Hate Speech Detection for Micro-blogging Websites

Submitted By

Shivani Bambhaniya

17MCEC01

DEPARTMENT OF COMPUTER ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Hate Speech Detection for Micro-blogging Websites

Major Project

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering (CSE)

Submitted By

Shivani Bambhaniya

(17MCEC01)

Guided By

Dr. Sanjay Garg

DEPARTMENT OF COMPUTER ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2019

Certificate

This is to certify that the major project entitled **"Hate Speech Detection for Micro-blogging Websites"** submitted by **Shivani Bambhaniya (Roll No: 17MCEC01)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering (Specialization in title case, if applicable) of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-I, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Sanjay Garg
Guide & Professor,
CE / IT Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Priyanka Sharma
Professor,
Coordinator M.Tech - CSE (Specialization)
Institute of Technology,
Nirma University, Ahmedabad

Dr. Madhuri Bhavsar
Professor and Head,
CE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Student Name**, **Roll No.**, give undertaking that the Major Project entitled "**Title of the Project**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering (Specialization in title case, if applicable)** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student
Date:
Place:

Endorsed By
Dr. Sanjay Garg
(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr. Sanjay Garg**,Professor, Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Madhuri Bhavsar**, Hon'ble Head of Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Shivani Bambhaniya

17MCEC01

Abstract

Hate speech detection is a problem of filtering negative textual content on social media. I have limited this problem to content on micro-blogging website Twitter.Input to our problem is collection of tweets by various users. Before applying any classification or clustering approach to solve the problem data needs to be in specific form.For that Regex library of python is used. Another python library NLTK is used. As the data is in text format we have use feature extraction techniques which gave us set of features which can be fed to classification algorithm. The classification algorithm that is used for this problem in this report is Naive Bayes Classification, Support Vector Machine and Random forest algorithm.Accuracy measurement for all the algorithm is done simply in the form of Precision and Recall. Other learning approach like LSTM network,deep learning etc can be used which tend to give higher accuracy than simple classification approach.This leads to Future scope for improving this problem.

Abbreviations

SVM	Support Vector Machine
RE	Regular Expression
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
UG	Uni-Gram
TG	Tri-Gram
POS	Part of Speech
LSTM	Long Short Term Mermory

Contents

List of Figures

Chapter 1

Introduction

1.1 Understanding the Problem

Online Social network has a sub division called microblogging websites for example Twitter, Facebook, Instagram and etc. Such websites are more popular with users from Different background, interest and culture. The reason for the success for such websites is freedom to share thoughts and opinion through number of posts. With such privilege comes the problem of controlling the content on social media such that many people tend to offend others from different background with hateful comments or posts, as the communication here is indirect people tend to show more aggression. This kind of activities on social networks causes hate crime which can put serious effects on businesses and life. Hate crime incident includes bullying, harassment, verbal abuse, damage to property, physical assault etc. According to survey hate crime rose by 40%, from 5,949 in 2016-17 to 8,336 in 2017-18 in UK.

1.2 Finding Solution of Problem

As websites is not directly able to control the online activity of users because of tremendous numbers users and amount of data, we have to find some other way to deal with it. One of them is use data science approaches to do sentiment analysis of text data. The main idea here is to find sentiment polarity of text data so that we can classify text into hate or non-hate statement. Once we are able to recognize whether the data is hateful, it is easy for us to deal with negative content on social media either by removing it or not letting it appear on site at all which is the another area of research called Real Time

Hate Speech Detection. We can find the sentiment polarity of user itself from the previous posts, comments, Pages he follows, Likes on type posts. This can lead us to more effective way of hate speech detection.[1][2]

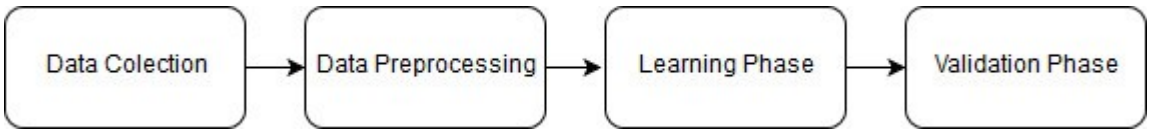

Figure 1.1: Phases to Solve Problem

For this project I am trying to build an algorithm for hate speech detection on twitter data, which are of the form comments on post. Twitter is a website containing more than 336M users from all across the globe. This gives us surety for highly diverse network of user will be there and also the hate speech. For this project I am using dataset which i got from author of paper [3] . This is unprocessed raw data set target with yes = hate, no = non-hate. To increase the accuracy of algorithm, the data pre-processing is done. Tweets are sort sentences having less than 150 characters mostly. This small data contains spelling mistakes , local slang words. Also they do contain Hash-tags, References(Handles), emogies and puntuations.

To train a model for hate speech detection various machine learning approaches are suggested to use. we will compare performance of Nave Bayes, Random Forest, SVM and at the last we will use LSTM. From these four approaches first three are Machine learning model while LSTM is deep learning model.For Validating the Quality of our model we do testing with our test data and calculate accuracy from it for Machine learning models and we calculate Loss for deep learning approach.

Chapter 2

Literature Survey

2.1 Data Pre Processing Techniques

Online text contains noise and uninformative parts, which needs to be removed. Because they causes problem called high dimensionality and makes our classification problem more difficult. For example HTML Tags, Punctuation marks, Repeated Characters, scripts and advertisement etc.The process of dealing with this high dimensionality problem is called Data Pre-processing. Some of the algorithms for this purpose are tokenization, stemming, stop word, removal of punctuation and symbol, word normalization etc.[4]
The flow that I have followed is as per fig:1.2:.

1. Removal of URL and Hashtags: has been trending over social media for past few years. it gives ability people to make their post relatable to some subject so that other people having interest in subject can view their post.removing text stating with and https—http—ptd.

2. Replace tabs and line break with blank space: Extra tabs and line break do not contain any information about the sentiment of the post so we need to remove them. we perform replacing tab and extra spaces with single space.

3. Removing Punctuation:punctuation contains the type of statement. for example where the sentence is question or exclamatory statement. But multiple occurrences does not convey any extra meaning. so we remove them by replacing punctuation with respective tags .

    - !!!! ->EXCL

Figure 2.1: Steps for Pre-processing

- ???? ->QUES

4. Removing repeating Vowel : People tend enter character multiple times to show high intensity of their emotion,to deal with it we perform Replacing word with repeating characters into the original form For example: haaaaapppyyyy ->happy Saaddd ->sad

5. Convert emoticons into tag: Emogi provides very effective medium to express emotions of Users.As each emogi contains specific meaning we need to divide them into SMILEY_POSITIVE and SMILEY_NEGATIVE

SMILEY_POSITIVE	**SMILEY_NEGATIVE**
0:-)	:-(
:)	:(
:D	:(
:*	:(
:o	:"(
:P	:((
;)	

Table 2.1: Positive-Negative Emogies

2.2 Stemming Algorithm

There are basically two types of stemming algorithm trivial one (afix removal) and the static one.The trivial algorithm simply removes the parts of the string after some characters which is not efficient as not all the words will be having same length.
For example Cutting down the last 2 characters of a string will work fine for Greater->Great but fails for Greatest->Greate.
Because of this reasons we have statistical stemming algorithm. For example they replaces the word in following way shown in table.

Removes Suffix like -ly, -able etc.	relatable:relate
Removes -ed, -ing	moving:move

Table 2.2: Stemming Algorithm Working

2.3 Lemmatization

It is a more effective way of performing stemming rather than just replacing a word which is done in static stemming algorithm it will will differentiate between noun and verbs. Then it will only normalize noun.

2.4 Feature based sentiment analysis

Feature based sentiment analysis can be divided into subparts as :

- Feature extraction
- Sentiment prediction
- Sentiment classification

Where feature extraction is a phase in which features are identified .the features can be of three types:

- Semantic Feature: It works on context or semantic orientation of text by adding text at the end of statement.
- Lexico-structural Feature: this includes special symbol frequencies and word distribution throughout the text.

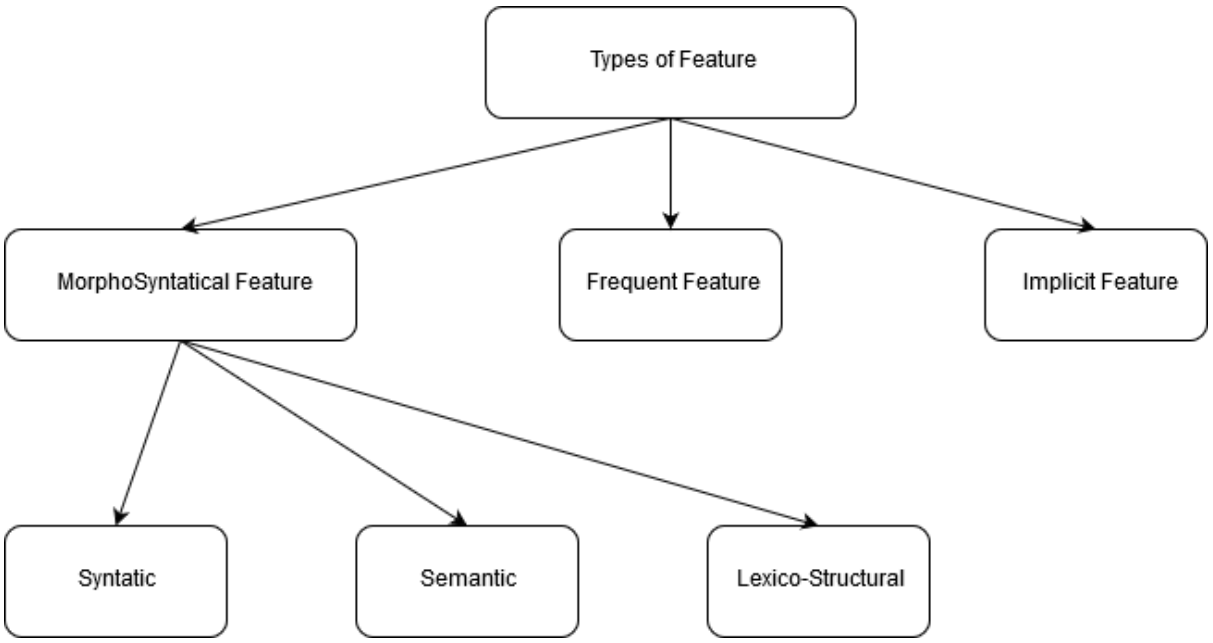

Figure 2.2: Types of Feature

- Syntactic Feature: This includes POS tagging and word N-grams.
- Frequent Feature: this is also called hot Features. This are the feature which people are more interested in or in another way hot topics at that time.
- Implicit feature: This are the indirect meaning of the n-grams in text. for Example: expensive ->More price.

Figure 2.3: Types of Feature

The feature extraction process can be divides into two parts. Lets discuss them in detail.

- Feature selection:
  - NLP based feture uses nouns , pronouns, phrases, adjective, and verbs. They have high accuracy but low recall.
  - Clustering based: this are machine learning based technique so this need previous experience to extract feature.

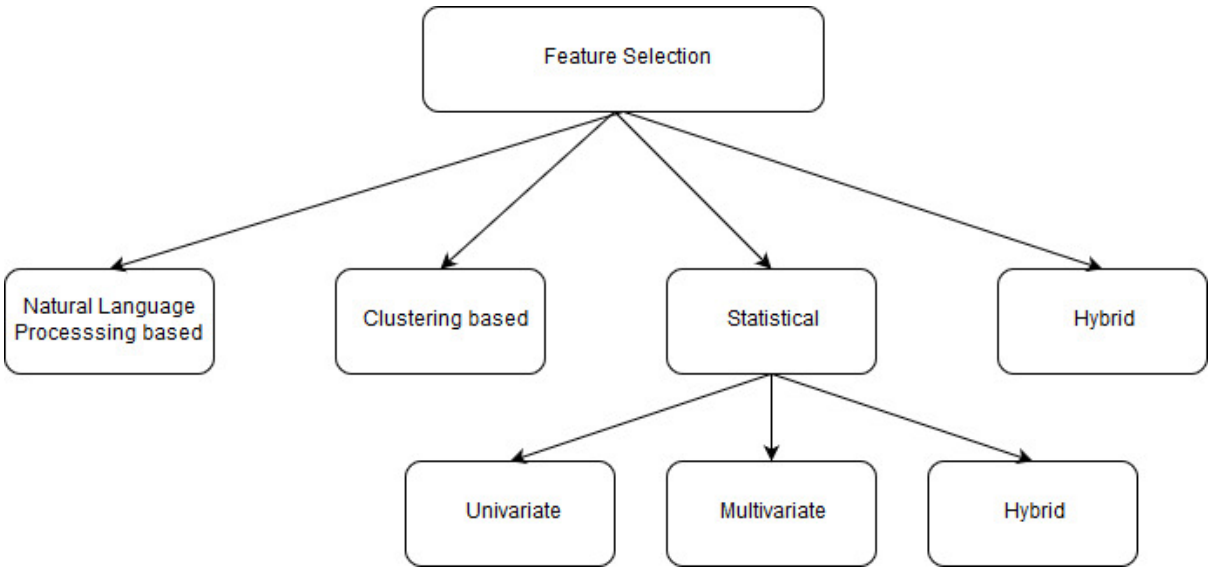

Figure 2.4: Types of Feature Selection Method

  - Univariate Statistical : this is a filtering method which uses information gain, chi-square, occurrence frequency etc as base algorithm.
  - Multivariate Statistical: recursive feature elimination , decision tree these are the approaches for multivariate statistical feature selection.
  - Hybrid Statistical: these method applies both univariate and multivariate approach to get high accuracy.

- Feature cleansing: while generating a feature set there are multiple non useful features are created to remove those feature cleansing is done. In this phase basically features are combined with their identical features then all the useful feature are being marked and rest unmarked features are simply deleted.[5]

2.5 Algorithms for Building Model

Here you can see in below figure that Classifier initially is divide into 2 type first is Machine learning approaches and the other is Natural language processing. Machine learning

r if we derive features which are highly helpful in classifying hate or no-hate polarity of the text, the challenge here is we need to have UsWorkent s befeatextraction method which can help with this problem. For example, Bag outrords. Bag of word for the hate/non-hate sentiment polarity can be created with splitting bag of word for happy/unhappy

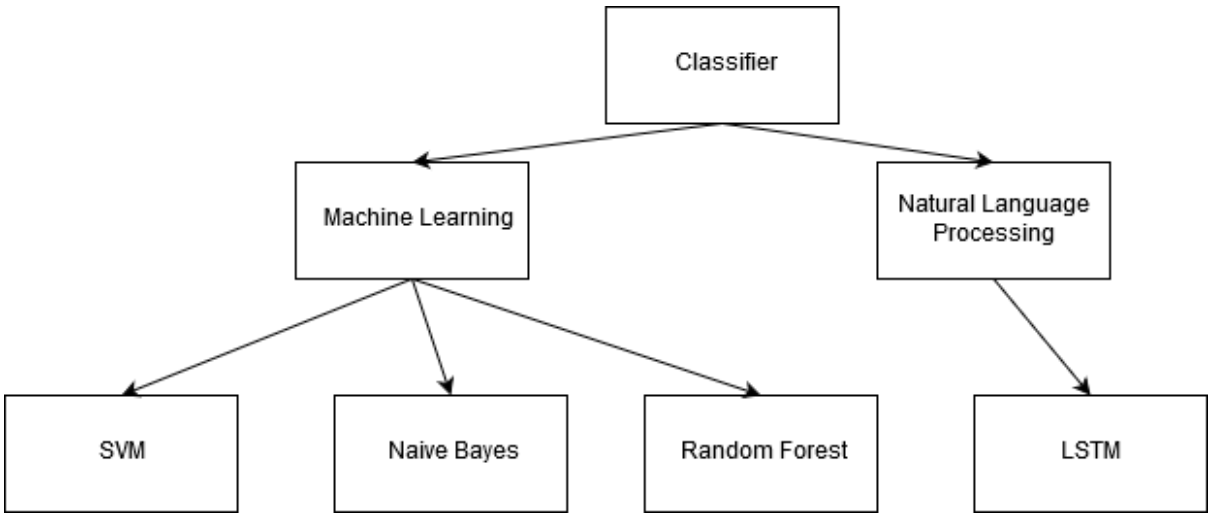

Figure 2.5: Approaches for Classification Problem

,sad, anger and other emotions.

While on the Other hand we have another efficient approach to follow which falls under natural language processing techniques.Human speaks not even multiple languages but also has grammar rules and slang. For textual data one more constraint is added as human doesn't use complete sentences but abbreviations, they tend to mix up slang of other languages. With these characteristic of data NLP Tends to work effectively some of the areas where NLP is Used mostly are,

- Speech to text and text to speech conversion
- content summarizing
- document classification etc.

Now let's understand how these approaches work In detail one by one.

1. Support Vector machine: SVM gives good performance because of its ability to handle high featured and sparse data.Out input data is sparse as it can contain feature count for many emotions, N-grams etc.

   SVM allows us to classify data using hyper plane .this hyper plane can be of the form linear , radious etc.

   SVM requires the data to be in a specific format . which is numeric representation of textual data as attribute for example no. of positive as well as negative words , smiley count etc.

From SVM example figure you can see that Data can be sparse in many ways and many dimentions, by dimention we mean number of features. For now let's take example of SVM with only 2 features/dimetions. For figure on the left hand side you can see that data is linearly separable, so for this case the hyperplane which clssifies all the data point is linear equation of the type

$$Z = PX + QY$$

but for the right figure this is not the case, there is no way you can Create a linear hyper plane which divides the data point with optimal loss rate. so for this case the hyperplane equation is like

$$Z^2 = X^2 + Y^2$$

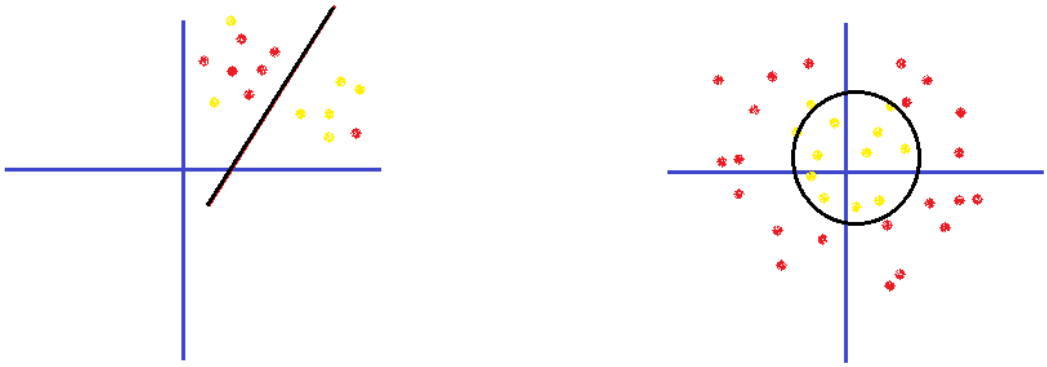

Figure 2.6: SVM Example

.

2. Naive bayes classifier:[6] and the partial timeline method can be used to implement this model. the nave bayes classifier model conducts three calculations processes, likelihood and probability.

   Processes are occurrence level of label on previous data. Likelihood calculation is possibility that word appears on particular label Probability is calculation of measuring occurrence on sentence.[7] The probability is calculate as below, where

label is one of the target classes and features needs to be a numeric value.

$$P(label|Features) = \frac{P(Features|Label) * P(label)}{P(Features)}$$

For our case the features contains more than 1 feature so, the calculation will be as below.

$$if Features = F1, F2, ..., Fn$$

$$P(Label|Features) = P(F1|Label) * P(F2|Label) * ... * P(Fn|Label) * P(Label)$$

lets have small look at how naive bayes work on data, naive bayes creates a distribution fuction which as is shown in figure. Suppouse we need to find that data point z belongs to which class. naive bayes will create probability distribution function with maximum probability at point x , that whether the data point will be classified as class 1 or not.

So, we can understand it in a way that all the data point simiar to x have high probability that they belongs class 1.

So, for each class probability is calculated and the class with highest probability at data point z is declared as target class.

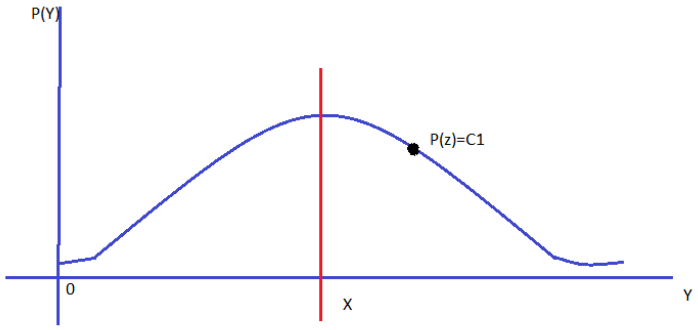

Figure 2.7: Naive-Bayes Example

3. Random Forest Algorithm: This algorithm is useful for both regression as well as

classification problems. random forest is most useful when number of features are relatively more and we can not decide which are important features for this. So, without Principle component analysis and Dimensionality reduction we can have efficient results with Random forest.Lets understand how random forest works and take figure 2.8 for you reference.

(a) Large Training Dataset is dvided into N Parts, where N is constant and given as input to random forest algorithm to tell number of maximum decision trees to be created. but take a note that,

$$N < Numberof Features$$

(b) In the Next step N number of Decision trees are created with similarly As it is created separately for each data subset.

Forest.png

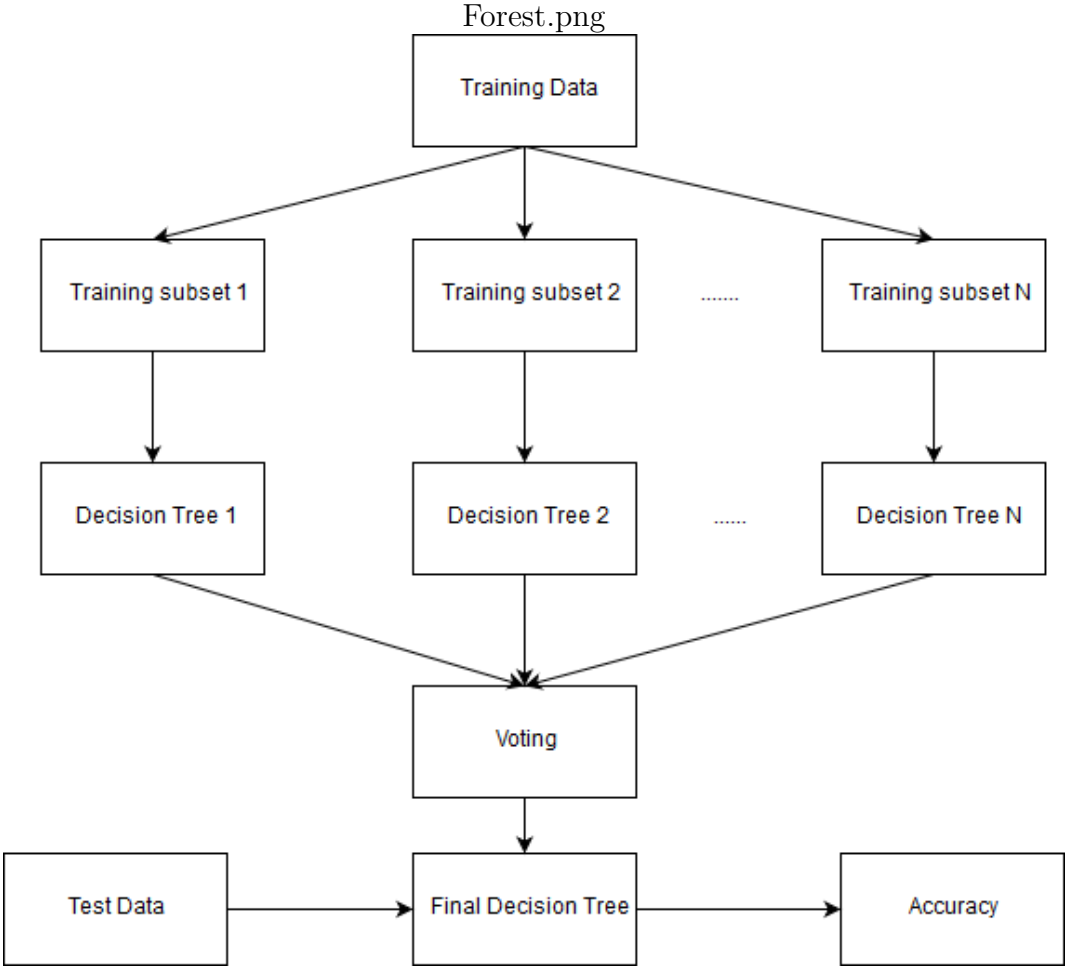

Figure 2.8: Random Forests Example

(c) From This N decision tree now its time to choose the best one this happens via random forest's voting mechanism.

(d) Its time to check the accuracy check for the chosen decision tree.

Till Now we are Done understanding all three of Machine learning approaches now its turn to understand Natural Language Processing technique. which is Long Short Term Memory.Two highly recommended from markable researchers in this area are Recurrent Neural Network and LSTM [Long Short Term Memory] Tweet can not be classied as adding up accuracy of tokens in sentence as all of them are highly dependent on each other. Above two algorithms helps in calculating dependencies among the tokens as they support backward analysis.

So, from this two LSTM is Recommended algorithm as RNN contains problem of

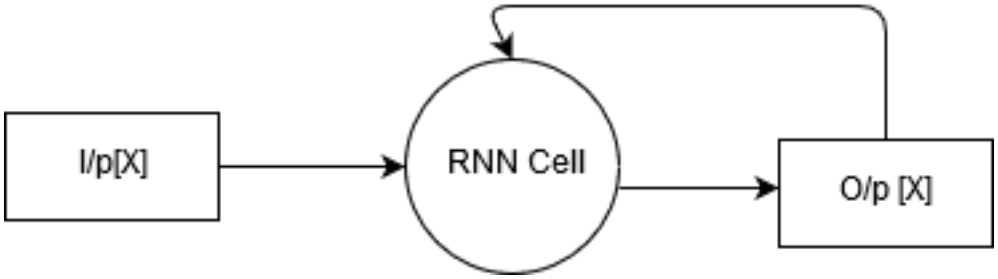

Figure 2.9: RNN Working Diagram

Vanishing Gradient Problem, which means For the high dimention data the effect of One important feature gets nullied which tends to wrong prediction in future. On the other hand LSTM Reduces the chanes of this as it contains Various Gates which Stores Important information as below:

- Input Modulation Gate - Removes Redundant Features From Input.
- Forgot Gate - Identies which feature to remember from previous trainings.
- Output Gate- Modulates the Cell Output to come to Final Output.

In Real World data we can not predict its content to be hate or non-hate just by reading it in forward diection for example,

- he said ,Teddy Bears are on sale!
- he said ,Teddy is worst president :(

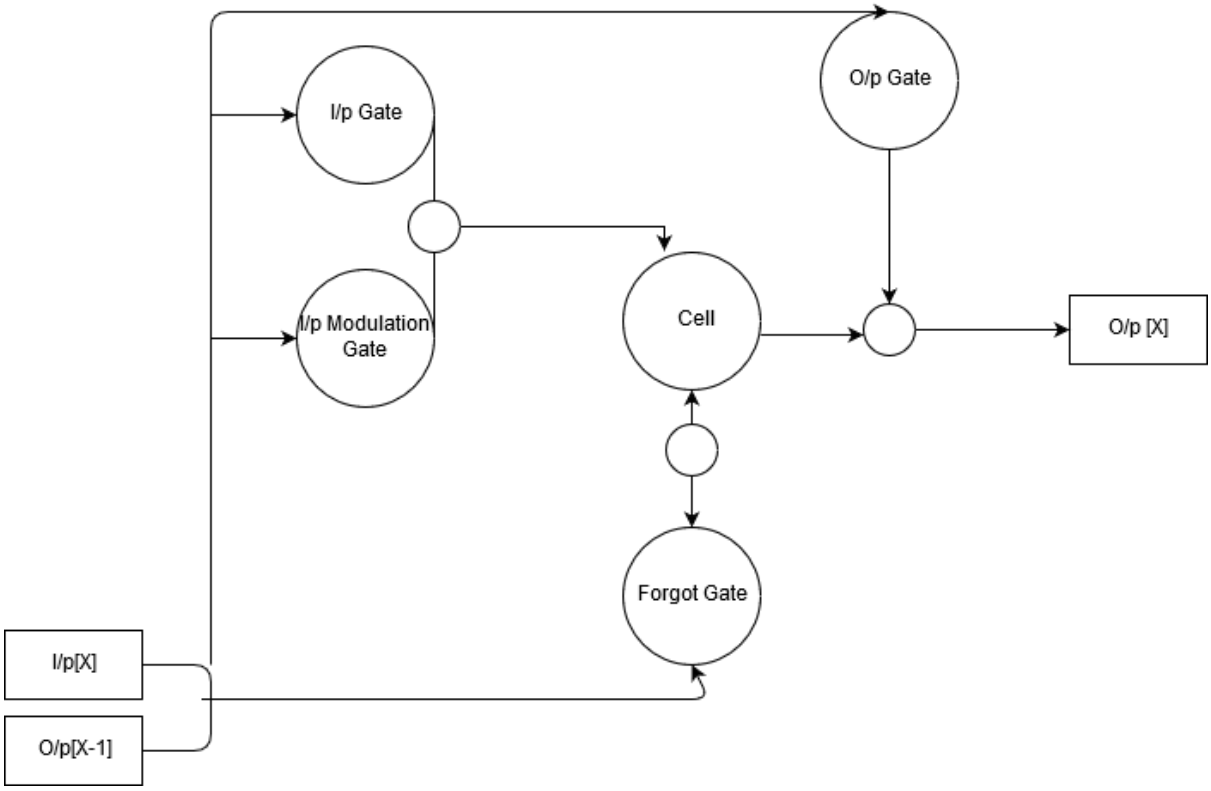

Figure 2.10: LSTM Working Diagram

Here after the word teddy we can not predict the tweet to be positive or negative, For this reason i have used Bi-directional LSTM. LSTM output is followed many layers each layer having specic funtionality. Below is diagram for Used Layers.

- Concat Layer - It concatinates o/p weight matrix derived from both forward and backward LSTM.
- AttentionLayer-AttentionLayerProvides Weights for input data from previous layer to tell on which portion to pay attention to. Attention layer can be implemented using activation Funtion, which is function that is used to map O/p of one layer to i/p to another layer. I have used softmax activation from keras library.
- Post Attention Layer - This is the Final layer which predicts the label for tweet. Stochastic Gradient Descent is used as optimizer to make the process faster for larger data set and Finally we add to this layer cross-entropy loss function to evalute quality of model.

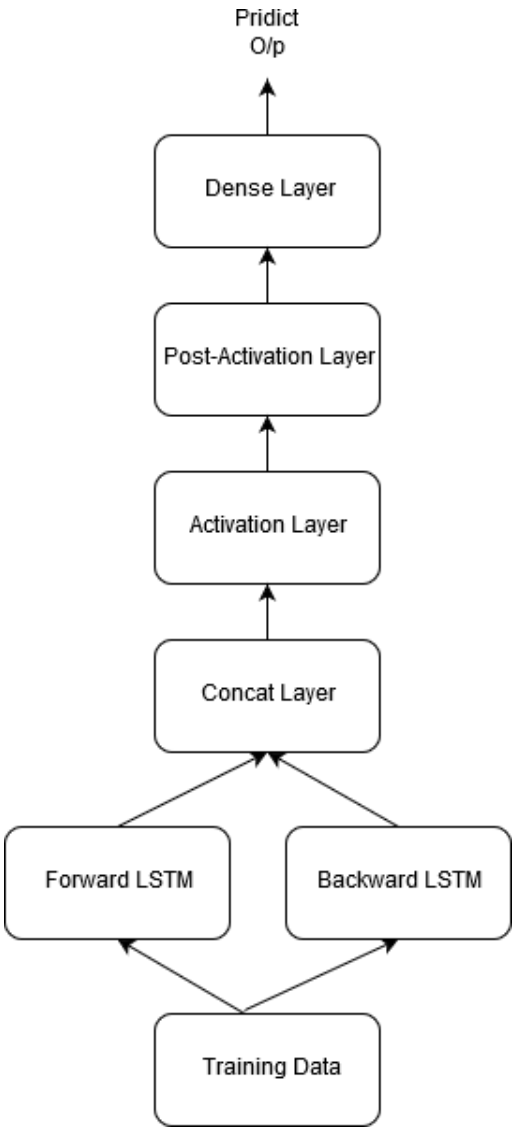

Figure 2.11: LSTM Layer Structure

Chapter 3

Methodology

This chapter contains information about prerequisites for Creating a Hate Speech Detection model.Which Contains Planning and Data Collection phase.

3.1 Dataset

3.1.1 Dataset Requirement

For this problem we needed data containing two main attributes Text and sentiment. As our target application is Twitter the data must have been of the form such that it contains URL, Handles, and Hash-tags and length of the text should be around 150 characters. Twitter's API for developer allows to download tweets Using twitters SearchAPI and RestAPI. As DataSet is already available English Text i am going to use that only.

3.1.2 Dataset Description

This is a raw twitter comment dataset from twitter. This dataset contains around 6K tweets extracted using the twitter API.It contains the following 2 fields: Figure 3.1 and 3.2 gives us information about how our dataset looks like as well as ratio of positive to negative tweets.

Field	Example
Target: the polarity of the tweet	Yes = Hate, No = Non-Hate
Text	The text of the tweet A happy life is this..

	ï»¿Text	Tweet_id	Target
0	Knowing ki Vikas kitna samjhata hai Priyanka a...	1	no
1	I am Muhajir .. Aur mere lye sab se Pehly Paki...	2	no
2	Doctor sab sahi me ke PhD (in hate politics) ...	3	no
3	Poore Desh me Patel OBC me aate Hain sirf gujr...	4	no
4	Sarkar banne ke bad Hindu hit me ek bhi faisla...	5	yes

Figure 3.1

```
In [5]: runfile('C:/Users/Shivani Bambhaniya/preprocessing/LSTM.py', wdir='C:/
Users/Shivani Bambhaniya/preprocessing')
Dataset Size:  4579
Hate Tweet Dataset Size:  1661
Non-Hate Tweet Dataset Size:  2914
```

Figure 3.2: Graphical Representation of Data

3.2 WorkPlan

Steps:

1. Collection of data: This phase includes creating or downloading a datasets which matches our requirement.For our case the requirement was the tweet text portion should contain handles, hash-tags and URL.I was able to finding dataset with matching characteristic from Kaggle Dataset.

2. Pre-processing the dataset: Pre-processing is very essential task for creating a model for hate speech detection, as it allow us to remove all the unnecessary features from tweet text like handles,URLs,Hash-tag, unnormalized Verbs etc. This phase reduces the dimentionality of our feature set.

3. Divide the Dataset into Training and Testing Data: Choosing a correct proportion for the division of training and testing data is very essential. As it affects the accuracy of our model. Very small training set or very small testing set can not derive a good quality model.

4. Extract and store separately features from Traning and Testing data: As set of feature is given as input to classification algorithm ,separate feature set is created for training and testing data. Separate feature set is created because features are strictly dependent on dataset.

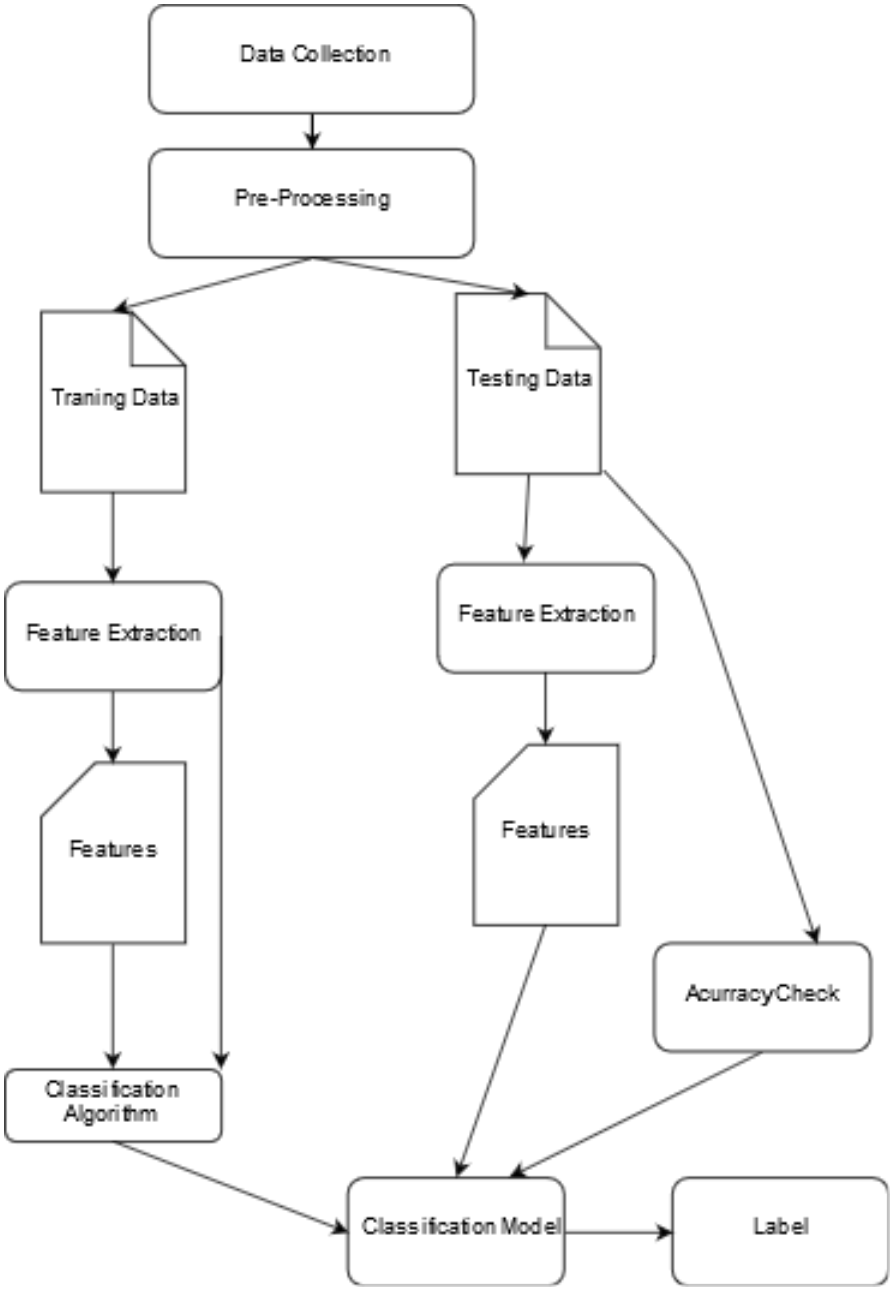

Figure 3.3: Work Flow for Implementation

5. Run classification algorithm having traning data and traning data feature set as input: We have training feature set input ready from previous step, we can give as a input to classification model. Uptill now SVM, Naive-bayes classifier and random forest is used as classification approach. we will discuss them in later section.

6. Classification model is ready: The trained classification model will be ready after this step.

7. Run accuracy check function with input as test data, test data feature set and

classification model: to check the quality of the model testing data and extracted feature set from testing data is given as input to classification model.

8. Output labels: Model predicts the target for tweet text and gives us the label set for each tweet.

9. Measure accuracy: With output labels vs label in dataset we can measure the accuracy of model. The equation for that is as below.

$$Accuracy = \frac{TruePositive + FalseNegative}{Total} * 100$$

Chapter 4

Implementation

4.1 Implementation

As per our work flow chart first step is data collection. Which is already done and information is in section 3.1.Now let move toward next step which is data Preprocessing.

4.1.1 Data Preprocessing Implementation

As we have discussed in section 2.1 preprocessing is done in multiple steps lets follow the fig:2.1 and proceed sequentially.I have used Python's re REGEX library for this purpose.in below table i have listed regular expression to find out specific patterns in text. To remove Negation we have simply removed them with RE with or condition. below is

Pattern	RegularExpression
URL	"(http—https—ftp)://[a_zA_Z0_9\./]+"
Hashtag	"(#(+))"
Handles—Reference	"(@(+))"
Repeating	"(.)1,"
Stop Words	"(of—the—The—a—an—A—An—For—To—to—at—in—by—BY—In—At)"

List of negation words:

(never—no—nothing—nowhere—noone—none—not—havent—hasnt—hadnt—cant—couldnt—shouldn

```
308
309 def PreProcessing(tweet):
310     tokenised_tweet = TokeniseTweet(tweet)
311     happy, sad, anger, fear, surprise, disgust = GetEmoticons(tokenised_tweet)
312     hashtags = GetHashTags(tokenised_tweet)
313     intensifiers = GetIntensifiers(tokenised_tweet)
314     negations = FindAllNegations(tokenised_tweet)
315     usernames = GetUserNames(tokenised_tweet)
316     urls = GetURLs(tweet)
317     upper_case_words = FindUpperCaseWords(tweet)
318     processed_tweet = ProcessTweet(tokenised_tweet)
319     delimiter = ' '
320     processed_tweet = delimiter.join(processed_tweet)
321     punctuations_marks_count = GetPunctuationMarks(processed_tweet)
322     processed_tweet = RemovePunctuations(processed_tweet)
323     processed_tweet_tokenised = TokeniseTweet(processed_tweet)
324     processed_tweet = delimiter.join(processed_tweet_tokenised)
325     repetitive_words = CheckRepetitions(processed_tweet_tokenised)
326     char_n_grams = CharNGrams(processed_tweet,3)
327     word_n_grams = WordNGrams(processed_tweet_tokenised, 3)
328     word_count = CountOfWordsInATweet(processed_tweet_tokenised)
329     char_count = CountOfCharsInATweet(processed_tweet_tokenised)
330     laugh_words = FindInternetLaughs(processed_tweet_tokenised)
331     word_length_mean = char_count/word_count
332     print( happy, sad, anger, fear, surprise, disgust, hashtags, usernames,urls, pu
333     return happy, sad, anger, fear, surprise, disgust, hashtags, usernames, urls, p
334
335 tweet = "Heyyyyy I am verrryyy happppyyyy #U @U http://google.com LOLLL"
336 word = PreProcessing(tweet)
```

Figure 4.1: Complete code for Preprocessing

You can see from above image that that we have applied all preprocessing techniques in for loop and that for loop will iterate for each tweet.The below image shows the output for the given code.You can see that each origional sentencr is preprocessed and the (count of them) features are extracted then given as input to SVM, Random Forest and Naive bayes..

4.2 Feature Extraction implementation

For the Feature Extraction purpose i have use Python NLTK libtrary, which lets us create N-gram.we have used UniGram, biGram and triGram lets discuss about all three of them.

1. UniGram: Uni-gram is a sequence of word with only one word in it. as uniGram has only one word there they tend to give higher performance than any other.

2. BiGram: BiGram is squence of 2 Word where give the context of a sentence and first word you can predict the next word in sentence.

3. Tri-gram: Tri-gram is sequence of three word where given first two words and context of sentence you can predict the third word.

```
In [12]: runfile('C:/Users/Shivani Bambhaniya/preprocessing/preprocessing.py',
wdir='C:/Users/Shivani Bambhaniya/preprocessing')
[] [] [] [] [] [] ['#u'] ['@U'] ['http://google.com'] {}
['Heyyyyy¿I¿am¿verrryyy¿happppyyyy¿HASHTAG¿USER¿URL¿LOLLL'] ['H', 'e', 'y', 'y',
'y', 'y', 'y', '¿', 'I', '¿', 'a', 'm', '¿', 'v', 'e', 'r', 'r', 'r', 'y', 'y',
'y', '¿', 'h', 'a', 'p', 'p', 'p', 'p', 'y', 'y', 'y', 'y', '¿', 'H', 'A', 'S',
'H', 'T', 'A', 'G', '¿', 'U', 'S', 'E', 'R', '¿', 'U', 'R', 'L', '¿', 'L', 'O',
'L', 'L', 'L', 'He', 'ey', 'yy', 'yy', 'yy', 'yy', 'y¿', '¿I', 'I¿', '¿a', 'am',
'm¿', '¿v', 've', 'er', 'rr', 'rr', 'ry', 'yy', 'yy', 'y¿', '¿h', 'ha', 'ap',
'pp', 'pp', 'pp', 'py', 'yy', 'yy', 'yy', 'y¿', '¿H', 'HA', 'AS', 'SH', 'HT',
'TA', 'AG', 'G¿', '¿U', 'US', 'SE', 'ER', 'R¿', '¿U', 'UR', 'RL', 'L¿', '¿L',
'LO', 'OL', 'LL', 'LL', 'Hey', 'eyy', 'yyy', 'yyy', 'yyy', 'yy¿', 'y¿I', '¿I¿',
'I¿a', '¿am', 'am¿', 'm¿v', '¿ve', 'ver', 'err', 'rrr', 'rry', 'ryy', 'yyy',
'yy¿', 'y¿h', '¿ha', 'hap', 'app', 'ppp', 'ppp', 'ppy', 'pyy', 'yyy', 'yyy',
'yy¿', 'y¿H', '¿HA', 'HAS', 'ASH', 'SHT', 'HTA', 'TAG', 'AG¿', 'G¿U', '¿US',
'USE', 'SER', 'ER¿', 'R¿U', '¿UR', 'URL', 'RL¿', 'L¿L', '¿LO', 'LOL', 'OLL',
'LLL'] ['Heyy¿I¿am¿verryy¿happyy¿HASHTAG¿USER¿URL¿LOLL'] ['LOLLL'] [] [] 1 45 45.0
[]
```

Figure 4.2: Output Preprocessing

always number of UniGram<BiGram<TriGram.let's now look at the implementation part of feature extraction process.

```
def BuildFeatureVectorForTweet(tweet):

    #print "BuildFeatureVectorForTweet Called"
    global char_n_grams_index, word_n_grams_index
    happy, sad, anger, fear, surprise, disgust, hashtags, usernames, \
    urls, punctuations_marks_count, repetitive_words, char_n_grams, \
    word_n_grams, upper_case_words, intensifiers, negations, \
    word_count, char_count, word_length_mean, laugh_words = pre.PreProcessing(tweet)

    feature_vector = []
    #print char_n_grams_index
    #print word_n_grams_index
    feature_vector = AddEmoticonFeatures(feature_vector, happy, sad, disgust, anger,
    #feature_vector = AddCharNGramFeatures(feature_vector, char_n_grams_index, char_
    #feature_vector = AddWordNGramFeatures(feature_vector, word_n_grams_index, word_
    feature_vector = AddRepetitiveWordsFeature(feature_vector, repetitive_words)
    feature_vector = AddPunctuationMarksFeature(feature_vector, punctuations_marks_c
    feature_vector = AddUpperCaseWordsFeature(feature_vector, upper_case_words)
    feature_vector = AddWordCountFeature(feature_vector, word_count)
    feature_vector = AddCharCountFeature(feature_vector, char_count)
    feature_vector = AddAverageWordLengthFeature(feature_vector, word_length_mean)
    feature_vector = AddLaughWordsCount(feature_vector, laugh_words)
    feature_vector = AddIntensifersFeature(feature_vector, intensifiers)
    feature_vector = AddNegationsFeature(feature_vector, negations)
    return feature_vector
```

Figure 4.3: Feature Extraction: Uni,Bi,Tri Grams

```
fv:  [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 45, 45.0, 0, 0, 0, 0, 0]
```

Figure 4.4: Feature Extraction output

As you can see that for sample data file Extract Feature function has extracted following features and (1) indicates one occurrence of that specific feature.

4.2.1 Naive Bayes Classifier Implementation

With use of Python's NLTK library only i have Trained model with naive bayes classifier.Here Below is code snippet for classification.

```
K=10;
k=5;
isObj   = lambda sent: sent in ['0','4']
makeObj = lambda sent: 'obj' if isObj(sent) else sent

train_tweets = [x for i,x in enumerate(Tweets) if i % K !=k]
test_tweets = [x for i,x in enumerate(Tweets) if i % K==k]

train_tweets=[ (words, makeObj(sent)) for (words, sent) in train_tweets ]
v_train = nltk.classify.apply_features(features,train_tweets)
v_test  = nltk.classify.apply_features(features,test_tweets)
classifier = nltk.NaiveBayesClassifier.train(v_train)
classifier.show_most_informative_features()
accuracy= nltk.classify.accuracy(classifier, v_test)
```

Figure 4.5: Code Snippet NaiveBayes Classification

4.2.2 SVM Classifier Implementation

The Code which i have written for naive bayes classification is written for SVM classifier.The only change is replacing the classifiew as below:

$$classifier = nltk.classify.SvmClassifier.train(v\_train)$$

4.2.3 Random Forest Classifier Implementation

The working of random forest is divided into 2 part: 1.Creatinf a decision treee for each sub traning set 2. choosing the best among them.The below given graph shows the working:

To implement random forest classifier in python i have used sklearn library, which provides a base model for creating a random forest classifier. Below given is code snippet for that.

```
K=10;
k=5;
isObj    = lambda sent: sent in ['0','4']
makeObj = lambda sent: 'obj' if isObj(sent) else sent
train_tweets = [x for i,x in enumerate(Tweets) if i % K !=k]
test_tweets = [x for i,x in enumerate(Tweets) if i % K==k]

train_pred=train_tweets[:-1];
test_pred=test_tweets[:-1];
print(train_pred);
RFClassifier=RandomForestClassifier(n_estimators=100)
RFClassifier.fit(train_tweets,)
test_label=RFClassifier.predict(test_tweets)
tree = RFClassifier.estimators_[5]
export_graphviz(tree, out_file = 'dt.dot', feature_names = features, rounded = Tr
(graph, ) = pydot.graph_from_dot_file('dt.dot')
graph.write_png('dt.png')
```

Figure 4.6: Random Forest Classification Algorithm Implementation

4.2.4 LSTM Classifier Implementation

Impementation of lstm can be divide into many parts as per figure 2.11 LSTM layer Scturcture. You can see from Figure 4.7 that Intially we have create two different model left which if forward and right which is backward, then we have combined them both using model.add function.

```
left = Sequential()
right = Sequential()
model = Sequential()


#Add second Bidirectional LSTM layer

left.add(LSTM(output_dim=hidden_units, init='uniform', inner_init='uniform',
               forget_bias_init='one', return_sequences=True, activation='tanh',
               inner_activation='sigmoid',input_shape=(128,1)))

right.add(LSTM(output_dim=hidden_units, init='uniform', inner_init='uniform',
               forget_bias_init='one', return_sequences=True, activation='tanh',
               inner_activation='sigmoid',  go_backwards=True,input_shape=(128,1)))

concat = Concatenate([left, right])
model.add(concat)
#Rest of the stuff as it is

model.add(Embedding(1000, 128, input_length=128))
model.add(Bidirectional(LSTM(64)))
```

Figure 4.7: LSTM Implementation

Chapter 5

Results

The next step is comparing result of all above algorithm. We have seen Two approaches for solution of this problem Naive Bayes classification and SVM classification techniques.The Performance Matrix which we are going to use for machine learning approaches is Accuracy while for NLP approach LSTM i have used Loss as performance matrix.You can See Below two formula lets understand it.

$$ACCURACY = \frac{FalseNegative * TruePositive * 100}{TotalData}$$

where, False negative= Number Of Correct non-hate prediction.
True positive =Number Of Correct hate prediction.

$$LOSS = \sum_{1}^{N} Yi * Log(Yi) - Yi * Log(1 - Yi)$$

Where, N= Number of target.

Model	Accuracy
Naive Bayes	73%
SVM	75 %
Random Forest	72%

Now Lets understand how model.evaluate function works, The input parameters for this function a are as below:

- Text Input Array
- Target Array
- Batch Size [Sample Size to For each evaluation step]
- verbose
- weights indicating importance of each sample set
- Nuumber of steps for this evaluation process
- callbacks to otherprocess this evaluation process

Model.evaulate function returns o/p 0.3159622715427771 this value indicates the loss done with this model.

5.1 Conclusion And Future Scope

Naive Bayes usually tends to work better than SVM Because of relevant Dimentionality in feature scope for SVM.As you can see from table in previous section that SVM work more efficiently than other two approaches.Algorithm like Random Forest and LSTM network are reported as efficient algorithms. As for India Hindi is the usually mixed in Tweets, Classification of those tweets needs hindi sentiment classification technique.[8][9][10][11]

If we talk about improvement we cant see more room for SVM, while for LSTM we have some future ideas. LSTM can work much better if we input it with clean data, by clean data we meant rather than giving long sentences with spelling and grammatical mistake we can pre-process it as what user meant same as how Googles search engine nds out what user meant.

5.2 Publication

[1] Shivani Bambhaniya , Dr.Sanjay Garg *Hate Speech Detection For Social Networking Site.* In TENCON 2019, Premier international technical conference of IEEE Kerala Section (Communicated)

Bibliography

[1] M. I. F. Ika Alfina, Rio Mulia and Y. Ekanata, "ate speech detection in the indonesian language: A dataset and preliminary study,"

[2] T. F. P. F. E. I. F. M. Giulio Angiani, Laura Ferrari and S. Manicardi, "A comparison between preprocessing techniques for sentiment analysis in twitter,"

[3] V. S. S. S. A. Deepanshu Vijay*, Aditya Bohra* and M. Shrivastava, "A dataset for detecting irony in hindi-english code-mixed social media text,"

[4] X. EmmaHaddia and YongShib, "The role of text pre-processing in sentiment analysis techniques for sentiment analysis in twitter,"

[5] S. A. Muhammad Zubair Asghar, Aurangzeb Khan and F. M. Kundi, "A review of feature extraction in sentiment analysis,"

[6] P. S. Naufal Riza Fatahillah and C. Haryawan, "Implementation of naive bayes classifier algorithm on social media (twitter) to the teaching of indonesian hate speech,"

[7] N. Tarasova, "Classification of hate tweets and their reasons using svm,"

[8] D. R. ZiqiZhang and J. Tepper, "Hatespeechdetectionusingaconvolution-lstmbaseddeep neuralnetwork,"

[9] D. R. Ziqi Zhang1 and J. Tepper2, "Detecting hate speech on twitter using a convolution-gru based deep neural network,"

[10] C. StevenZimmerman and UdoKruschwitz, "Improvinghatespeechdetectionwith-deeplearningensembles,"

[11] T. Pratama and A. Purwarianti, "Topic classification and clustering on indonesian complaint tweets for bandung government using supervised and unsupervised learning,"