EasyChair Preprint

№ 4810

Roman Urdu Multi-Class Offensive Text Detection using Hybrid Features and SVM

Tauqeer Sajid, Mehdi Hassan, Mohsan Ali and Rabia Gillani

December 25, 2020

Roman Urdu Multi-Class Offensive Text Detection using Hybrid Features and SVM

Tauqeer Sajid
Department of Computer Science
National Cybercrime Forensics Lab
Air University, Sector E-9, Islamabad
Pakistan
raotauqeer36@gmail.com

Mehdi Hassan
Department of Computer Science
National Cybercrime Forensics Lab
Air University, Sector E-9, Islamabad
Pakistan
mehdi.hassan@mail.au.edu.pk

Mohsan Ali
Department of Computer Science
National Cybercrime Forensics Lab
Air University, Sector E-9, Islamabad
Pakistan
amohsan636@gmail.com

Rabia Gillani
Department of Computer Science,
National Cybercrime Forensics Lab
Air University, Sector E-9, Islamabad
Pakistan
160923@students.au.edu.pk

***Abstract*—Hate content has become a significant issue worldwide due to the increase in social networking sites. Detection of hate content from a language other than English is challenging. We propose a new technique that automatically detects the *Roman Urdu* comments from YouTube videos into five classes. These classes, including, R*eligious Hate, Violence Promotion, Extremist (Racist), Threat/Fear, and Neutral*. We have generated dataset by scrapping *Roman Urdu* comments from YouTube videos and labeled by the language experts. We have considered N-grams and TF-IDF values for feature extraction followed by SVM classification. Some classes have relatively less instances, and we employed SMOTE for class-balancing. The developed model offers a high classification performance of 77.45% using the 10-Fold cross-validation technique. The proposed approach offers superior classification results as compared to others.**

Keywords—hate speech, n-gram, tf-id, machine learning, deep learning, youtube

I. INTRODUCTION

In Pakistan, most people used *Roman Urdu* for comments on video-sharing and social media platforms like YouTube, Facebook, and Twitter. From the last few years in Pakistan, tremendous growth in the number of people using YouTube. YouTube is the second most visited website in Pakistan [1]. People from different religions, cultures, and educational backgrounds use YouTube. Sometimes people upload videos which might be inappropriate for various cultures or religions, which may lead to verbal assaults in comments because of differences in people's opinions. Such type of actions may create law and order situation in the country. People have freedom of speech so they can comment on any content on YouTube, which leads to the use of abusive language, racist comments, religious hate, and sometimes people even give menace. Hate speech makes a terrible impact on society and damages people's mental health, so people commit suicide [2].

Hate speech is a huge issue, although, for English and some other languages, there is much work that has been done in the hate speech detection field, for *Roman Urdu* there is no work done to detect hate speech. This increases the importance of detection hate speech in *Roman Urdu* to remove such content, so we can save people from cyberbullying. Also, manually removing such content is challenging, which also increases the importance of an automated system, which can detect hate content from comments on YouTube.

We divided the hate speech into five different classes: Religious Hate, Violence Promotion—Extremist (Racist), Threat/Fear, and Neutral. Similar approaches were used to detect hate speech used in previous research [3] and [4]. There is no international legal definition of hate speech. However, according to UN hate speech is: any kind of communication in speech, writing or behavior that attacks or uses abusive or discriminatory language which refers to a group or single person based on religion, ethnicity, nationality, race, color, gender or other identity factors [5].

In this research paper, we proposed a solution that can identify hate speech into five classes. People use youtube as a medium to spread hate speech in the country. So, we decided to use youtube as a source. We made our scrapper that scrape the comments from YouTube videos. We annotate these comments into their respective class. We train our model using n-gram with norm L1 and L2 of term frequency-inverse document frequency (TF-IDF) as features values and classify the comments. We evaluate the model using metric scores and a confusion matrix. In this research, we perform comparative analysis using Logistic Regression (LR), Support Vector Machines (SVM), SGDClassifier, Naive Bayes (NB) using n-gram with L1 and L2 norm of TF-IDF values and document to vector features with Logistic Regression (LR), Support Vector Machines (SVM) and SGDClassifier as classifier models. Our results show that on our Roman Urdu dataset, Support Vector Machine (SVM) performs better than all other models on n-gram with norm L2 TF-IDF features values. We also did hyperparameter tuning of our machine learning models using 10-Fold cross-validation. We make a YT Monitor web interface that scrapes the comments from a given link or keyword and classifies comments into its respected classes of hate speech.

We organized this paper as follows. In section II, we will overview related work, which approaches people used to solve the hate speech problem. In section III, we will explain our methodology; which steps we follow to solve hate speech problem. Section IV will describe the results of the comparative analysis of machine learning models using different features. In section, V concludes our research.

II. RELATED WORK

Detection and checking of a Hate Speech in social media cannot be an easy task. Every day many people write text on social media; they use informal languages. Different people use different languages; that is why some words for some people, are a joke, but for other people, hate speech [3]. This point is difficult to distinguish.

Different machine learning and deep learning approaches have used to detect hate speech. In some researches, sentence structure used to capture hate speech [6], many others used Lexical features [7], and a bag of words [8] approaches to detect hate speech. Previous research observed that these features were not entirely useful to understand the hate speech from text and failed. On the other hand, the N-gram feature with TF-IDF also used in research, which showed better results [9] [10].

Lexical features have two main approaches, dictionary-based and corpus-based. In Lexical features, it involves the words of the same meaning tagged in a static dictionary with polarity labels and semantic orientation scores. In a bag of words (BOW), the text is tokenized into words, followed by its word frequencies. As a bag of words (BOW) did not care about word order, semantic of words, and grammar, it mostly used for basics works of natural language processing (NLP) [11].

Linguistic Features are comprised of sample length, parts-of-speech, average length of words, number of periods, punctuations, URLs, capitalized letters, polite words, insults words, hate speech words and one letter words. These features did not provide much importance in studies and did not show much improvement in the classifier accuracy [12]. Sentiment analysis features show their importance in hate speech detection, and it has seen that they are closely related. It assumed that most negative sentiment concerns hate speech [13]. Hate speech shows higher negative polarity where hate speech present in document [14].

Recently, word embedding has proposed to detect hate speech [15]. In Word embedding's tokens are devour sequentially in the matrix through the concatenation of tokens embedding's [16]. Also, Deep learning algorithms recently used to detect hate speech, such as Convolutional Neural Network (CNN and LSTM). Convolutional Neural Network (CNN) is used to detect hate speech from Twitter in recent researches [17] and [18]. They also perform further analysis using a word embedding, so they understand the effect of the feature selection process on different models.

III. METHODOLOGY

For this research, we obtained a new Roman Urdu dataset for hate speech detection from YouTube comments. For hate speech detection flowchart can be found in Fig. 1.

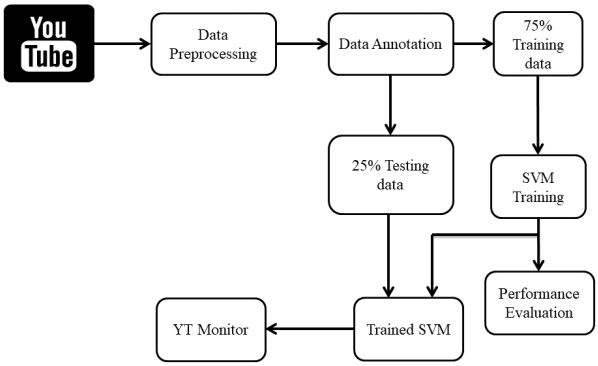

Fig. 1. Block diagram of proposed methodology

A. Data Crawling

The first step is to crawl YouTube comments data; for this purpose, we developed a YouTube comments crawler. This crawling process had done using the scrapy library with AJAX request. If we give a crawler link of any video, it will scrape all comments from this video, or if we provide a YouTube channel link, the scraper will scrape all comments from all videos uploaded on the provided YouTube channel. We also can search for videos by keyword we input the keyword in the search box, and the crawler will scrape the top videos against the given keyword, and top videos will show on the YT Monitor interface. From shown videos against keyword searches, we can select which video comments we want to scrape and simply click the scrape button, and comments will scrape for that selected video. For this research, we query different keyword and find videos that have comments contains offensive in *Roman Urdu*. We also query many offensive *Roman Urdu* words on YouTube to finds a video that has comments related to offensive keywords. We scrape about 16806 comments data from YouTube.

B. Data Pre-processing

In pre-processing, we filter the comments dataset first, and we remove duplicate comments from data. To decide whether a given document is roman or not, we used a pre-defined set of roman words named as roman dictionary; dictionary contains the set of roman words and words possible write up. For example, some people use to write the word kafir or kafar. So, we stored all the possible roman words that a user can write in the document. The roman dictionary is compiled by the language experts in our team. Also, we remove those comments which contain languages other than *Roman Urdu*. After filtering the comments, we got 16300 comments, which is labeled by the language experts.

TABLE I. FEW SAMPLES FROM THE DATASET

Document	Label
This Randi doesnt know the difference between fuel tank and bomb.	Violence Promotion
Pakistan zindabad pak army zindabad pakistan isi zindabad.	Neutral
Tu pak k nitale kuto hramiyo aagar bhart tum pe mut bhi de na to tum to use amrit samaz kar pi jaoge.	Extremist
Sahaba se bughuz sirf harmi karskta hn.	Religious
meri khawish hai me gun uthaon aur un sb ko maar do jo wahan milen gaye muje.	Threat

For this research, the dataset annotated into five different labels: Religious Hate, Violence Promotion, Extremist (Racist), Threat/Fear, and Neutral. Three annotators annotate this dataset. We provide the essential guide for data annotation to avoid any biasness and if we have a clash in any comment label, we finalized its label by majority voting. Three annotators annotated every comment in the dataset. After the annotations process complete by three annotators, we have all the comments by annotators agreement. The number of class labels are depicted in Fig. 2.

Fig. 2, we can observe that classes are imbalanced. For violence Promotion class we have 5264 records, for Neutral class, we have 4405 records, for Religious Hate class we have 3914 records, for Extremist (Racist) class we have 2186, and for Threat/Fear, we just have 1037 records. So to solve this imbalanced class problem, we used Synthetic Minority Over-sampling Technique (SMOTE) [19].

For *Roman Urdu* data, we first need to make a list of stop words because we did not have pre-defined stop words for

Roman Urdu. We convert all the comments to lowercase and remove the following unnecessary elements from comments:

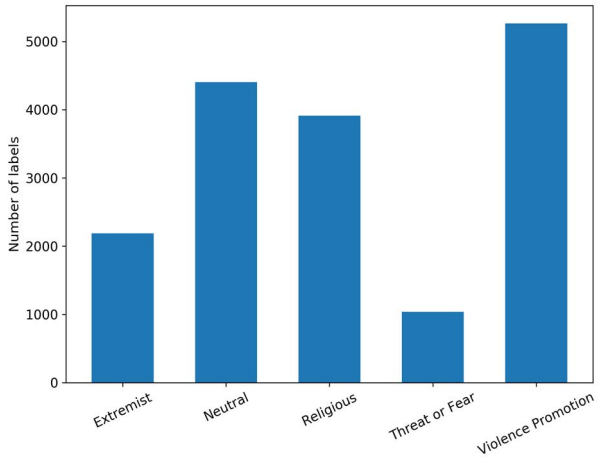

Fig. 2. Distribution of obtained dataset 1

1) Bad symbols
2) Stop words
3) Non-Ascii characters
4) Punctuations
5) Uniform resource locator (URLs)
6) Emoji's

Next step which is a challenge for us, In *Roman Urdu*, every person have different writing style like some people write kafir word as Kfar or Kafr or Kafar. So, we have to convert the word into its original form for that problem, but we did not have stemming or lemmatization pre-defined for *Roman Urdu* data. We did not have a dictionary for *Roman Urdu* data to convert the word into its original form. For this challenging task, we start working on making a dictionary for *Roman Urdu* data to convert all different forms of the same meaning word into one word. For this research, we build that dictionary and convert different forms of the same meaning word in our data into one word. Our dataset is not published online yet and hope so will be available soon, because we are performing experiments based on that data.

C. Feature Extraction

For this research, we use n-grams features from uni-gram to tri-gram and give the weights with TF-IDF values. We use TF-IDF to remove biasness from those tokens, which occurs very frequently in data but are very less informative. TF-IDF is computationally and mathematically easy to implement for problem in hand. The other important thing in the TF-IDF is that it is very simple to calculate the similarity between two or more documents. Basic calculations such as addition and subtraction process is used to extract the most descriptive terms from the dataset. Common terms or words in the dataset not affect the results due to IDF (e.g. "is", "are", "am", etc.).When we complete the feature extractions process, we provide the data to models for classification. A formula [4] that used to compute TF-IDF is:

$$\mathrm{TF-IDF}(t) = tf(t,d) \times idf(t) \quad (1)$$

We use the L1 and L2 norm of TF-IDF performing experiments. L1 norm of TF-IDF is defined as:

$$v_{norm} = \frac{v}{|v_1| + |v_2| + \ldots + |v_n|} \quad (2)$$

In L2 norm n is the total number of documents. L2 norm of TF-IDF is defined as:

$$v_{norm} = \frac{v}{\sqrt{{v_1}^2 + {v_2}^2 + \ldots + {v_n}^2}} \quad (2)$$

The L1 and L2 are normalization techniques, which are used to normalize the vectors such as TF-IDF vector. The L1 normalization is also called Manhattan .The input to the L1 normalization is the absolute values of the vector TF-IDF. The input to the L2 normalization vector is the square root of the TF-IDF vector.

The dimensions of the uni-gram, bi-gram and tri-grams used for the feature extraction of the textual data. The vectors dimensions are shown in the TABLE II given below:

TABLE II. FEATURES EXTRACTION DIMENSION OF TF-IDF

Features	Dimensions
Uni-gram	(25865, 13897)
Uni+bi-gram	(25865, 26002)
Uni+bi+tri-gram	(25865, 28884)

D. Classification Models and Evaluation

For this research, we use different classifiers Logistic Regression (LR), Support Vector Machine (SVM), SGDClassifier, Naive Bayes (NB), and document to vector features with Logistic Regression, Support Vector Machine (SVM) and SGDClassifier. We use Scikit-Learn and Keras for the implementation of these models. Scikit-Learn is a trendy library that provides highly efficient classification models which nowadays, almost every researcher uses in their research for text and other classification. For better results, we tune our model parameters using GridSearchCV, also to avoid our model from overfitting, we use 10-Folds cross-validation and evaluate our model. Mostly in the research field, 10-Fold cross-validation is used. In 10-Fold cross-validation, we divided the dataset into 10 parts where 9/10 parts of data use for training data, and the 1/10 data use for testing the model.

E. YT Monitor

YT Monitor is a web-based application developed to scrape comments and performs the task of multi-class offensive detection in *Roman Urdu* data. It has an input field where user give URL or keyword to scrape comments of videos from YouTube. Input will pass to the YouTube comment scraper, which we made, and the comment will scrape for respected input. After scraping the comments, we apply pre-processing steps on scraped comments and pass the pre-processed comments to our machine learning model, which classifies the comments to its respected classes. We show different graphs as a result, such as a pie chart, line graph, word cloud.

IV. RESULTS

The comparative analysis results of machine learning models Logistic Regression (LR), Support Vector Machine (SVM), SGDClassifier (SGD) and Naive Bayes (NB) using different combinations of feature parameters of TF-IDF shown in TABLE I, document to vector features with Logistic Regression (LR), Support Vector Machine (SVM), SGDClassifier (SGD) shown in TABLE II.

TABLE III. COMPARISON OF MODELS WITH DIFFERENT N-GRAM FEATURES AND TF-IDF VALUES

N-gram with TF-IDF Norm	Accuracy			
	LR	***SVM***	***SGD***	***NB***
word uni-gram with L1 norm	0.6369	0.7075	0.6565	0.6565
word uni-gram + bi-gram with L1 norm	0.6330	0.6956	0.6581	0.6690
word uni-gram + bi-gram + tri-gram with L1 norm	0.6338	0.6944	0.6607	0.6734
word uni-gram with L2 norm	0.6879	0.7298	0.7038	0.6836
word uni-gram + bi-gram with L2 norm	0.6970	**0.7326**	0.7179	0.7010
word uni-gram + bi-gram, tri-gram with L2 norm	0.6973	0.7318	0.7174	0.7017

TABLE III presents that machine learning algorithms perform better on the L2 norm of TF-IDF than on the L1 norm of TF-IDF. The best model is Support Vector Machine, and its accuracy is 73.26%, on uni-gram, bi-gram with TFIDF norm L2. Support Vector Machine (SVM) performs better on uni-gram, bi-gram, tri-gram with TFIDF norm L2, and achieved 73.18% accuracy. SGDClassifier performs better on uni-gram, bi-gram with TFIDF norm L2, and achieved 71.79% accuracy. Naïve Bayes (NB) performs better on uni-gram, bi-gram, tri-gram with TFIDF norm L2, and achieved 70.17% accuracy.

TABLE IV. COMPARISON OF DOCUMENT TO VECTOR FEATURES WITH DIFFERENT MACHINE LEARNING MODELS

Models	Accuracy
LR	**0.6142**
SVM	**0.6142**
SGD	0.6069

TABLE IV demonstrates that Logistic Regression (LR) and Support Vector Machine (SVM) performs the same and better than SGDClassifier (SGD) using the document to vector features. We got 61.42% accuracy for Logistic Regression (LR) and Support Vector Machine (SVM) using the document to vector features, and for SGDClassifier (SGD), we got 60.69% accuracy.

Machine learning models perform better using n-gram with the L2 norm of TF-IDF values on our dataset. We tune machine learning models using 10-Fold cross-validation on n-gram with the L2 norm of TF-IDF shows in TABLE III.

TABLE V. COMPARISON OF TUNED MODELS WITH DIFFERENT N-GRAM FEATURES AND TF-IDF VALUES

N-gram with TF-IDF Norm	Accuracy			
	LR	***SVM***	***SGD***	***NB***
uni-gram with L2 norm	0.7386	0.7743	0.7038	0.7077
uni-gram + bi-gram with L2 norm	0.7606	0.7734	0.7179	0.7337
uni-gram + bi-gram + tri-gram with L2 norm	0.7614	**0.7745**	0.7174	0.7312

TABLE V shows that after tuning the parameters of models, Support Vector Machine (SVM) performs best on our data. We tune Support Vector Machine (SVM) on Kernel and Regularization C with different values. We get the best parameters of Support Vector Machine (SVM) C is 100, and the kernel is rbf on uni-gram, bi-gram, tri-gram with L2 norm of TFIDF values. Support Vector Machine (SVM) precision, recall, and f-score can see in TABLE IV.

TABLE VI. FINAL TUNED SUPPORT VECTOR MACHINE (SVM) MODEL SCORES ON TEST DATA

	Precision	Recall	F-Score
Religious Hate	0.78	0.71	0.74
Violence Promotion	0.69	0.74	0.72
Extremist (Racist)	0.86	0.76	0.81
Threat/Fear	0.83	0.96	0.89
Neutral	0.72	0.69	0.70

TABLE VI shows that the violence promotion class precision is 0.69, which shows that model 31% predict the other four classes as violence promotion. Recall for neutral comments is 0.69, comparatively low than other class's recall, which shows 31% comments which are neutral but misclassified by the model. Recall for Threat/Fear is 0.96, which is better.

We also compute the confusion matrix for the tuned Support Vector Machine (SVM), which can see in Fig. 3. Confusion matrix, also known as an error matrix, is a specific table that visually describes the performance of a supervised classification machine learning algorithm.

Fig. 3 we can see that model did several misclassifications. To increase the score of model improvements can be done in this area. The final accuracy of the Support Vector Machine (SVM) model obtained on testing data is 77.45%.

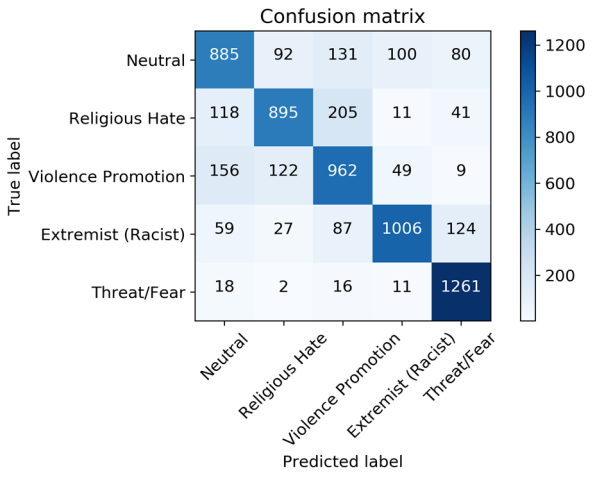

Fig. 3. Confusion matrix of final tuned SVM model

V. CONCLUSION

In this paper, we proposed a solution to detect the *Roman Urdu* hate speech on YouTube. We trained machine learning models based on features, n-gram with TF-IDF values of L1, L2 norm, and document to vector features. We performed a comparative analysis of Logistic Regression (LR), Support Vector Machine (SVM), SGDClassifier, and Naive Bayes (NB) on different feature values with 10-Fold hyperparameter tuning. The research results showed that Support Vector Machine (SVM) performs outperforming the other models based on unigram, bi-gram, and tri-gram with the L2 norm of TFIDF values. Performance evaluation reports of different models proved that SVM is more accurate than any other model on our dataset. SVM is 77.45% accurate, which is higher than any other mentioned machine learning models. In the future, the accuracy of the model is increasable by means of model training on more *Roman Urdu* data collected from YouTube.

ACKNOWLEDGMENT

This research is carried out under National Cybercrime Forensics Lab and supported by ORIC, Air University, Islamabad.

REFERENCES

[1] "Alexa - Top Sites in Pakistan - Alexa." https://www.alexa.com/topsites/countries/PK (accessed Jun. 26, 2020).

[2] "Bullying and suicide - Wikipedia." https://en.wikipedia.org/wiki/Bullying_and_suicide (accessed Jun. 27, 2020).

[3] M. O. Ibrohim and I. Budi, "A Dataset and Preliminaries Study for Abusive Language Detection in Indonesian Social Media," Procedia Comput. Sci., vol. 135, pp. 222–229, 2018, doi: 10.1016/j.procs.2018.08.169.

[4] A. Gaydhani, V. Doma, S. Kendre, and L. Bhagwat, "Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach," 2018, [Online]. Available: http://arxiv.org/abs/1809.08651.

[5] A. Guterres, "United Nations Strategy and Plan of Action on Hate Speech," no. May, pp. 1–5, 2019, [Online]. Available: https://www.un.org/en/genocideprevention/documents/advising-and-mobilizing/Action_plan_on_hate_speech_EN.pdf.

[6] L. Silva, M. Mondal, D. Correa, F. Benevenuto, and I. Weber, "Analyzing the targets of hate in online social media," Proc. 10th Int. Conf. Web Soc. Media, ICWSM 2016, no. June, pp. 687–690, 2016.

[7] N. D. Gitari, Z. Zuping, H. Damien, and J. Long, "A lexicon-based approach for hate speech detection," Int. J. Multimed. Ubiquitous Eng., vol. 10, no. 4, pp. 215–230, 2015, doi: 10.14257/ijmue.2015.10.4.21.

[8] C. K. Themeli, "Hate Speech Detection using different text representations in online user comments," no. October 2018, 2018, doi: 10.13140/RG.2.2.12991.25764.

[9] J. Salminen et al., "Anatomy of online hate: Developing a taxonomy and machine learning models for identifying and classifying hate in online news media," 12th Int. AAAI Conf. Web Soc. Media, ICWSM 2018, no. Icwsm, pp. 330–339, 2018.

[10] B. Mathew, N. Kumar, Ravina, P. Goyal, and A. Mukherjee, "Analyzing the hate and counter speech accounts on Twitter," 2018, [Online]. Available: http://arxiv.org/abs/1812.02712.

[11] C. Themeli, G. Giannakopoulos, and N. Pittaras, "A study of text representations for Hate Speech Detection," no. February 2020, 2019.

[12] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," 25th Int. World Wide Web Conf. WWW 2016, pp. 145–153, 2016, doi: 10.1145/2872427.2883062.

[13] A. Schmidt and M. Wiegand, "A Survey on Hate Speech Detection using Natural Language Processing," no. 2012, pp. 1–10, 2017, doi: 10.18653/v1/w17-1101.

[14] N. Bauwelinck and E. Lefever, "Measuring the Impact of Sentiment for Hate Speech Detection on Twitter," no. c, pp. 17–22, 2019.

[15] H. Faris, I. Aljarah, M. Habib, and P. A. Castillo, "Hate speech detection using word embedding and deep learning in the Arabic language context," ICPRAM 2020 - Proc. 9th Int. Conf. Pattern Recognit. Appl. Methods, no. March, pp. 453–460, 2020, doi: 10.5220/0008954004530460.

[16] S. Zimmerman, C. Fox, and U. Kruschwitz, "Improving hate speech detection with deep learning ensembles," Lr. 2018 - 11th Int. Conf. Lang. Resour. Eval., pp. 2546–2553, 2019.

[17] B. Gambäck and U. K. Sikdar, "Using Convolutional Neural Networks to Classify Hate-Speech," no. August, pp. 85–90, 2017, doi: 10.18653/v1/w17-3013.

[18] Z. Zhang, D. Robinson, and J. Tepper, "Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network," Eurpoean Semant. Web Conf., pp. 745–760, 2018, [Online]. Available: https://doi.org/10.475/123_4.

[19] "imblearn.over_sampling.SMOTE — imbalanced-learn 0.5.0 documentation." https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html (accessed Jun. 27, 2020).