

IDENTIFICATION AND CLASSIFICATION OF HATE SPEECH IN SOCIAL MEDIA

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

NAVEEN KRISHNA G (37110494)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

March - 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with "A" grade by NAAC | 12B Status by UGC | Approved by
AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Mr.
NAVEEN KRISHNA G (17SCS5133), who carried out the project entitled
**"IDENTIFICATION AND CLASSIFICATION OF HATE SPEECH IN
SOCIAL MEDIA"** under my supervision from November 2020 to March
2021.

Internal Guide

Dr. Usha Nandini, M.E., Ph.D.

Head of the Department

Dr. S. Vigneshwari M.E., Ph.D.,

Dr. L. Lakshmanan M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **NAVEEN KRISHNA G (17SCS5133)** hereby declare that the Project Report entitled **“IDENTIFICATION AND CLASSIFICATION OF HATE SPEECH IN SOCIAL MEDIA”** done by me under the guidance of **Dr. Usha Nandini, M.E., Ph.D.**, at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr. S. Vigneshwari M.E., Ph.D.,** and **Dr. L. Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Usha Nandini, M.E., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Hate speech is a language that violates people by considering their specific characteristics such as physical appearances, sexual orientation, gender identity, national or ethnic origin, descent, religion or other. These hate language is expressed in different format, styles and styles targeting different groups and minorities which can occur with different linguistic forms such as in subtle forms or even through humour. In recent decades, with the excessive involvement of social networks and social media platforms in day to day life, the digital interaction between its users has become rapid, easy and time convenient. In spite of its numerous advantages, the anonymity associated with these interactions often leads to the emergence of more hateful, offensive and aggressive communication styles.

These intrude at a fast and uncontrollable pace and apparently causes severe damage to the targets, being vital that governments and social networking platforms are successful in detecting and regulating aggressive and hateful behaviors occurring on a regular basis on multiple online platforms. The hate speech detection due to its abstractness considered, its far from being trivial. Therefore this paper is proposed to deliver and complement current methodology and solutions on the detection of hate speech online, focusing on social media. The existing works to identify the two categories using the lexical methods showed ineffective performance metrics values which led to misclassification the works with supervised machine learning approaches indeed gave significant results in distinguishing hate and non hate but the presence or absence of certain words of both the classes can serve as both merit and demerit to achieve accurate classification. In this project a binary classification of tweets into hate speech non hate speech is performed using multi class classifiers among the two classifiers logistic regression and ensemble tree.

TABLE OF CONTENTS

CHAPTER NO.		PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	ix
	LIST OF ABBREVIATION	x
1	INTRODUCTION	1
2	LITERATURE SURVEY	4
	2.1 REVIEWS OF LITERATURE SURVEY	4
3	AIM AND SCOPE OF THE PROJECT	9
	3.1 AIM	9
	3.2 SCOPE OF THE PROJECT	9
	3.3 PROPOSED SYSTEM	9
4	ALGORITHMS AND TECHNIQUES USED	11
	4.1 INTRODUCTION	11
	4.2 ALGORITHMS USED	11
	4.2.1 LOGISTIC REGRESSION	11
	4.2.2 ENSEMBLE DECISION TREE	13
	4.3 METHODOLOGY	14
	4.3.1 DATA EXTRACTION	14
	4.3.2 DATA SPLIT	14
	4.3.3 DATA PREPROCESSING	15
	4.3.4 FEATURE EXTRACTION TECHNIQUES	17
	4.3.5 CLASSIFICATION	19

	4.3.6 EXPERIMENTAL SETUP	20
	4.4 HARDWARE AND SOFTWARE REQUIREMENTS	20
	4.4.1 HARDWARE REQUIREMENTS	20
	4.4.2 SOFTWARE REQUIREMENTS	20
5	RESULTS AND DISCUSSIONS	21
	5.1 RESULTS	21
	5.2 DISCUSSIONS	25
6	CONCLUSION AND FUTURE WORK	27
	6.1 CONCLUSION	27
	6.2 FUTURE WORK	28
	REFERENCES	29
	APPENDIX	31
	A. SAMPLE CODE	31
	B. SCREENSHOTS	40
	C. PUBLICATION	41

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.1	ARCHITECTURE DIAGRAM	10
4.1	LOGISTIC REGRESSION MODEL	12
5.1	WORD TFIDF VALUES	22
5.2	ACCURACY, F1 SCORE AND CONFUSION MATRIX OF LOGISTIC REGRESSION	23
5.3	CLASSIFIED TEST DATA INFORMATION	24

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
4.1	DATA SPLIT TABLE	15

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
NLP	NATURAL LANGUAGE PROCESSING
ML	MACHINE LEARNING
TF	TERM FREQUENCY
IDF	INVERSE DOCUMENT FREQUENCY
SEO	SEARCH ENGINE OPTIMIZATION
LR	LOGISTIC REGRESSION
ESC	ENSEMBLE TREE CLASSIFIER

CHAPTER 1

INTRODUCTION

The world of social media today provides a user to express what they feel without actually revealing their identity. But sometimes this may lead to spread of hate speech among the community. Spreading comments against or targeting a particular group of people and making criticism about their gender, race, colour, caste, sex and other aspects is considered as hate speech. For example, in may 2020, A African American named George Floyd's Murder created some social disputes among the public. During this incident many people made comments against the African American community in social media and used it's anonymity as a tool to hide themselves. This is a major example to prove that hate speech will have a huge negative impact which reflects on the community. With social media giants such as Twitter with almost 300 million and Facebook which as almost around 2.8 billion users worldwide, A system is required to regulate and monitor the hate speech and remove it when ever its possible.

According to the New York Times, "Personal attacks motivated by bias or prejudice reached a 16-year high in 2018, with a significant upswing in violence against Latinos outpacing a drop-in assault targeting Muslims and Arab-Americans." This rise can also be attributed to the rise of digital social media platforms which facilitate the spread of hate speech. Majority (59.6%) of all hate crime incidents involve race and ethnicity bias. The next highest bias was religion which accounted for 18.7% of all incidents. Our research validated these results when we observed that the most commonly occurring negative words in hate tweets were related to race or religion. These instances of vitriol can subconsciously incite hatred in our society, particularly in the minds of our impressionable younger generations. According to public research, 22% of respondents say they feel less safe in their community because of online hate speech and over 85% support setting up a task force to monitor cyber bullying. The issue has become so widespread now that the first lady of the United States, Melania Trump, has made it her main goal to combat cyberbullying and hate speech online.

The automatic detection of hate speech is a challenging task due to disagreements on different hate speech definitions. Therefore, some content might be hateful to some individuals and not to others, based on their concerned definitions. As the statistics show, not enough is being done to reduce the spread of hate speech. Some research works like Fortuna and Nune 2018 lists some main rules for hate speech detection. In this manner, (Waseem and Hovy 2016) have given us 11 parameters with which hate speech classification can be done.

Twitter officially says that, “Hateful conduct: You may not promote violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or disease. We also do not allow accounts whose primary purpose is inciting harm towards others on the basis of these categories”. In recent years, hate speech has been increasing in-person and online communication. The social media as well as other online platforms are playing an extensive role in the breeding and spread of hateful content – eventually which leads to hate crime. For example, according to recent surveys, the rise in online hate speech content has resulted in hate crimes including Trump's election in the US , the Manchester and London attacks in the UK, and terror attacks in New Zealand.

To tackle these harmful consequences of hate speech, different steps including legislation have been taken by the European Union Commission. Recently, the European Union Commission also enforced social media networks to sign an EU hate speech code to remove hate speech content within 24 hours. However, the manual process to identify and remove hate speech content is labor-intensive and time-consuming. Due to these concerns and widespread hate speech content on the internet, there is a strong motivation for automatic hate speech detection. The automatic detection of hate speech is a challenging task due to disagreements on different hate speech definitions. Therefore, some content might be hateful to some individuals and not to others, based on their concerned definitions. According to, hate speech is: “the content that promotes violence against

individuals or groups based on race or ethnic origin, religion, disability, gender, age, veteran status, and sexual orientation/gender identity”. Despite these different definitions, some recent studies claimed favorable results to detect automatic hate speech in the text. The proposed solutions

Employed the different feature engineering techniques and ML algorithms to classify content as hate speech. Regardless of this extensive amount of work, it remains difficult to compare the performance of these approaches to classify hate speech content. To the best of our knowledge, the existing studies lack the comparative analysis of different feature engineering techniques and ML algorithms.

CHAPTER 2

LITERATURE SURVEY

These days, hate speech is very common on social media. Therefore, in previous years, some of the researchers have applied a supervised ML-based text classification approach to classify hate speech content. Different researchers have employed different variety of feature representation techniques namely, dictionary-based, Bag-of-words-based, N-grams-based, TFIDF-based and Deep-Learning-based.

2.1 REVIEW OF LITERATURE

Title : Detecting Hate Speech on the World Wide Web

Author : W. Warner and J. Hirschberg

Year : 2012

This paper were one of the first initiatives to automate the detection of hate speech in the World Wide Web. This paper presents an approach to detecting hate speech in online text, where hate speech is defined as abusive speech targeting specific group characteristics, such as ethnic origin, religion, gender, or sexual orientation. While hate speech against any group may exhibit some common characteristics, we have observed that hatred against each different group is typically characterized by the use of a small set of high frequency stereotypical words; however, such words may be used in either a positive or a negative sense, making our task similar to that of words sense disambiguation. In this paper they described the definition of hate speech, the collection and annotation of our hate speech corpus, and a mechanism for detecting some commonly used methods of evading common “dirty word” filters. They described pilot classification experiments in which they classify anti semitic speech reaching an accuracy 94%, precision of 68% and recall at 60%, for an F1 measure of .6375.

Title : Locate the Hate: Detecting Tweets against Blacks

Author : . Kwok and Y. Wang

Year : 2013

Although the social medium Twitter grants users freedom of speech, its instantaneous nature and retweeting features also amplify hate speech. Because Twitter has a sizeable black constituency, racist tweets against blacks are especially detrimental in the Twitter community, though this effect may not be obvious against a backdrop of half a billion tweets a day. This applied a supervised machine learning approach, employing inexpensively acquired labelled data from diverse Twitter accounts to learn a binary classifier for the labels “racist” and “nonracist”. The classifier has a 76% average accuracy, suggesting that with further improvements, our work can contribute data on the sources of anti-black hate speech. The first racial oriented research was by Kwok et.al who decided to follow Warner and Hirschberg path used Unigram feature extraction technique and implemented a Naive Bayes supervised model to detect tweets as racist and non-racist tweets.

Title : Hate Speech, Machine Classification and Statistical Modelling of Information Flows on Twitter: Interpretation and Communication for Policy Decision Making

Author : P. Burnap and M. L. Williams

Year : 2014

This research was motivated to investigate the spread of hate speech immediately after Lee Rigby murder in UK. They trained a Bayesian Logistic Regression supervised classifier to find hateful and non-hateful tweets related to this particular event. They used typed dependencies along with Bag of words (BoW) features to distinguish hate speech utterances from clean speech ones which gave the F1 score value of 0.77. Bag-of-words approaches tend to have high recall but lead to high rates of false positives since the presence of offensive words can lead to the misclassification of tweets as hate speech.

Title : Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection

Author : H. Watanabe, M. Bouazizi, and T. Ohtsuki

Year : 2018

With the rapid growth of social networks and microblogging websites, communication between people from different cultural and psychological backgrounds has become more direct, resulting in more and more “cyber” conflicts between these people. Consequently, hate speech is used more and more, to the point where it has become a serious problem invading these open spaces. Hate speech refers to the use of aggressive, violent or offensive language, targeting a specific group of people sharing a common property, whether this property is their gender (i.e., sexism), their ethnic group or race (i.e., racism) or their beliefs and religion. While most of the online social networks and microblogging websites forbid the use of hate speech, the size of these networks and websites makes it almost impossible to control all of their content. Therefore, arises the necessity to detect such speech automatically and filter any content that presents hateful language or language inciting to hatred. In this paper, an approach was proposed to detect hate expressions on Twitter. This approach is based on unigrams and patterns that are automatically collected from the training set. These patterns and unigrams are later used, among others, as features to train a machine learning algorithm. The experiments on a test set composed of 2010 tweets show that our approach reaches an accuracy equal to 87.4% on detecting whether a tweet is offensive or not (binary classification), and an accuracy equal to 78.4% on detecting whether a tweet is hateful, offensive, or clean (ternary classification).

Title : Challenges in Discriminating Profanity from Hate Speech

Author : S. Malmasi and M. Zampieri

Year : 2018

In this study, the problem of distinguishing general profanity from hate speech in social media’s approached, something which has not been widely considered. Using a new dataset annotated specifically for this task, they employ supervised

classification along with a set of features that includes -grams, skip-grams and clustering-based word representations. They applied approaches based on single classifiers as well as more advanced ensemble classifiers and stacked generalization, achieving the best result of 80% accuracy for this 3-class classification task. Analysis of the results reveals that discriminating hate speech and profanity is not a simple task, which may require features that capture a deeper understanding of the text not always possible with surface -grams. The variability of gold labels in the annotated data, due to differences in the subjective adjudications of the annotators, is also an issue.

Title : Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter

Author : Z. Zhang and L. Luo

Year : 2018

In recent years, the increasing propagation of hate speech on social media and the urgent need for effective countermeasures have drawn significant investment from governments, companies, and researchers. A large number of methods have been developed for automated hate speech detection online. This aims to classify textual content into non-hate or hate speech, in which case the method may also identify the targeting characteristics (i.e., types of hate, such as race, and religion) in the hate speech. However, the significant difference between the performance of the two (i.e., non-hate v.s. hate) is noticed. In this work, we argue for a focus on the latter problem for practical reasons. It shows that it is a much more challenging task, as the analysis of the language in the typical datasets shows that hate speech lacks unique, discriminative features and therefore is found in the ‘long tail’ in a dataset that is difficult to discover. This then proposes Deep Neural Network structures serving as feature extractors that are particularly effective for capturing the semantics of hate speech. The methods are evaluated on the largest collection of hate speech datasets based on Twitter, and are shown to be able to outperform the

best performing method by up to 5 percentage points in macro-average F1, or 8 percentage points in the more challenging case of identifying hateful content. This work explored combining Convolutional and Gated recurrent unit networks, they also compared their model performance with all previous deep learning models. They stated that their work sets a new benchmark for future researches in this area.

Title : Inducing a Lexicon of Abusive Words – a Feature-Based Approach

Author : M. Wiegand, J. Ruppenhofer, A. Schmidt, and C. Greenberg

Year : 2018

This work addressed the detection of abusive words. The task is to identify such words among a set of negative polar expressions. They proposed novel features employing information from both corpora and lexical resources. These features are calibrated on a small manually annotated base lexicon which we use to produce a large lexicon. From this work it shows that the word-level information we learn cannot be equally derived from a large dataset of annotated microposts. It demonstrates the effectiveness of our (domain-independent) lexicon in the cross-domain detection of abusive microposts. This work used Lexical, linguistics and word embedding to extract features and classification of hate was done using SVM classifier which obtained an F1 score of 0.81.

CHAPTER 3

AIM AND SCOPE OF THE PROJECT

3.1 AIM

Websites offer an open space for people to discuss and share thoughts and opinions, their nature and the huge number of posts, comments and messages exchanged makes it almost impossible to control their content. Furthermore, given the different backgrounds, cultures and beliefs, many people tend to use and aggressive and hateful language when discussing with people who do not share the same backgrounds.

3.2 SCOPE OF THE PROJECT

Through social media platforms (such as Facebook, Twitter, YouTube, Instagram and Snapchat), 3.19 billion users converse and interact with each other by generating and sharing content. The age of digital media has allowed any online speech or content to be shared by one tap of a screen without a second thought for the consequences. Hate speech is likely to result in actual harm to those who are being targeted. To avoid such complications, there is a requirement for such a regulation. The proposed project lays a base for such problems.

3.3 PROPOSED SYSTEMS

The hate speech classification process is done using the text classification methodology which involves the preprocessing techniques, feature extraction techniques and machine learning algorithms. Identification and classification of hate speech is hindered by the noise in the dataset which results in reducing the accuracy of the machine learning model. The preprocessing of data set involves the procedures to remove the unfamiliar and unwanted characters used with informality. Feature selection is done to reduce the number of features without sacrificing the

performance of the classifier. The feature extraction technique that is used is TFIDF vectorizer. Binary classification is performed. The two classes' labels are “hate”, and “non-hate”.

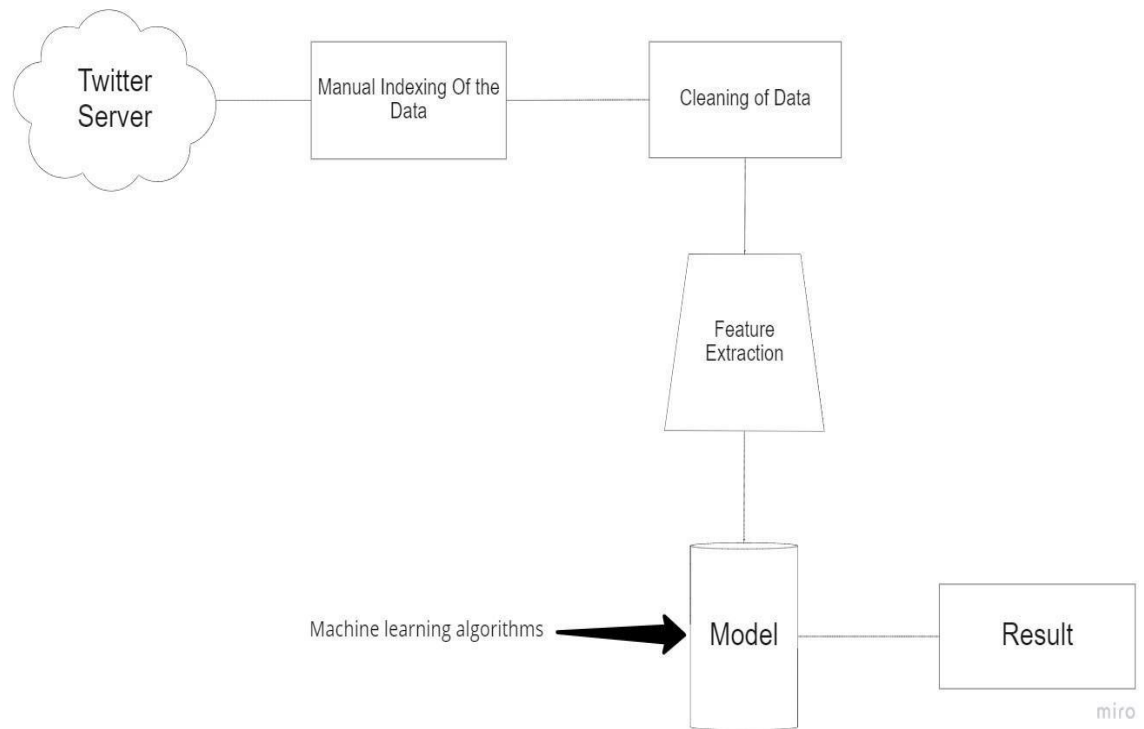


Fig. 3.1 : Architecture Diagram.

CHAPTER 4

ALGORITHMS AND TECHNIQUES USED

4.1 INTRODUCTION

Machine learning algorithms are programs (math and logic) that adjust themselves to perform better as they are exposed to more data. The “learning” part of machine learning means that those programs change how they process data over time, much as humans change how they process data by learning. So a machine-learning algorithm is a program with a specific way to adjusting its own parameters, given feedback on its previous performance in making predictions about a dataset. Machine learning is a class of methods for automatically creating models from data. Machine learning algorithms are the engines of machine learning, meaning it is the algorithms that turn a data set into a model. Which kind of algorithm works best (supervised, unsupervised, classification, regression, etc.) depends on the kind of problem you’re solving, the computing resources available, and the nature of the data. Ordinary programming algorithms tell the computer what to do in a straightforward way. For example, sorting algorithms turn unordered data into data ordered by some criteria, often the numeric or alphabetical order of one or more fields in the data.

4.2 ALGORITHMS USED

4.2.1 Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Sometimes logistic regressions are difficult to interpret; the Intellectus Statistics tool easily allows you to conduct the analysis, then in plain English interprets the output.

Logistic Regression is a statistical model used to determine if an independent variable has an effect on a binary dependent variable. This means that there are

only two potential outcomes given an input. For example, it may be used to determine if an email is spam, or not, using the rate of misspelled words, a common sign of spam. Other forms of regression analysis, like a linear regression, require the definition of a threshold to distinguish the binary classes. Linear regression allows for a probability to be established, but it must then be applied to a logistic regression to make the distinct classification.

A commonly used model is a sigmoid function. In the sigmoid function, also known as a squashing function, outputs are contained between the boundaries of 0 and 1. Here, we can use the model:

$$y = e^{(b_0 + b_1 \cdot x)} / (1 + e^{(b_0 + b_1 \cdot x)})$$

in the function above, there are variables b_0 , and b_1 . These are called the weights, or coefficient values. b_0 represents the bias, or intercept, and b_1 is the coefficient. These weights are learned and trained from the existing data set. The product of this formula will produce a percentage, or probability, that will be mapped over discrete classes. The defined separation between two classes is known as the decision boundary.

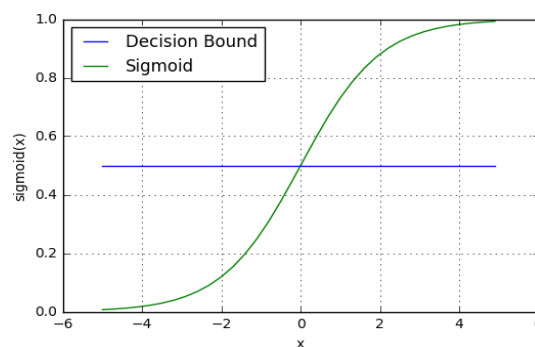


Fig. 4.1 : Logistic Regression Model

4.2.2 Ensemble Decision Tree

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble consists of only a concrete finite set of alternative models, but typically allows for much more flexible structure to exist among those alternatives.. Ensemble methods are a fantastic way to capitalize on the benefits of decision trees, while reducing their tendency to overfit. However, they can get pretty complex and can turn into black box models. One of the best things we can do as data scientists and machinelearning engineers is make sure we know what's really going on under the hood when we call that `sklearn.fit()` method. Decision Trees can be used for regression or classification, though they are more popular for classification problems. Generally, if you want to use a decision tree for a regression model, you should use an ensemble method. Decision Trees are non-parametric, which is just a fancy way to say that we aren't making any assumptions about how our data is distributed and our model's structure (parameters) will be determined from user input and the observations in our sample, rather than being fixed from the data. Non-parametric models are great when we have a lot of data, but not too much knowledge surrounding the data. Decision Trees are interpretable, which means that after we build the model, we can also make inferences about our data, not just predictions. In sklearn, we can do this using the `feature_importances_` attribute. Decision Trees are fast because they're simple and "greedy". Finally, data preprocessing is easier with Decision Trees because we don't have to scale our data — the splits that occur at each node are worried about one feature at a time! Depending on the algorithm used for determining split, categorical features may not have to be encoded numerically either. However, with scikit-learn's current CART algorithm (again, more later), you will want to preprocess categorical data with the recommended `sklearn.preprocessing.LabelEncoder`.

4.3 METHODOLOGY

4.3.1 Data Extraction

Twitter is the most popular social media which has hate speech content because it has a large amount of opinions with linguistic diversity. The data that is used in this paper are scraped from twitter using twitter scraper. The dataset contains 1K tweets in English. This dataset is classified into two classes “Hate” and “Non-Hate” by manual indexing. This manually indexed dataset is used to construct the features for each class and then used to train the classifier. The data that is used in this paper are scraped from twitter using twitter scraper. The dataset contains 10K tweets in English. This dataset is classified into two classes “Hate” and “Non- Hate” by manual indexing. This manually indexed dataset is used to construct the features for each class and then used to train the classifier.

4.3.2 Data Split

The preprocessed data is split into 80-20 ratio, that is., 80% of data is considered as Training Data and 20% of the data is considered as Test Data. The Table I shows the class-wise distribution of the overall dataset as well as data set after splitting. The 80% of training data will be used to train the machine learning model and the 20% of test data is used for the evaluation the classification model. The main aim of deciding the splitting ratio is that all three sets should have the general trend of our original dataset. If our dev set has very little data, then it is possible that we’ll end up selecting some model which is biased towards the trends only present in the dev set. Same is the case with training sets — too little data will bias the model towards some trends found only in that subset of the dataset. The models that we deploy are nothing but estimators learning the statistical trends in the data. Therefore, it is important that the data that is being used to learn and that being used to validate or test the model follow as similar statistical distribution as possible. One of the ways to achieve this as perfectly as possible is to select the subsets — here the training set, the dev set and/or the test set randomly. The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

Table 4.1: Data Split

	Class	Total Instances	Training Instance	Test Instances
0	Non - Hate	9300	7447	1853
1	Hate	700	553	147
Total		10000	8000	2000

4.3.3 Data Preprocessing

In order for machine learning algorithms to identify and classify hate speech on social media with high accuracy, noiseless clean data is needed. When dealing with a Twitter dataset, bear in mind that most tweets are short, contain worthless or unknown characters, and are written in a casual manner. Tweets, in particular, have a special format that must be omitted or parsed, since they contain usernames, URLs, and hashtags.

To preprocess your text simply means to bring your text into a form that is predictable and analyzable for your task. A task here is a combination of approach and domain. For example, extracting top keywords with TF-IDF (approach) from Tweets (domain) is an example of a Task. Identification and classification of hate speech is hindered by the noise in the dataset which results in reducing the accuracy of the machine learning model. The preprocessing of data set involves the procedures to remove the unfamiliar and unwanted characters used with informality.

Considering the twitter platform, the usage of casual and unstuffy languages is significantly high which makes text preprocessing much needed. In this project few preprocessing methods are used. We start with punctuation removal, stop-words removal. Then, the tweets are lowercased and finally lemmatization is done.

Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and etc. The intuition behind using stop words is that, by removing low information words from text, we can focus on the important words instead.

Lowercasing ALL your text data, although commonly overlooked, is one of the simplest and most effective form of text preprocessing. It is applicable to most text mining and NLP problems and can help in cases where your dataset is not very large and significantly helps with consistency of expected output.

Lemmatization on the surface is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that, lemmatization tries to do it the proper way. It doesn't just chop things off, it actually transforms words to the actual root. For example, the word “better” would map to “good”. It may use a dictionary such as WordNet for mappings or some special rule-based approaches.

WordNet is a lexical database of semantic relations between words in more than 200 languages. WordNet links words into semantic relations including synonyms, hyponyms, and meronyms. The synonyms are grouped into synsets with short definitions and usage examples. WordNet can thus be seen as a combination and extension of a dictionary and thesaurus. While it is accessible to human users via a web browser, its primary use is in automatic text analysis and artificial intelligence applications. WordNet was first created in the English language and the English WordNet database and software tools have been released under a BSD style license and are freely available for download from that WordNet website.

WordNet can be used to get information about the Synonyms (words that have the same meaning), Hypernyms (the generic term used to designate a class

of specifics), Hyponyms (a member of a class of terms), Holonyms (Name of a whole of which other words are a part), Meronyms (parts of the holonym).

WordNet also provides information on co-ordinate terms, derivatives, senses and more. For SEOs, understanding what the search engines "know" about your target terms/phrases is a step towards understanding how to construct excellent site architectures. The extent to which search engines may classify data is unknown, but one could certainly presume that any data available to WordNet is also able to be utilized by the commercial search engines.

4.3.4 Feature Extraction Techniques

In feature selection, The goal is to reduce the number of features without sacrificing the performance of the classifier. The feature extraction technique that is used is TFIDF vectorizer. The Term Frequency Inverse Document Frequency (TFIDF) is a measure that calculates how pertinent a character or a word is in the document. This is obtained by the multiplication of two metrics. number of times a particular word appears in a document and the inverse document frequency. In this paper both character and word TFIDF vectorizer is use to select the features needed for hate speech detection in twitter.

TF-IDF(Term Frequency/Inverse Document Frequency) is one of the most popular IR(Information Retrieval) technique to analyze how important a word is in a document. According to a research, 83% of text-based recommender systems uses TF-IDF. TF-IDF weighs the importance of words in a document. For example, "the" is commonly used in any documents so that TF-IDF does not consider "the" important to characterize documents. On contrary, "python" is used in IT relevant topic so that TF-IDF considers "python" as important feature word to recognize topic and category.

TF: Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

It's useful to understand how TF-IDF works so that you can gain a better understanding of how machine learning algorithms function. While machine learning algorithms traditionally work better with numbers, TF-IDF algorithms help them decipher words by allocating them a numerical value or vector. This has been revolutionary for machine learning, especially in fields related to NLP such as text analysis. To get a deeper understanding of how machine learning algorithms work, you can first learn how TF-IDF operates. Though machine learning algorithms are typically best at working with numbers, TF-IDF algorithms aid in the decoding of words by assigning a numerical value or vector to them. This has revolutionised machine learning, especially in fields related to natural language processing (NLP), such as text analysis.

In text analysis with machine learning, TF-IDF algorithms help sort data into categories, as well as extract keywords. This means that simple, monotonous tasks, like tagging support tickets or rows of feedback and inputting data can be done in seconds. TF-IDF algorithms are used in deep learning text processing to sort data into groups and extract keywords. Easy, repetitive processes, such as marking help tickets or rows of comments and inputting data, can now be completed in seconds.

Although it's a practical method, it's far from optimal, and from a theoretical perspective, it's open to some serious challenges. Content marketers also have a problem using TF-IDF based tools for a number of reasons. TF-IDF was created for informational retrieval purposes, not content optimization as some people have put forward. TF-IDF is a formula that counts the frequency of a term in a document within a corpus and dampens the weighting of terms that occur too frequently. While it is a realistic approach, it is far from perfect, and from a theoretical standpoint, it poses considerable obstacles. For a variety of factors, content marketers have difficulty using TF-IDF-based software. The TF-IDF was designed for information processing, not content optimization, as some have said. The TF-IDF formula counts the number of times a word appears in a text within a corpus and reduces the weighting of words that appear so often.

$TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document})$.

IDF: Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$IDF(t) = \log_e(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it})$.

4.3.5 Classification

Considering the identification of hate speech, the classification process is done by using supervised machine learning classification algorithms. Two ML algorithms are used in this paper for classifying hate and non-hate tweets such as Logistic Regression and Ensemble Tree. These two classifiers are largely used in the previous works of hate speech detection. scikit-learn package is used to train all the necessary models.

When the dependent variable is dichotomous, logistic regression is the best regression analysis to use (binary). The logistic regression, like all regression analyses, is a predictive analysis. To interpret data and explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables, logistic regression is used.

Ensemble approaches in statistics and machine learning use different learning algorithms to achieve greater predictive performance than any of the constituent learning algorithms alone. A machine learning ensemble, unlike a statistical ensemble in statistical mechanics, consists of only a concrete finite set of alternative models, but frequently makes for much more flexible structure to exist among those

alternatives. Ensemble techniques are a great way to take advantage of decision trees' advantages while minimising their propensity to overfit.

4.3.6 Experimental Setup

As the first step of the classification process, the tweets are preprocessed. In the preprocessing phase, All the tweets present in the dataset undergoes tokenization, then converted to lowercase, followed by stop words removal, removing extra spaces, mentions, links, punctuations and numbers and then undergoes lemmatization. As for the next step, feature extraction techniques word TFIDF vectorizer and character TFIDF vectorizer is performed. Lastly, two supervised Machine Learning algorithms are used to identify and detect hate speech.

4.4 HARDWARE AND SOFTWARE REQUIREMENTS

4.4.1 Software Requirements

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

4.4.2 Hardware Requirements

Processor : Pentium III

Hard Disk : Minimum 80 GB

RAM : Minimum 2GB

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 RESULTS

In the process of classification, the two performance metrics are taken into account to evaluate the classifiers such as accuracy and F1 score. Both the classifiers Logistic Regression and Ensemble tree classifiers performed well giving high accuracies with the TFIDF vectorizers. But the Ensemble Tree classifier gave both F1 score and accuracy higher than Logistic Regression such as 0.9505 and 0.7545 respectively. Considering the confusion matrix of the Logistic Regression classifier, out of 147 hate tweets belonging to hate speech of the test set, only 15 tweets were correctly classified and the remaining 132 tweets were incorrectly classified as Non – Hate. From the 1853 instances belonging to the Non-Hate class, almost 1849 tweets were correctly classified as non – hate and only 4 tweets happened to be misclassified as hate speech. In terms of the confusion matrix of the Ensemble tree classifier, out of 147 hate tweets belonging to hate speech of the test set, 57 tweets were correctly classified and the remaining 90 tweets were incorrectly classified as Non – Hate. From the 1853 instances belonging to the Non-Hate class, almost 1844 tweets were correctly classified as non – hate and only 9 tweets happened to be misclassified as hate speech. In this project, two classifiers are used over two TFIDF feature extraction vectorizer such as TFIDF Word vectorizer and TFIDF Character vectorizer. The experimental results showed that the Ensemble tree classifier with the both the vectorizers showed the best results. The reasons behind the results are analyzed.

In the process of text classification, feature selection is an important aspect. In this paper, TFIDF vectorizers are used as several research papers have experimented and proved that TFIDF feature extraction technique has given higher performance than other binary and term frequency feature extraction methods. Also from the experimental results, it is certain that the Feature TFIDF scores helped to achieve high accuracy scores.

This may be because of some words that considered to be hate are used day-today lives frequently. The tweets which contain those words might not be deliberated as a hate speech. To detect whether the comment is slang or abusive word is intended to be hate or not, TFIDF approach is used here because it will assign a low weightage to such kind of words. Considering the Machine Learning classification algorithms, there is an absolute necessity to compare different classification algorithms in order to identify the best performing classifier in the dataset taken. There are also many research papers which proved that there is no one permanent algorithm which gives high performance on all dataset. Hence, in our paper, two classification algorithms such as Logistic Regression and Ensemble tree classifiers are used. From the experimental results, although both the algorithms performed well having high accuracy value, Ensemble tree model has high value for both accuracy and F1 score. Hence the Ensemble tree model is chosen as the best performing model for our dataset. In this project, binary classification is performed. The two classes' labels are "hate", and "non-hate".

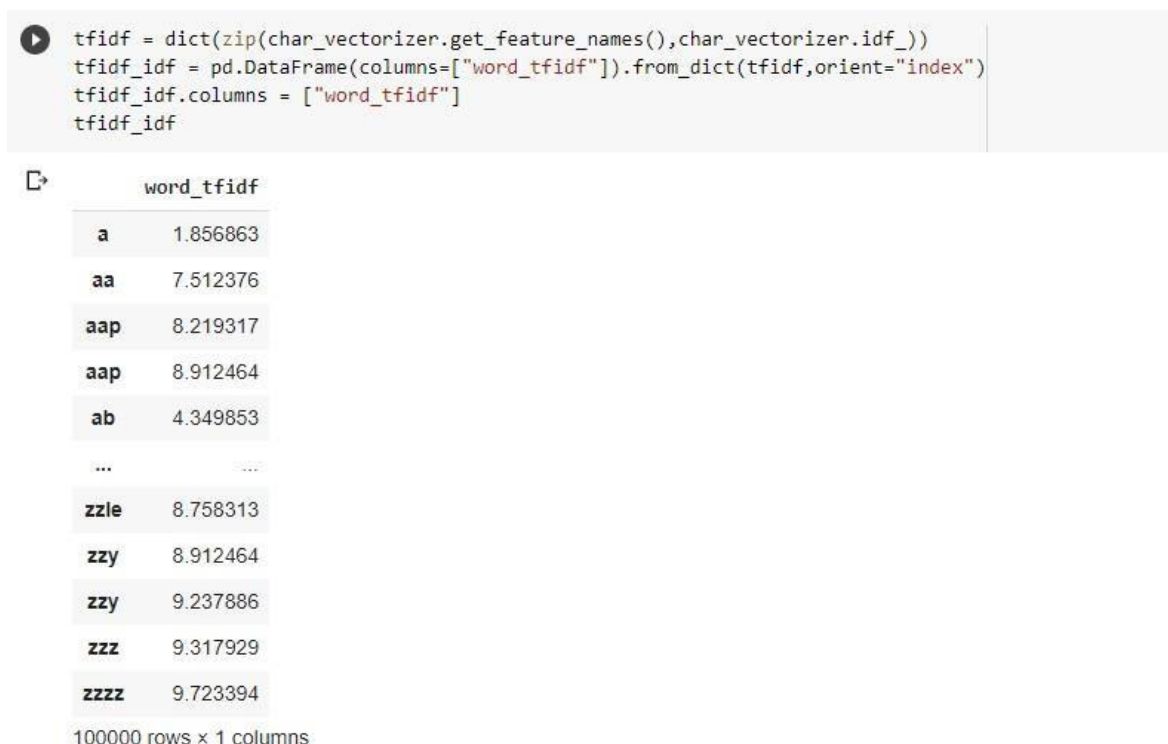


Fig. 5.1 : Word TFIDF values

Term Frequency/Inverse Document Frequency (TF-IDF) is a common IR (Information Retrieval) technique for determining the importance of a word in a document. According to a survey, TF-IDF is used by 83 percent of text-based recommender programmes. The TF-IDF evaluates the significance of terms in a text. For eg, since “the” is widely used in all documents, the TF-IDF does not consider “the” to be relevant in document classification. In the other hand, since “python” is used in IT-related topics, TF-IDF finds “python” to be an appropriate function term for recognising subject and category.

```

▶ train_target = train['label']
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

train_x, val_x, train_y, val_y = train_test_split(train_features, train_target)

model = LogisticRegression(solver='newton-cg')
model.fit(train_x, train_y)
preds = model.predict(val_x)
print(accuracy_score(preds, val_y))
print(f1_score(val_y, preds, average="macro"))
CM = confusion_matrix(val_y, preds)
TN = CM[0][0]
print(TN)
FN = CM[1][0]
print(FN)
TP = CM[1][1]
print(TP)
FP = CM[0][1]
print(FP)

```

```

↳ 0.925531914893617
0.48066298342541436
174
14
0
0

```

Fig. 5.2 : Accuracy, F1 score and confusion matrix of Logistic Regression

When the dependent variable is dichotomous, logistic regression is the safest regression analysis to use (binary). The logistic regression, like all regression analyses, is a statistical analysis. To characterise data and illustrate the relationship

between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables, logistic regression is used. Logistic regressions can be difficult to understand at times; the Intellectus Statistics tool makes it simple to perform the study and then interprets the results in plain English.

```
[ ] submission = pd.DataFrame()
    preds = model.predict(test_features)
    submission['id'] = test.id
    submission['label'] = preds
    submission.head()
```

	id	label
0	751	0
1	752	0
2	753	0
3	754	0
4	755	0

```
▶ submission['label'].value_counts()
```

```
0    249
1      1
Name: label, dtype: int64
```

```
[ ] submission.to_csv('submission.csv', index=False)
```

Fig. 5.3 : Classified Test Data Information

The tweets are preprocessed as the first step in the classification process. During the preprocessing step, all of the tweets in the dataset are tokenized, then translated to lowercase, stop words are removed, extra spaces, mentions, connections, punctuation, and numbers are removed, and then lemmatization is performed. The attribute extraction techniques word TFIDF vectorizer and character TFIDF vectorizer are used in the next step. Finally, to recognise and detect hate speech, two supervised Machine Learning algorithms are used.

5.2 DISCUSSION

Two classifiers are used in this paper over two TFIDF attribute extraction vectorizers, the TFIDF Word vectorizer and the TFIDF Character vectorizer. The Ensemble tree classifier with both vectorizers generated the best results, according to the results of the experiments. The explanations for the outcomes are investigated.

The features selection is an important part of the text classification process. TFIDF vectorizers are used in this paper since several research papers have tested and shown that TFIDF feature extraction outperforms other binary and term frequency feature extraction methods. It is also evident from the experimental results that the Feature TFIDF scores assisted in achieving high accuracy scores. This may be attributed to the fact that certain words that are perceived to be hateful are commonly used in daily life. The tweets which contain those words might not be deliberated as a hate speech. To detect whether the comment is slang or abusive word is intended to be hate or not, TFIDF approach is used here because it will assign a low weightage to such kind of words.

When it comes to Machine Learning classification algorithms, comparing various classification algorithms is a must in order to find the best performing classifier in the dataset. There are also several research papers that show that there is no one-size-fits-all algorithm that performs well across all datasets. As a result, we use two classification algorithms in our paper: Logistic Regression and Ensemble Tree Classifiers. Although both algorithms performed well with high

precision values, the Ensemble tree model has a high value for both accuracy and F1 score, according to the experimental findings. Hence the Ensemble tree model is chosen as the best performing model for our dataset.

Binary classification is used in this project. The labels for the two disciplines are "hate" and "non-hate." The vectorizers and classifiers performed exceptionally well for the "non-hate" class among these classes. The models, on the other hand, underperformed when it came to "hate."

According to the Ensemble Tree classifier's confusion matrix, 39 percent of genuine hate speech tweets were misclassified as non-hate speech using the best performing model. This might be due to the fact that the misclassified tweets don't contain any words that clearly demonstrate hate speech, such as "f**k" and "ni**a." The classification model neglected to classify the tweets with sarcastic hate content as hatred. For example, tweets like "carolyn cooper - ugly, poor, ignorant and black!".

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The hate speech classification process is done using the text classification methodology which involves the preprocessing techniques, feature extraction techniques and machine learning algorithms.

Both the TFIDF char vectorizer and TFIDF word vectorizer performs very well with both the Logistic Regression algorithm and Ensemble tree model. Although both the models performed well, Ensemble tree had high scores in terms of all metrics. Also, the misclassification of hate tweets in Ensemble tree classifier was low as compared to LR classifier. Hence Ensemble Tree was chosen to be the best performing model for the dataset chosen in this paper.

It can be seen from the results that while classifying the Non- hate class tweets, the models achieved relatively high accuracy. But while detecting the hate speech, the model is showing lower performance. This might be because of the absence of the words that strongly exhibit the hate content. In addition to that, the sarcastic comments tend to get classified as Non – hate by the machines. Moreover, the scarcity in the training data may also be a factor of misclassification. In order to avoid this kind of misclassification in our system and inorder to acquire more accurate classification results, more training data of hate speech which are present without the explicit mentions of particular racial or homophobic comments must be used.

The detection of hate speech depends on we people's subjective understanding of what hate speech s. So the future objective is to understand the social biases and make the model to correctly spot these biases.

6.2 FUTURE WORK

This project can be further tuned up by implementing ternary classification with three classes such as hate, offensive and neither. The dependent, or "goal" variable is divided into three classes using a ternary division — ternary comes from a latin root that sounds similar to "tri." This operation may be performed by a variety of models, but they can vary slightly from binary problems. Ternary classifications, as well as any model with more than two goals, are classified as multinomial or multiclass problems.

The ternary classification is processed based on preprocessing technique, and the results of tweets sent by the users are obtained. We can also use a hybridization technique using two optimization algorithms and one machine learning classifier, namely particle swarm optimization and genetic algorithm and decision tree for classification accuracy by sentiment analysis.

In such case, considering only the accuracy may be misleading. Thus, recall is considered which gives the proportion of hate tweets that are correctly classified. If the recall is high, it refers that a large number of hate tweets are correctly classified as hate tweets, thus favoring the classification model. Hence the models which have high scores for both accuracy and recall would be chosen as best performing model.

REFERENCES

- [1] Abro,S., Shaikh,S., Khand,H,Z.,Ali,Z., Khan,S., Mujtaba,G., Automatic Hate Speech Detection using Machine Learning: A Comparative Study. International Journal of Advanced Computer Science and Applications (IJACSA),2020, Volume 11, Issue 8.
- [2] Burnap, P. and M.L. Williams, Us and them: identifying cyber hate on Twitter across multiple protected characteristics. EPJ Data Science, 2016. 5(1): p. 11.
- [3] Cavnar, W.B. and J.M. Trenkle. N-gram-based text categorization. in Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval. 1994. Citeseer.
- [4] Davidson, T., Warmesley, D., Macy,M., and Weber, I.. Automated hate speech detection and the problem of offensive language. arXiv preprint arXiv:1703.04009, 2017.
- [5] Fortuna, P. and S. Nunes, A survey on automatic detection of hate speech in text. ACM Computing Surveys (CSUR), 2018. 51(4): p. 85.
- [6] Kwok,I.,andWang,Y. 2013. Locate the hate: Detecting tweets against blacks. In AAAI.
- [7] Mujtaba, G., et al., Prediction of cause of death from forensic autopsy reports using text classification techniques: A comparative study. Journal of forensic and legal medicine, 2018. 57: p. 41-50.
- [8] Wang, W.,Chen, L., Thirunarayan, K., and Sheth, A. P. 2014. Cursing in english on twitter. In CSCW, 415–425.
- [9] Warner, W., and Hirschberg, J. 2012. Detecting hate speech on the world wide web. In LSM, 19–26.
- [10] Waseem, Z. and D. Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. in Proceedings of the NAACL student research workshop. 2016.

- [11] Waseem,Z.,2016. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In Proceedings of the First Workshop on NLP and Computational Social Science. Association for Computational Linguistics, Austin, Texas, 138–142.
- [12] Z. Zhang and L. Luo, "Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter," vol. 1, no. 0, pp. 1–5, 2018.
- [13] Zhang, M.-L. and Z.-H. Zhou, A k-nearest neighbor based algorithm for multi-label classification. GrC, 2005. 5: p. 718-721.

APPENDIX

A. SAMPLE CODE

```
{  
  "cell_type": "code",  
  "metadata": {  
    "id": "-bPkWtMdAy4h",  
    "colab": {  
      "base_uri": "https://localhost:8080/",  
      "height": 419  
    },  
    "outputId": "31ef2615-3d09-4890-cd09-9bfe281a245a"  
  },  
  "source": [  
    "tfidf                                                    =  
dict(zip(word_vectorizer.get_feature_names(),word_vectorizer.idf_))\n",  
    "tfidf_idf                                                =  
pd.DataFrame(columns=[\"word_tfidf\"]).from_dict(tfidf,orient=\"index\")\n",  
    "tfidf_idf.columns = [\"word_tfidf\"]\n",  
    "tfidf_idf"  
  ],  
  "execution_count": 8,  
  "outputs": [  
    {  
      "output_type": "execute_result",
```

```

"data": {
    "text/html": [
        "<div>\n",
        "<style scoped>\n",
        "  .dataframe tbody tr th:only-of-type {\n",
        "    vertical-align: middle;\n",
        "  }\n",
        "\n",
        "  .dataframe tbody tr th {\n",
        "    vertical-align: top;\n",
        "  }\n",
        "\n",
        "  .dataframe thead th {\n",
        "    text-align: right;\n",
        "  }\n",
        "</style>\n",
        "<table border='1' class='dataframe'>\n",
        " <thead>\n",
        "   <tr style='text-align: right;'>\n",
        "     <th></th>\n",
        "     <th>word_tfidf</th>\n",
        "   </tr>\n",
        " </thead>\n",

```

```

" <tbody>\n",
    " <tr>\n",
    " <th>aaahh</th>\n",
    " <td>7.215608</td>\n",
    " </tr>\n",
    " <tr>\n",
    " <th>aaahh nabilah</th>\n",
    " <td>7.215608</td>\n",
    " </tr>\n",
    " <tr>\n",
    " <th>aaahhhhh</th>\n",
    " <td>7.215608</td>\n",
    " </tr>\n",
    " <tr>\n",
    " <th>aap</th>\n",
    " <td>7.215608</td>\n",
    " </tr>\n",
    " <tr>\n",
    " <th>aaps</th>\n",
    " <td>7.215608</td>\n",
    " </tr>\n",
    " <tr>\n",

```

```

"    </tr>\n",
"    <tr>\n",
"        <th>zone</th>\n",
"        <td>7.215608</td>\n",
"    </tr>\n",
"    <tr>\n",
"        <th>zone goodtogreat</th>\n",
"        <td>7.215608</td>\n",
"    </tr>\n",
"    <tr>\n",
"        <th>zoo</th>\n",
"        <td>7.215608</td>\n",
"    </tr>\n",
"    <tr>\n",
"        <th>zpamdelacruz</th>\n",
"        <th>zpamdelacruz wed</th>\n",
"        <td>7.215608</td>\n",
"    </tr>\n",
" </tbody>\n",
"</table>\n",
"<p>8726 rows × 1 columns</p>\n",
"</div>"

```

```

],

    },

    "metadata": {

        "tags": []

    },

    "execution_count": 8

}

{

    "cell_type": "code",

    "metadata": {

        "id": "3oQZQ9wLAy4h",

        "colab": {

            "base_uri": "https://localhost:8080/",

            "height": 419

        },

        "outputId": "0c0dda86-74e2-49b4-e94e-d16022c3e781"

    },

    "source": [

        "tfidf = dict(zip(char_vectorizer.get_feature_names(),char_vectorizer.idf_))\n",

        "tfidf_idf\n",

        "pd.DataFrame(columns=[\"word_tfidf\"]).from_dict(tfidf,orient=\"index\")\n",

        "tfidf_idf.columns = [\"word_tfidf\"]\n",

```

```

"tfidf_idf"

],

"execution_count": 9,

"outputs": [

{

  "output_type": "execute_result",

  "data": {

    "text/html": [

      "<div>\n",

      "<style scoped>\n",

      "  .dataframe tbody tr th:only-of-type {\n",

      "    vertical-align: middle;\n",

      "  }\n",

      "\n",

      "  .dataframe tbody tr th {\n",

      "    vertical-align: top;\n",

      "  }\n",

      "\n",

      "  .dataframe thead th {\n",

      "    text-align: right;\n",

      "  }\n",

      "</style>\n",

      "<table border='1' class='dataframe'>\n",

```

```

" <thead>\n",
    "<th>word_tfidf</th>\n", "
    </tr>\n",
    "    <tr>\n",
    "        <th>a</th>\n",
    "        <td>1.817445</td>\n",
    "text/plain": [
    "    word_tfidf\n",
"metadata": {
    "id": "-Xeck41YAy4h",
    "colab": {
        "base_uri": "https://localhost:8080/"
    },
    "outputId": "16d4645c-75ac-48a4-96a9-d988d1acc63e"
},
"source": [
    "train_target = train['label']\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.metrics import accuracy_score, f1_score, confusion_matrix\n",
    "\n",
    "train_x, val_x, train_y, val_y = train_test_split(train_features,train_target)\n",
    "\n",

```

```

"model = LogisticRegression(solver='newton-cg')\n",
"model.fit(train_x,train_y)\n",

"preds = model.predict(val_x)\n",
    "print(accuracy_score(preds,val_y))\n",
    "print(f1_score(val_y, preds, average='macro'))\n",
    "CM = confusion_matrix(val_y, preds)\n",
    "TN = CM[0][0]\n",
    "print(TN)\n",
    "FN = CM[1][0]\n",
    "print(FN)\n",
    "TP = CM[1][1]\n",
    "print(TP)\n",
    "FP = CM[0][1]\n",
    "print(FP)\n"
],
"execution_count": 10,
"outputs": [
    "name": "stdout"
}
"cell_type": "code",

```

```

"metadata": {
    "id": "ItATQ9fKAy4i",
    "colab": {
        "base_uri": "https://localhost:8080/"
    },
    "outputId": "35a855c9-dd34-4b51-e9a1-9eb2de0f65d5"
},
"source": [
    "model = ensemble.ExtraTreesClassifier(n_estimators=100)\n",
    "model.fit(train_x,train_y)\n",
    "preds = model.predict(val_x)\n",
    "print(accuracy_score(preds,val_y))\n",
    "print(f1_score(val_y, preds, average=\"macro\"))\n",
    "CM = confusion_matrix(val_y, preds)\n",
    "TN = CM[0][0]\n",
    "print(TN)\n",
    "FN = CM[1][0]\n",
    "print(FN)\n",
    "TP = CM[1][1]\n",
    "print(TP)\n",
    "FP = CM[0][1]\n",
    "print(FP)"
]

```

B. SCREENSHOTS

[illegible]

```
tfidf = dict(zip(word_vectorizer.get_feature_names(), word_vectorizer.idf_))
tfidf_idf = pd.DataFrame(columns=["word_tfidf"]).from_dict(tfidf, orient="index")
tfidf_idf.columns = ["word_tfidf"]
tfidf_idf
```

	word_tfidf
aaa	10.416541
aap	8.306328
aap congress	10.011076
aap spokesperson	9.605611
aaps	10.193398
...	...
zoo	9.094785
zootopia	9.856925
zoro	10.193398
zoro jrock	10.193398
zumba	10.011076

20000 rows x 1 columns

My Drive - Google Drive

colab.research.google.com/drive/1Dp4WVh2ccuk9dm9-2NmcZT3pc

Final_Year_Project_Code.ipynb

File Edit View Insert Runtime Tools Help Last saved at 3:20 PM

Comment Share

Connecting Editing

```
0
0

model = ensemble.ExtraTreesClassifier(n_estimators=100)
model.fit(train_x, train_y)
preds = model.predict(val_x)
print(accuracy_score(preds, val_y))
print(f1_score(val_y, preds, average="macro"))
CN = confusion_matrix(val_y, preds)
TN = CN[0][0]
print(TN)
FN = CN[1][0]
print(FN)
TP = CN[1][1]
print(TP)
FP = CN[0][1]
print(FP)

0.9574468885184383
0.5898718382513681
179
8
1
0

[ ] submission = pd.DataFrame()
preds = model.predict(test_features)
submission['id'] = test.id
submission['label'] = preds
submission.head()

id label
0 751 0
1 752 1
2 753 0
3 754 0
4 755 0

[ ] submission['label'].value_counts()

0 245
```

Type here to search

ENG 03:25 PM 20-02-2021

IDENTIFICATION AND CLASSIFICATION OF HATE SPEECH IN SOCIAL MEDIA

Naveen Krishna G

Department of Computer Science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India

Abstract— Technology mixed with society is doing incredible things in today's world. People who are attracted to social networks because of their privacy, on the other hand, are dragging out the nastiest of people in the form of hate speech. Hate speech on social media is a significant cultural concern that can exacerbate abuse ranging from lynching to ethical cleansing. Current works that used lexical approaches to differentiate two groups yielded low output metrics scores, resulting in significant misclassification. Since use of controlled machine learning methods yielded promising results in discriminating between hate and non hate, the appearance or the absence of specific terms from both groups will act as both a merit and a detriment to correct classification. Using binary classifiers, this method separates tweets into hate speech and non-hate speech types. Logistic Regression and Ensemble Tree Classifier are the two classifiers used. As can be shown, Ensemble Tree Classifier outperforms for the TFIDF feature extraction technique used, with a cumulative accuracy of 0.95.

Keywords—Hate speech; Natural Language Processing; Text Preprocessing; Feature Extraction; Text Classification; Machine Learning

I. INTRODUCTION

The world of social media today provides a user to express what they feel without actually revealing their identity. But sometimes this may lead to spread of hate speech among the community. Spreading comments against or targeting a particular group of people and making criticism about their gender, race, colour, caste, sex and other aspects is considered as hate speech. For example, in May 2020, A African American named George Floyd's Murder created some social disputes among the public. During this incident many people made comments against the African American community in social media and used it's anonymity as a tool to hide themselves. This is a major example to prove that hate speech will have a huge negative impact which reflects on the community. With social media giants such as Twitter with almost 300 million and

Facebook which has almost around 2.8 billion users worldwide, A system is required to regulate and monitor the hate speech and remove it when ever its possible.

Some research works like Fortuna and Nune 2018 lists some main rules for hate speech detection. In this manner, (Waseem and Hovy 2016) have given us 11 parameters with which hate speech classification can be done. We also do not allow accounts whose primary purpose is inciting harm towards others on the basis of these categories". In this paper, A machine learning model is being trained in order to classify given tweets as hate or non-hate. From the obtained results its evident that the feature extraction part plays a major role in hate speech detection and gives a clear idea about the challenges faced while doing a accurate classification of tweets.

II. RELATED WORK

In the year 2013, Kwok et al. Despite the fact that social media platforms like twitter enables users to express their opinions on the public platform, it also in some ways encourage the amplification of hate comments. Because Twitter has a good amount of users from the black community, racist comments against these community creates a malignant Twitter community, even if this does not create a large scale impact against a half a billion tweets a day. This paper approaches this problem with a supervised machine learning technique and classifies the tweet to "racist" and "nonracist". N. Djuric et al. discussed the topic of recognising hate speech in user comments in 2015. Hate speech is characterised as "abusive speech aimed at members of a social community based on their race, belief, or gender." This problem is spreading to websites that allow users to leave reviews, negatively impacting their company and overall experience. This paper suggests using neural language models to learn a low-dimensional representation of user comments that can be used as an input to a classification algorithm. This method gives an 80% accuracy while implementing Logistic Regression algorithm. In this method, The binary classification of clean and hateful speech is done by "Paragraph2vec" and CBOW (Continuous Bag of Words) techniques. In

2016, Many other scholars have used Waseem et al work as a base to study predictive features for hate speech and clean speech detection. They granted us access to their massive 16k Twitter data corpus, which is specifically used for hate speech-related works in English. Their work was focused on classifying hate tweets from non-hate tweets by using character n-grams feature extraction technique and they trained a Logistic Regression model with the F1 accuracy score of 0.73. In 2018, Zhang et al. identified that In recent times, the amount of hate comments targeting a particular community on social media platforms and the requirement for regulation of such problems is being the major concern of investment for government, researchers and companies. Hate speech can be detected automatically using a variety of methods. These methods try to classify a textual content into hate or non-hate, where the method also tries to identify the particular targeting characteristics in hate speech such as type of hate like religion or race. However, the difference between performance of each class (non-hate vs hate) is noted. This paper suggests using a Deep Neural Network as a function extractor to capture the semantics of hate speech more effectively. This approach is measured using Twitter's hate speech datasets, and it outperforms the strongest method by almost 5% in macro-average F1 and 8% in more complicated situations. During the year 2016, Chikashi Nobata et al. mentioned that Hate speech language detection on social media platforms is becoming an unavoidable problem in recent times. Many current commercial systems use regular expressions and blacklists, however this method is ineffective while working with more harmless, butterfingered hate speech statements. In this paper, to detect hate language on online user comments, they developed a machine learning based solution which is more effective than the deep learning approach. They also created a corpus of abusive language annotated user comments. As a final step, a detection tool is used to analyse the abusive content.

III. PROPOSED SYSTEM

A. Data

Twitter is the most popular social media which has hate speech content because it has a large amount of opinions with linguistic diversity. The data that is used in this paper are scraped from twitter using twitter scraper. The dataset contains 1K tweets in English. This dataset is classified into two classes "Hate" and "Non-Hate" by manual indexing. This manually indexed dataset is used to construct the features for each class and then used to train the classifier. The data that is used in this paper are scraped from twitter using twitter scraper. The dataset contains 10K tweets in English. This dataset

is classified into two classes "Hate" and "Non-Hate" by manual indexing. The features for each class are designed using this manually indexed dataset, which is then used to train the classifier.

B. Data Split

The preprocessed data is divided into an 80-20 ratio in this paper, which means that 80 percent of the data is considered Training Data and 20 percent is considered Test Data. Table I represents the overall dataset's class-wise distribution as well as the data set after dividing. The machine learning model will be trained with 80% of the training data and 20% of the test data will be used to evaluate the classification model.

C. Data Preprocessing

Identification and classification of hate speech is hindered by the noise in the dataset which results in reducing the accuracy of the machine learning model. The preprocessing of data set involves the procedures to remove the unfamiliar and unwanted characters used with informality. Considering the twitter platform, the usage of casual and unstuffy languages is significantly high which makes text preprocessing much needed. In this paper few preprocessing methods are used. We start with punctuation removal, stop-words removal. Then, the tweets are lowercased and finally lemmatization is done.

Table I – Data Split

	Class	Total Instances	Training Instance	Test Instances
0	Non - Hate	9300	7447	1853
1	Hate	700	553	147
		10000	8000	2000

D. Feature Extraction Techniques

The objective of feature selection is to limit the number of features while maintaining the classifier's performance. TFIDF vectorizer is the feature extraction method used. TFIDF (Term Frequency Inverse Document Frequency) is a metric that determines how important a character or word is in a document. This is calculated by multiplying two metrics. The inverse document frequency and the number of times a specific word appears in a

document. The character and word TFIDF vectorizers are used in this paper to select the characteristics required for hate speech detection on Twitter.

E. Classification

Considering the identification process of the hate comments, the classification process is done by using supervised machine learning classification algorithms. Two ML algorithms are used in this paper for classifying hate and non-hate tweets such as Logistic Regression and Ensemble Tree. These two classifiers are largely used in previous works of hate speech detection. scikit-learn package is used to train all the necessary models.

F. Experimental Setup

As first step of our classification process, the tweets are preprocessed. In the preprocessing phase, All the tweets present in the dataset undergoes tokenization, then converted to lowercase, followed by stop words removal, removing extra spaces, mentions, links, punctuations and numbers and then undergoes lemmatization. As for the next step, feature extraction techniques word TFIDF vectorizer and character TFIDF vectorizer is performed. Lastly, two supervised Machine Learning algorithms are used to identify and detect hate speech.

IV. RESULTS

Two performance metrics, such as precision and F1 score, are used in the assessment of classifiers during the classification process. Both the classifiers Logistic Regression and Ensemble tree classifiers performed well giving high accuracies with the TFIDF vectorizers. But the Ensemble Tree classifier gave both F1 score and accuracy higher than Logistic Regression such as 0.9505 and 0.7545 respectively.

Furthermore, the confusion matrix of both classifiers using TFIDF vectorizers is shown in Tables II and III.

Considering the confusion matrix of the Logistic Regression classifier, out of 147 hate tweets belonging to hate speech of the test set, only 15 tweets were correctly classified and the remaining 132 tweets were incorrectly classified as Non - Hate. From the 1853 instances belonging to the Non-Hate class, almost 1849 tweets were correctly classified as non - hate and only 4 tweets happened to be misclassified as hate speech.

In terms of the confusion matrix of the Ensemble tree classifier, out of 147 hate tweets belonging to hate speech of the test set, 57 tweets were correctly

classified and the remaining 90 tweets were incorrectly classified as Non - Hate. From the 1853 instances belonging to the Non-Hate class, almost 1844 tweets were correctly classified as non - hate and only 9 tweets happened to be misclassified as hate speech.

Table II – Confusion Matrix of Logistic Regression

Class	Classified as	
	Hate	Non - Hate
Hate	15	4
Non - Hate	132	1849

Table III – Confusion Matrix of Ensemble Tree Classifier

Class	Classified as	
	Hate	Non - Hate
Hate	57	9
Non - Hate	90	1844

V. DISCUSSION

Two classifiers are used in this paper over two TFIDF attribute extraction vectorizers, the TFIDF Word vectorizer and the TFIDF Character vectorizer. The Ensemble tree classifier with both vectorizers generated the best results, according to the results of the experiments. The explanations for the outcomes are investigated.

The features selection is an important part of the text classification process. TFIDF vectorizers are used in this paper since several research papers have tested and shown that TFIDF feature extraction outperforms other binary and term frequency feature extraction methods. It is also evident from the experimental results that the Feature TFIDF scores assisted in achieving high accuracy scores. This may be attributed to the fact that certain words that are perceived to be hateful are commonly used in daily life. The tweets which contain those words might not be deliberated as a hate speech. To detect whether the comment is slang or abusive word is intended to be hate or not, TFIDF approach is used here because it will assign a low weightage to such kind of words.

When it comes to Machine Learning classification algorithms, comparing various classification algorithms is a must in order to find the best performing classifier in the dataset. There are also several research papers that show that there is no one-size-fits-all algorithm that performs well across all datasets. As a result, we use two classification algorithms in our paper: Logistic Regression and Ensemble Tree Classifiers. Although both algorithms performed well with high precision values, the Ensemble tree model has a high value for both accuracy and F1 score, according to the experimental findings. Hence the Ensemble tree model is chosen as the best performing model for our dataset.

Binary classification is used in this paper. The labels for the two disciplines are "hate" and "non-hate." The vectorizers and classifiers performed exceptionally well for the "non-hate" class among these classes. The models, on the other hand, underperformed when it came to "hate."

According to the Ensemble Tree classifier's confusion matrix, 39 percent of genuine hate speech tweets were misclassified as non-hate speech using the best performing model. This might be due to the fact that the misclassified tweets don't contain any words that clearly demonstrate hate speech, such as "f**k" and "ni**a." The classification model neglected to classify the tweets with sarcastic hate content as hatred. For example, tweets like "carolyn cooper - ugly, poor, ignorant and black!".

VI. CONCLUSION

Hate speech is classification in this paper is done using a text classification approach that includes preprocessing methods, feature extraction techniques, and machine learning algorithms.

Both the TFIDF char vectorizer and TFIDF word vectorizer performs very well with both the Logistic Regression algorithm and Ensemble tree model. Although both the models performed well, Ensemble tree had high scores in terms of all metrics. Also, the misclassification of hate tweets in Ensemble tree classifier was low as compared to LR classifier. Hence Ensemble Tree was chosen to be the best performing model for the dataset chosen in this paper.

As can be seen from the findings, the models classified the Non-hate class tweets with a high degree of accuracy. But while detecting hate speech, the model is showing lower performance. This might be because of the absence of the words that strongly exhibit the hate content. In addition to that, the sarcastic comments tend to get classified as Non - hate by the machines. Moreover, the scarcity in the

training data may also be a factor of misclassification. More hate speech training data, which are present without clear mentions of specific racist or homophobic remarks, must be used in order to prevent this type of misclassification in our system and to obtain more accurate classification results.

The ability to identify hate speech is contingent on our subjective perceptions of what constitutes hate speech. As a result, the long-term goal is to better understand social biases and develop a model that can adequately detect them.

V. REFERENCES

- [1] Abro,S., Shaikh,S., Khand,H.Z.,Ali,Z., Khan,S., Mujtaba,G., Automatic Hate Speech Detection using Machine Learning: A Comparative Study. International Journal of Advanced Computer Science and Applications (IJACSA),2020, Volume 11, Issue 8.
- [2] Burnap, P. and M.L. Williams, Us and them: identifying cyber hate on Twitter across multiple protected characteristics. EPJ Data Science, 2016. 5(1): p. 11.
- [3] Cavnar, W.B. and J.M. Trenkle. N-gram-based text categorization. in Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval. 1994. Citeseer.
- [4] Davidson, T., Warmley, D., Macy,M., and Weber, I. Automated hate speech detection and the problem of offensive language. arXiv preprint arXiv:1703.04009, 2017.
- [5] Fortuna, P. and S. Nunes, A survey on automatic detection of hate speech in text. ACM Computing Surveys (CSUR), 2018. 51(4): p. 85.
- [6] Kwok,I.,andWang,Y. 2013. Locate the hate: Detecting tweets against blacks. In AAAI.
- [7] Mujtaba, G., et al., Prediction of cause of death from forensic autopsy reports using text classification techniques: A comparative study. Journal of forensic and legal medicine, 2018. 57: p. 41-50.
- [8] Wang, W.,Chen, L., Thirunarayan, K., and Sheth, A. P. 2014. Cursing in english on twitter. In CSCW, 415-425.
- [9] Warner, W., and Hirschberg, J. 2012. Detecting hate speech on the world wide web. In LSM, 19-26.

- [10] Waseem, Z. and D. Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. in Proceedings of the NAACL student research workshop. 2016.
- [11] Waseem, Z., 2016. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In Proceedings of the First Workshop on NLP and Computational Social Science. Association for Computational Linguistics, Austin, Texas, 138–142.
- [12] Z. Zhang and L. Luo, "Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter," vol. 1, no. 0, pp. 1–5, 2018.
- [13] Zhang, M.-L. and Z.-H. Zhou, A k-nearest neighbor based algorithm for multi-label classification. *GrC*, 2005. 5: p. 718-721.