The resulting representations highlight intriguing linear substructures of the word vector space, and training is based on aggregated global word-word co-occurrence statistics from a corpus. GloVe has the benefit that, unlike Word2vec, it does not rely solely on local statistics (local context information of words) to generate word vectors, but also incorporates worldwide statistics (word co-occurrence). The GloVe approach is based on the crucial insight that the co-occurrence matrix can be used to infer semantic links between words. GloVe is a word vector technique that uses a corpus's global and local statistics to create a principled loss function that incorporates both. For our model for hate speech detection, we use the pre-existing embedding **‘glove.6B.100d’** obtained from https://nlp.stanford.edu/projects/glove/

Part III: Classification Models

We perform 3 binary classifications in our model:

1. Hate speech vs Non-Hate Speech

If it is hate speech, whether we can classify it into-

1.1 Targeted vs Non-Targeted

1.2 Aggressive vs Non-Aggressive.

More information on these classes will be described in the dataset section next up. Given below are the classification techniques our model uses.

1. **Random Forest(RF)**: The term "forest" refers to a grouping of trees. A Random Forest, on similar lines, is a collection of decision trees that are named random because they are a collection of largely uncorrelated trees that operate as a single model. The primary premise of random forest is that each tree expresses its prediction, and the random forest predicts its conclusion based on the majority decision. So, in our application for detecting hate speech, the trees label the statement as hate speech or not, and the random forest predicts whether the message is hate speech or not based on the majority choice. The logic goes for the other 2 sub-classifications as well. Vectorized text embeddings from TF-IDF, CBOW, and GloVE would be passed through the RF algorithm to perform the classifications.

2. **Support Vector Machines(SVM):** The "Support Vector Machine" (SVM) is a supervised machine learning technique that can solve classification and regression problems. It is, however, mostly employed to solve classification problems. Each data item is plotted as a point in n-dimensional space (where n is the number of features you have), with the value of each feature being the value of a certain coordinate in the SVM algorithm. Then we

accomplish classification by locating the hyper-plane that clearly distinguishes the two classes. SVM translates lower-dimensional input into higher-dimensional data using nonlinear or linear mapping. It looks for the linear optimal dividing hyperplane within this new dimension to separate the tuples between the sets. Information from two array scans is always discriminated by a hyperplane for sufficient nonlinear scaling to a sufficiently high dimension. With the use of support vectors, the SVM locates this hyperplane. Support vectors are those instances that are nearer to the margins. An unlimited number of separating lines could be drawn here. The target is to classify the "highest" one with the least classification error on previous unseen tuples. We do the same for our model for the three classification tasks for the TF-IDF, CBOW, and GloVE embeddings.

EXPERIMENTS

DATASET

Source: http://hatespeech.di.unito.it/hateval.html
The dataset is taken from the University of Turin (UniTO) Data Repository, on automatic hate speech detection in social media.(CSV File format)

Train Data: 9200 tweets

Test Data: 3000 tweets

Categories:

1. HS - a binary value indicating if HS is occurring against one of the given targets: 1 if occurs, 0 if not.

2. Target Range - if HS occurs (i.e. the value for the feature HS is 1), a binary value indicating if the target is a generic group of people (0) or a specific individual (1).

3. Aggressiveness - if HS occurs (i.e. the value for the feature HS is 1), a binary value indicating if the tweeter is aggressive (1) or not (0).

EVALUATION METHODS/METRICS

The following evaluation metrics are being used to figure how good and efficient our classification models are performing:

1. **Accuracy:** Accuracy is the proportion of true results among the total number of cases examined.

$$\textbf{Accuracy} = (TP+TN)/(TP+FP+FN+TN)$$

2. **Precision:** Precision gives the fraction of correctly identified as positive out of all predicted as positives.

$$\textbf{Precision} = (TP)/(TP+FP)$$

3. **Recall(or Sensitivity):** Recall gives the fraction you correctly identified as positive out of all positives

Recall = (TP)/(TP+FN)

4. **F1-Score:** It is defined as the harmonic mean of the model's precision and recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

5. **Specificity**: Specificity is the metric that evaluates a model's ability to predict the true negatives of each available category.

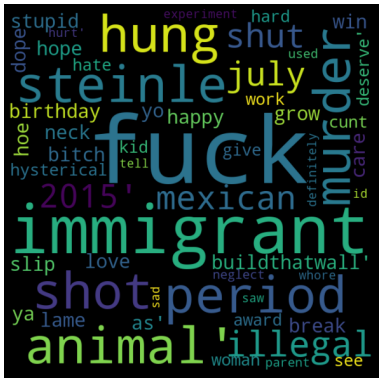
$$\textbf{Specificity} = \frac{\textbf{True Negatives}}{\textbf{True Negatives} + \textbf{False Positives}}$$

EXPERIMENTAL SETUP

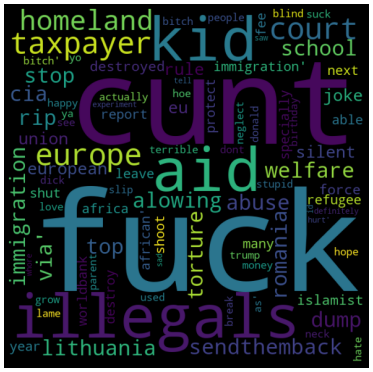
The training and testing data are first combined and performed an 80-20 split. Then text-preprocessing on the same is done using all the methods described in the previous section. Post pre-processing, word tokenization is done. Tokenization is a common task in Natural Language Processing (NLP). It's a fundamental step in both traditional NLP methods like Count Vectorizer and Advanced Deep Learning-based architectures like Transformers. We perform it using the pre-existing NLTK tokenizer.

Furthermore, prior to using the tokenized words into generating the word embeddings, we first make a word cloud for the 3 different classification classes as described above.

Word Cloud for Hate Speech (HS)



Word Cloud for Hate Speech that are Targeted(TG)



Word Cloud for Hate Speech that are aggressive(AG)

Next, we perform the text embeddings.
For TF-IDF, we take the pre-processed Twitter dataset. We use scikit learn for performing various algorithms related to TF-IDF. We will first use the TF-IDF vectorizer module to perform the vectorization. Then we do fitting and transformation on the pre-processed input data to finally get a matrix that contains the tf-idf values of each word in the document. We use this matrix as input to classifiers like SVM and RF.

For C-BOW, we referred to the paper published by Xin Rong[11] for the word2vec parameter updates. Using a window size of 4 considering two words to the left and one to the right of the center word and a learning rate of 0.0001, we find the final weight vectors after the backpropagation, to find the word embeddings of size 4 of the words which we can train the random forest and support vector machine architectures on.

We employed the glove.6B.100d.txt file for GloVe embeddings, as previously described. For each word, we'll get 100-dimensional vectors. However, GloVe vectors cannot be directly fed into classification models. As a result, we use the gensim module in Python to transform these vectors to the word2vec format. Each word in our corpus has its own vector. However, because the training data includes sentences, we must create sentence embeddings. We take the mean of the ith values in the word embedding of all the words in the sentence to build a vector to represent the ith sentence of the data.

Now we have embeddings for each sentence and the corresponding labels. We divided the data into training and testing sets in an 80-20 ratio. For SVM, the rbf kernel function was employed, and the hyperparameters C and gamma were tweaked using GridSearchCV.Gamma took values from [0.0001,0.001, 0.01, 0.1, 1, 10] and C took values from [0.01, 0.1, 1, 10, 100]. SVM models with different combinations of C and gamma were implemented on training data and the combination giving the maximum accuracy was run on test data.

In RF, more hyperparameters are varied. Thus RandomisedSearchCV was used instead of GridSearchCV. The number of estimators was varied from 10 to 100, the max split parameter took either log2 or sqrt value, maximum levels of the tree were varied from 10 to 110, the

minimum number of splits at each node were 2,5 or 10. The minimum number of leaves at each node took value from [1,2,4]. The combination with the best performance on training data was chosen and then implemented on test data.

RESULTS AND DISCUSSION

TF-IDF

	Accuracy	Precision	Recall	F1-Score	Specificity
RF(HS vs Non-HS)	**0.7329**	0.8182	0.7573	0.7804	0.7102
SVM(HS vs Non-HS)	0.7262	0.7902	0.7508	0.7700	0.6877
RF(AG vs Non-AG)	0.6719	0.7664	0.7145	0.7324	0.6198
SVM(AG vs Non-AG)	**0.6788**	0.7626	0.6992	0.7295	0.6458
RF(TR vs Non-TR)	0.8364	0.8515	0.8836	0.8672	0.7644
SVM(TR vs Non-TR)	**0.8533**	0.8783	0.8867	0.8825	0.7984

CBOW

	Accuracy	Precision	Recall	F1-Score	Specificity
RF(HS vs Non-HS)	**72.95%**	0.764	0.744	0.762	0.687
SVM(HS vs Non-HS)	70.70%	0.748	0.739	0.743	0.662
RF(AG vs Non-AG)	**68.38%**	0.721	0.696	0.869	0.761
SVM(AG vs Non-AG)	65.60%	0.749	0.690	0.718	0.596

RF(TR vs Non-TR)	**84.14%**	0.843	0.868	0.869	0.761
SVM(TR vs Non-TR)	80.27%	0.813	0.863	0.837	0.715

GloVE

	Accuracy	Precision	Recall	F1-Score	Specificity
RF(HS vs Non-HS)	66.6667%	0.645251	0.458333	0.535963	0.817529
SVM(HS vs Non-HS)	**70.2083 %**	0.663765	0.600394	0.630491	0.776734
RF(AG vs Non-AG)	64.1229%	0.592025	0.457346	0.516043	0.773424
SVM(AG vs Non-AG)	**81.8750%**	0.846154	0.024775	0.04814	0.998978
RF(TR vs Non-TR)	82.1606%	0.759312	0.734072	0.746479	0.87037
SVM(TR vs Non-TR)	**84.7917 %**	0.569767	0.251928	0.349376	0.963202

It can be observed from the above tabular results that, for Hate Speech(HS vs Non-HS) we get the best classification accuracy when we use TF-IDF as the word embeddings and Random Forest as the classifier. For Aggressiveness(AG vs Non-AG), we see that GloVE embedding performs the best giving a classification accuracy with the SVM classifier. For Targeted vs Non-Targeted(TR vs Non-TR) we see that GloVE embeddings again give the best classification accuracy as high as around 85%. It is also to be observed that this is the highest accuracy obtained amidst all classifications. Furthermore, it is also to be observed that GloVE performs the best with the SVM classifier and CBOW performs best with RF. Overall, we can conclude that TF-IDF embedding works the best for HS vs Non-HS classification, and GLoVE works the best for the sub-classifications of the same.

CONCLUSION

Hate speech in social media is now pervasive than ever been before. It is important to tackle the growing bias and the unchecked freedom to hide behind anonymity spewing hurtful words against innocents. We proposed using various Natural Language Processing and Machine Learning techniques to specifically tackle the issues of xenophobia and misogynism. After text-preprocessing on Twitter data, we used text-embeddings methods like TF-IDF, CBOW, and GLoVE to vectorize the word data into numerical form and make it easier for the classifier modes to train, classify and predict. We performed one main classification into Hate Speech(HS) vs Non-Hate Speech and then two sub-classifications of HS into Targeted(TR) vs. Non-Targeted and also Aggressive(AG) vs Non-Aggressive using Random Forest and Support Vector Machine algorithms. After evaluating the results based on several metrics, we can conclude that TF-IDF word embedding and vectorization work best for HS vs Non-HS classification, and the sub-classifications measuring aggressiveness and targeted-ness work best with GloVE embedding and SVM classifier.

REFERENCES

[1] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated Hate Speech Detection and the Problem of Offensive Language. ICWSM, 2017

[2] S. Kamble and A. Joshi, “Hate speech detection from code-mixed Hindienglish tweets using deep learning models,” 2018.

[3] S. Malmasi and M. Zampieri, “Detecting hate speech in social media,” in International Conference Recent Advances in Natural Language Processing, RANLP, 2017.

[4] J. H. Park and P. Fung. One-step and Two-step Classification for Abusive Language Detection on Twitter. ALW1 1st Work.Abus. Lang. Online, 2017.

[5] Bhavesh Pariyani, Krish Shah, Meet Shah, Tarjini Vyas, Shehshang Degadwala.Hate Speech Detection in Twitter using Natural Language Processing.Proceedings of the Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021). IEEE Xplore Part Number: CFP21ONG-ART; 978-0-7381-1183-4

[6] Ashwin Geet d’Sa, Irina Illina, Dominique Fohr. Classification of Hate Speech Using Deep Neural Networks. Revue d’Information Scientifique & Technique , Centre de Recherche sur l’Information Scientifique et Technique (CERIST), 2020, From Data and Information Processing to Knowledge Organization : Architectures, Models and Systems, 25 (01). Ffhal-03101938f

[7] Wei, B., Li, J., Gupta, A., Umair, H., Vovor, A., & Durzynski, N. (2021). Offensive Language and Hate Speech Detection with Deep Learning and Transfer Learning. arXiv preprint arXiv:2108.03305.

[8] Das, K. G., Garai, B., Das, S., & Patra, B. G. (2021). Profiling Hate Speech Spreaders on Twitter. In CLEF.

[9] Rahul, V. Gupta, V. Sehra and Y. R. Vardhan, "Hindi-English Code Mixed Hate Speech Detection using Character Level Embeddings," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 1112-1118, DOI: 10.1109/ICCMC51019.2021.9418261.

[10] Basile, V., Bosco, C., Fersini, E., Debora, N., Patti, V., Pardo, F. M. R., ... & Sanguinetti, M. (2019). Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and

women in twitter. In 13th International Workshop on Semantic Evaluation (pp. 54-63). Association for Computational Linguistics.

[11] Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.