DETECTION OF HATE SPEECH AND OFFENSIVE LANGUAGE IN TWITTER USING SENTIMENT ANALYSIS

A PROJECT REPORT

Submitted by

THIRUNAYAN MANIKANTAN. R	**113216104150**
ARAVIND KUMAR. D	**113216104016**
HARISH KRISHNA . K	**113216104045**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VELAMMAL ENGINEERING COLLEGE ANNA UNIVERSITY: CHENNAI 600066 APRIL 2020

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DETECTION OF HATE SPEECH AND OFFENSIVE LANGUAGE IN TWITTER USING SENTIMENT ANALYSIS"** is the bonafide work of "**THIRUNAYAN MANIKANTAN. R**(113216104150), **ARAVIND KUMAR. D**(113216104016), **HARISH KRISHNA. K**(113216104045)" who carried out the project work under my supervision.

SIGNATURE

DR. S. CHAKRAVARTHY
HEAD OF THE DEPARTMENT
Professor
Computer Science and Engineering
Velammal Engineering College
Velammal Nagar
Ambattur-Red Hills Road
Chennai-600 066

SIGNATURE

Ms. R. AMIRTHAVALLI
SUPERVISOR
Assistant Professor
Computer Science and Engineering
Velammal Engineering College
Velammal Nagar
Ambattur-Red Hills Road
Chennai-600 066

ANNA UNIVERSITY: CHENNAI 600 025

VIVA VOCE EXAMINATION

The Viva Voce Examination of this project work "**DETECTION OF HATE SPEECH AND OFFENSIVE LANGUAGE IN TWITTER USING SENTIMENT ANALYSYS"** is a bonafide record of project done at the **Department of Computer Science and Engineering**, Velammal Engineering College during the academic year **2019-20** by

THIRUNAYAN MANIKANTAN. R	**113216104150**
ARAVIND KUMAR. D	**113216104016**
HARISH KRISHNA. K	**113216104045**

of final year Bachelor of Engineering in Computer Science and Engineering submitted for the university examination on **22.09.2020**

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I wish to acknowledge with thanks to the significant contribution given by the management of our college **Chairman, Dr. M.V. Muthuramalingam,** and our **Chief Executive Officer Thiru. M.V.M. Velmurugan,** for their extensive support.

I would like to thank **Dr. N. Duraipandian, Principal** of Velammal Engineering College**,** for giving me this opportunity to do this project.

I wish to express my gratitude to our effective **Head of the Department, Dr. S. Chakravarthy,** for her moral support and for his valuable innovative suggestions, constructive interaction, constant encouragement and unending help that have enabled me to complete the project.

I wish to express my indebted humble thanks to our **Project Coordinators, Dr. R. Ramya Devi, Ms. M. Vijayalakshmi and Dr. S.L. Jayalakshmi,** Department of Computer Science and Engineering for their invaluable guidance in shaping of this project.

I wish to express my sincere gratitude to my **Internal Guide, Ms. R. Amirthavalli, Assistant Professor,** Department of Computer Science and Engineering for her guidance without her/his this project would not have been possible.

I am grateful to the entire staff members of Computer Science and Engineering for providing the necessary facilities and to carry out the project. I would especially like to thank my parents for providing me with the unique opportunity to work and for their encouragement and support at all levels. Finally, my heartfelt thanks to **the Almighty** for guiding me throughout the life.

ABSTRACT

The exponential growth of social media such as Twitter and community forums has revolutionised communication and content publishing, but is also increasingly exploited for the propagation of hate speech and the organisation of hate-based activities. The anonymity and mobility afforded by such media has made the breeding and spread of hate speech – eventually leading to hate crime – effortless in a virtual land scape beyond the realms of traditional law enforcement. Existing methods in the detection of hate speech primarily cast the problem as a supervised document classification task [33]. These can be divided into two categories: one relies on manual feature engineering that are then consumed by algorithms such as SVM, Naive Bayes, and Logistic Regression [3, 9, 11, 15, 19, 23, 35–39] (classic methods); the other represents the more recent deep learning paradigm that employs neural networks to automatically learn multi-layers of abstract features from raw data [13, 26, 30, 34] (deep learning methods). In this method We show that it is a much more challenging task, as our 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. We then propose Deep Neural Network structures serving as feature extractors that are particularly effective for capturing the semantics of hate speech. Our methods are evaluated on the largest collection of hate speech datasets based on Twitter, and are shown to be able to outperform state of the art by up to 6 percentage points in macro-average F1, or 9 percentage points in the more challenging case of identifying hateful content. As a proxy to quantify and compare the linguistic characteristics of hate and non-hate Tweets, we also propose to study the 'uniqueness' of the vocabulary for each class.

TABLE OF CONTENTS

LIST OF ABBREVATIONS

URL	Uniform Resource Locator
SVM	Support Vector Machine
OS	Operating System
SQL	Structured Query Language
OOP	Object Oriented Programing
RDBMS	Relational Database Management System
TF	Term Frequency
IDF	Inverse Document Frequency
GUI	Graphical User Interface
LAMP	Linux, Apache, MYSQL, Perl/Python/PHP
UML	Unified Moduling Language
DFD	Data Flow Diagram
CNN	Convolutional Neural Network

LIST OF FUGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 SCOPE OF THE PROJECT

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. we address the problem of sentiment classification on twitter dataset. We use a number of machine learning and deep learning methods to perform sentiment analysis. In the end, we use a majority vote ensemble method with 5 of our best models to achieve the classification.

1.2 NEED OF THE PROJECT

In this work, we argue for a focus on the latter problem for practical reasons. We show that it is a much more challenging task, as our 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. We then propose Deep Neural Network structures serving as feature extractors that are particularly effective for capturing the semantics of hate speech. Our methods are evaluated the largest collection of hate speech datasets based on Twitter, and are shown to be able to outperform state of the art by up to 6 percentage points in macro-average F1, or 9 percentage points in the more challenging case of identifying hateful content.

1.3 MACHINE LEARNING

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

Machine learning combines data with statistical tools to predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of task like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

1.3.1 Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain

Machine Learning

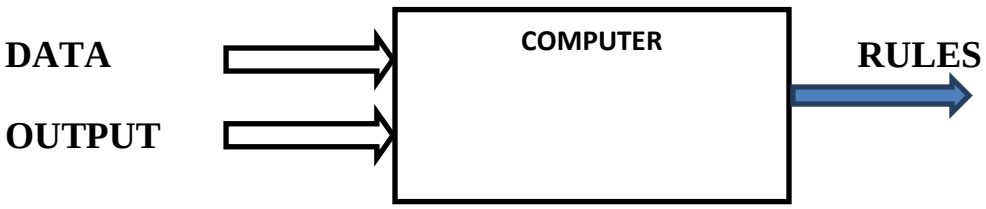


Figure:i

How does Machine learning work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector.** You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.

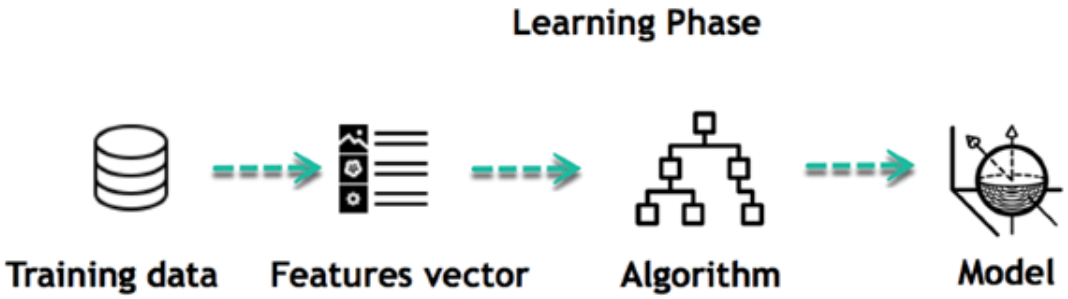


Figure:ii

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

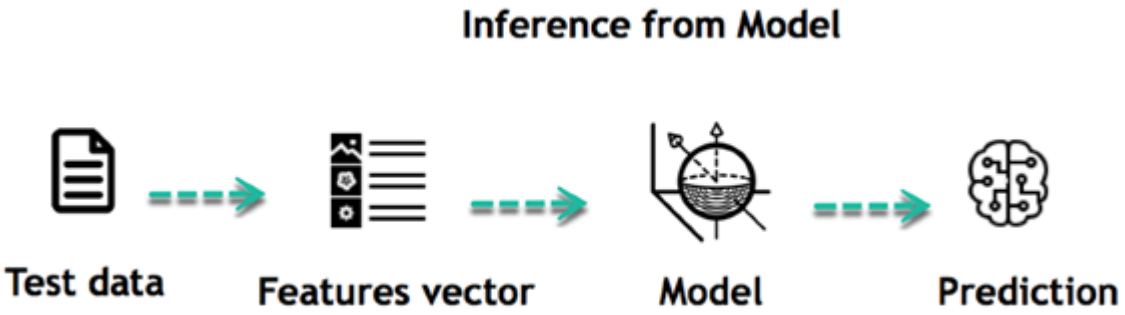


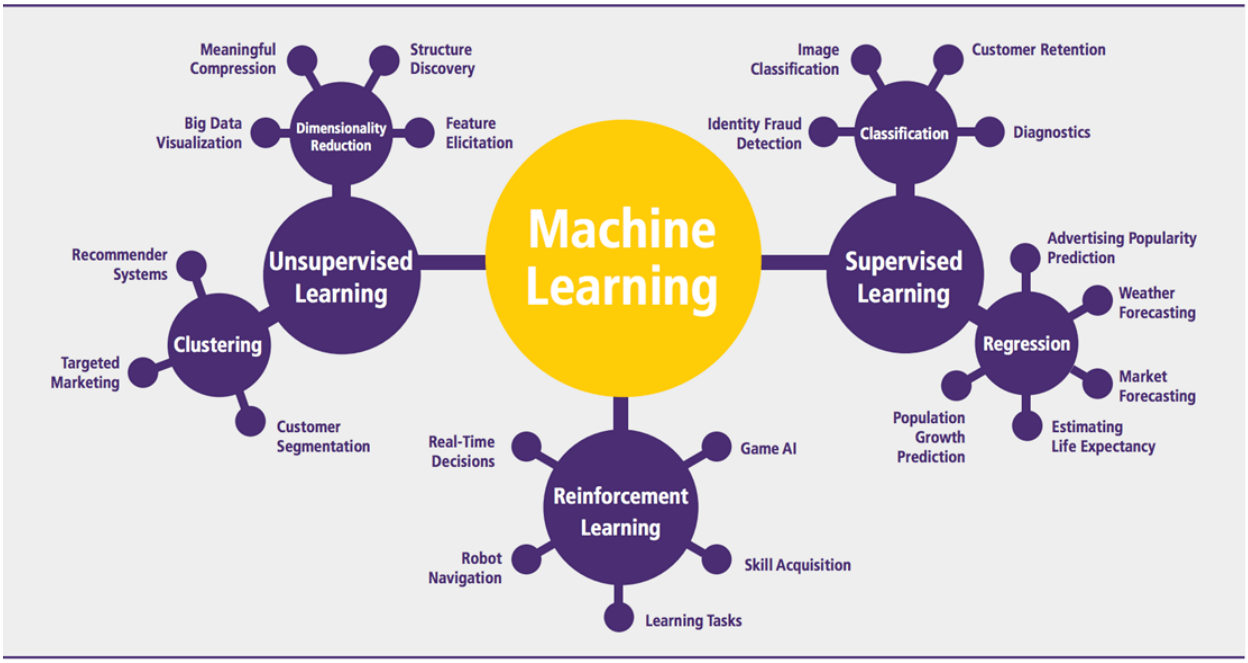
Figure:iii

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

Machine learning Algorithms and where they are used?



Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

- Classification task
- Regression task

Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only

be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

Algorithm Name	Description	Type
Linear regression	Finds a way to correlate each feature to the output to help predict future values.	Regression
Logistic regression	Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors)	Classification
Decision tree	Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a	Regression Classification

	new branch) until a final decision output is made	
Naive Bayes	The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event.	Regression Classification
Support vector machine	Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a non-linear solver.	Regression (not very common) Classification
Random forest	The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction.	Regression Classification
AdaBoost	Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome	Regression Classification
Gradient-boosting trees	Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it.	Regression Classification

Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

Algorithm	Description	Type
K-means clustering	Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans)	Clustering
Gaussian mixture model	A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters	Clustering
Hierarchical clustering	Splits clusters along a hierarchical tree to form a classification system. Can be used for Cluster loyalty-card customer	Clustering
Recommender system	Help to define the relevant data for making a recommendation.	Clustering
PCA/T-SNE	Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances.	Dimension Reduction

1.3.2 Application of Machine learning

Augmentation:

- Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

- Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry

- Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

Government organization

- The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry

- Healthcare was one of the first industry to use machine learning with image detection.

Marketing

- Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

1.3.3 Example of application of Machine Learning in Supply Chain

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs

CHAPTER 2

LITERATURE SURVEY

SNO	TITLE	YEAR	AUTHOR	CONCEPT	TECHNIQUE	FUTURE SCOPE	DRAWBACKS
1	Application of Machine Learning Techniques for Hate Speech Detection in Mobile Applications	2018	Bujar Raufi, Ildi Xhaferri	The proliferation of data through various platforms and applications is in constant increase. The versatility of data and its omnipresence makes it very hard to detect the trustworthiness and intention of the source. This is very evident in dynamic environments such as mobile applications. As a result, designing mobile applications that will monitor, control and block any type of malintents is important.	automatic hate speech detection, machine learning, artificial neural networks (ANNs)	The results indicated high level of classifier accuracy as well as applicability of the machine learning algorithms in mobile environments.	Not possible in hate speech and offensive language detection and prevention.
2	Analysis Text of Hate Speech Detection Using Recurrent Neural Network	2018	Arum Sucia Saksesi, Muhammad Nasrun, Casi Setianingsih	Twitter is very important for the success and destruction of one's image due to the many sentences of opinion that can compete the users. Examples of phrases that mean evil refer to hate speech to others. Evil perspectives can be categorized in hate speech, which hates speech is regulated in Article 28 of the ITE Law. Not a few people who intentionally and unintentionally oppose social media that contain hate speech. Unfortunately, social media does not have the ability to aggregate information about an existing conversation into a conclusion.	Hate Speech, Analysis text, Deep Learning, Recurrent Neural Network, RNN, LSTM	in this paper can make how to classify element of hate speech in the text by a computer, which later speech of hate can be recognized.	Needs an android device

SNO	TITLE	YEAR	AUTHOR	CONCEPT	TECHNIQUE	FUTURE SCOPE	DRAWBACKS
3	Text Analysis For Hate Speech Detection Using Back propagation Neural Network	2018	Nabiila Adani Setyadi, Muhammad Nasrun, Casi Setianingsih	They especially twitter is very important for the success and destruction of one's image due to the many sentences of opinion that can compete the users. To get hate speech information from the existing opinion data, in this final project will be done data processing with sentiment analysis using an Artificial Neural Network method optimized with back propagation algorithm. It can make how to classify element of hate speech in text by computer, which later speech of hate can be recognized	Hate Speech, Text Mining, Sentiment Analysis, Back propagation Neural Network	Hate speech is intimidates people from certain social groups oriented towards	Needs a high end processing platform
4	Detecting Hate Speech within the Terrorist Argument: A Greek Case	2018	Ioanna K. Lekea, Panagiotis Karampelas	Hate speech can be used by a terrorist group as a means of judging possible targets' guilt and deciding on their punishment, as well as a means of making people to accept acts of terror or even as propaganda for possibly attracting new members. To decide on how the automatic classification will be performed, we experimented with different text analyzing techniques such as critical discourse and content analysis and based on the preliminary results of these techniques a classification algorithm is proposed that can classify the communiqués in three categories depending on the presence of hate speech.	Greek terrorism, hate speech, tactics, targets, ideology, ethics, critical discourse analysis, content analysis	This paper presents a methodology for automatically detecting the presence of hate speech within the terrorist argument.	It is very limited because it is unable to handle a number of large existing data,

SNO	TITLE	YEAR	AUTHOR	CONCEPT	TECHNIQUE	FUTURE SCOPE	DRAWBACKS
5	Automatic Detection of Hate Speech on Facebook Using Sentiment and Emotion Analysis	2019	Axel Rodríguez, Carlos Argueta, Yi-Ling Chen	Hate speech has been an issue since the start of the Internet, but the advent of social media has brought it to unimaginable heights. To address such an important issue, in this paper, we explore a novel framework to effectively detect highly discussed topics that generate hate speech on Facebook. With the use of graph, sentiment, and emotion analysis techniques, we cluster and analyze posts on prominent Facebook pages. the definition of hate speech is conduct that uses direct assault with words on people who have particular traits, and this kind of assault usually has a tendency of violence or carries a tone of debasement.	Hate speech, Facebook, sentiment analysis, clustering	framework is able to identify the pages that promote hate speech in the comment sections.	Uses Matlab for processing

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System

Existing methods primarily cast the problem as a supervised document classification task .These can be divided into two categories: one relies on manual feature engineering that are then consumed by algorithms such as SVM, Naive Bayes, and Logistic Regression(classic methods); the other represents the more recent deep learning paradigm that employs neural networks to automatically learn multi-layers of abstract features from raw data(deep learning methods).

3.1.1 Disadvantage

- Existing studies on hate speech detection have primarily reported their results using micro-average Precision, Recall and F1.
- The problem with this is that in an unbalanced dataset where instances of one class (to be called the ‘dominant class’) significantly out-number others (to be called ‘minority classes’), micro-averaging can mask the real performance on minority classes.

3.2 Proposed System

All datasets are significantly biased towards non-hate, as hate Tweets account between only 5.8% (DT) and 31.6% (WZ). When we inspect specific types of hate, some can be even scarcer, such as ‘racism’ and as mentioned before, the extreme case of ‘both’. This has two implications. First, an evaluation measure such as the micro F1 that looks at a system’s performance on the entire dataset regardless of class difference can be biased to the system’s ability of detecting ‘non-hate’. In other words, a hypothetical system that achieves almost perfect F1 in identifying

‘racism’ Tweets can still be overshadowed by its poor F1 in identifying ‘non-hate’, and vice versa. Second, compared to non-hate, the training data for hate Tweets are very scarce. This may not be an issue that is easy to address as it seems, since the datasets are collected from Twitter and reflect the real nature of data imbalance in this domain. Thus to annotate more training data for hateful content we will almost certainly have to spend significantly more effort annotating non-hate.

3.2.1 Advantage

- Also, as we shall show in the following, this problem may not be easily mitigated by conventional methods of over- or under-sampling.
- Because the real challenge is the lack of unique, discriminative linguistic characteristics in hate Tweets compared to non-hate.
- As a proxy to quantify and compare the linguistic characteristics of hate and non-hate Tweets, we propose to study the ‘uniqueness’ of the vocabulary for each class.

3.3 FEASIBILITY STUDY

All projects are feasible, given unlimited resources and infinite time. Before going further into the steps of software development, the system analyst has to analyze whether the proposed system will be feasible for the organization and must identify the customer needs. The main purpose of a feasibility study is to determine whether the problem is worth solving. The success of a system also lies in the amount of feasibility studies performed on it. There are several feasibility studies that can be performed on any system. The three main feasibility tests that we perform are

- OPERATIONAL FEASIBILITY
- TECHNICAL FEASIBILITY
- ECONOMICAL FEASIBILITY

3.3.1 Operational Feasibility

During feasibility analysis, operational feasibility study is a must. This is due to the fact that according to software engineering principles, operational feasibility or in other words usability should be very high. A thorough analysis is performed and the system is found to be operational.

3.3.2 Technical Feasibility

A system analyst checks the technical feasibility of the proposed system by taking into account the hardware used by the system, data storage, processing and output and providing a thorough feasibility assessment. The system analyst has to check whether the company or user implementing the system has enough resources available for running the application smoothly without any hiccups. Since the requirements for this application is very less, the system is technically feasible.

3.3.3 Economical Feasibility

Before going any further into the development of the proposed system, the system analyst has to check the economic feasibility of the system and compares the required cost for running the system with the cost benefit that can be achieved by implementing the system.

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS:

System : i3core

Hard Disk:; 120 GB or above

Ram:4 GB(min) or above

4.2 SOFTWARE REQUIREMENTS:

Operating system : Windows7. Or above

Coding Language : Python

Tool : Anaconda Navigator

Database : MySQL

4.3 SOFTWARE DESCRIPTION

4.3.1 PYTHON

Python is a general-purpose, versatile and popular programming language. It's great as a first language because it is concise and easy to read, and it is also a good language to have in any programmer's stack as it can be used for everything from web development to soitware development and scientific applications. It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time.

4.3.1.1 Features of Python

A simple language which is easier to learn, Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

- **Free and open source**

  You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software's written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.
- **Portability**

  You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.
- **Extensible and Embeddable**

  Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.
- **A high-level, interpreted language**

  Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your computer

understands. You don't need to worry about any lower level operations.

- **Large standard libraries to solve common tasks**
  Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server You can use MySQLdb library using import MySQL db Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.
- **Object-oriented**
  Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively. With OOP, you are able to divide these complex problems into smaller sets by creating object

4.3.1.2 History and Versions of python

Python is predominantly a dynamic typed programming language which was initiated by Guido van Rossum in the year 1989. The major design philosophy that was given more importance was the readability of the code and expressing an idea in fewer lines of code rather than the verbose way of expressing things as in C++ and Java . The other design philosophy that was worth mentioning was that, there should be always a single way and a single obvious way to express a given task which is contradictory to other languages such as C++, Perl etc. Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The initial versions of Python were heavily inspired from lisp (for functional programming constructs). Python had heavily borrowed the module system, exception model and also keyword arguments

from Modula-3 language. Pythons' developers strive not to entertain premature optimization, even though it might increase the performance by a few basis points. During its design, the creators had conceptualized the language as being a very extensible language, and hence they had designed the language to have a small core library which was extended by a huge standard library. Thus as a result, python is used as a scripting language as it can be easily embedded into any application, though it can be used to develop a full-fledged application. The reference implementation of python is CPython. There are also other implementations like Jython, Iron Python which can use python syntax as well as can use any class of Java (Jython) or .Net class (Iron Python). Versions: Python has two versions 2.x version and 3.x version. The 3.x version is a backward incompatible release was released to fix many design issues which plagued the 2.x series. The latest in the 2.x series is 2.7.6 and the latest in 3.x series is 3.4.0.

4.3.1.3 Compilation, Execution and Memory Management

Compilation, Execution and Memory Management: 21 A Comparative Studies of Programming Languages (Comparative Studies of Six Programming Language) Just like the other Managed Languages, Python compiles to an intermediary code and this in turn is interpreted by the Python Runtime Environment to the Native Machine Code. The reference implementation (i.e. CPython) doesn't come with a JIT compiler because of which the execution speed is slow compared to native programming languages . We can use PyPy interpreter as it includes a JIT compiler rather than using the Python interpreter that comes by default with the python language, if speed of execution is one of the important factors . The Python Runtime Environment also takes care of all the allocation and deallocation of memory through the Garbage Collector. When a new object is created, the GC

allocates the necessary memory, and once the object goes out of its scope, the GC doesn't release memory immediately but instead it becomes eligible for Garbage Collection, which would eventually release the memory. Typing Strategies: Python is a strongly dynamic typed language. Python 3 also supports optional static typing . There are a few advantages in using a dynamic typed language, the most prominent one would be that the code is more readable as there is less code (in other words has less boiler-plate code). But the main disadvantage in having python as a dynamic programming language is that there would be no way to guarantee that a particular piece of code would run successfully for all the different data-types scenarios simply because it had run successfully with one type. Basically, we don't have any means to find out an error in the code, till the code has started running.

4.3.1.4 Strengths and Weaknesses and Application Areas:

Python is predominantly used as a scripting language used in developing standalone applications that are being developed with Static-Typed languages, because of the flexibility it provides due to its dynamic typed nature. Python favours rapid application development, which qualifies it to be used for prototyping. To a certain extent, Python is also used in developing websites. Due to its dynamic typing and of the presence of a Virtual Machine, there is a considerable overhead which translates to way less performance when we compare with native programming languages . And hence it is not suited.

4.3.2 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line

commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

4.3.2.1 WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QTConsole
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

Advanced conda users can also build your own Navigator applications

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What's new in 1.9?

- Add support for **Offline Mode** for all environment related actions.
- Add support for custom configuration of main windows links.
- Numerous bug fixes and performance enhancements.

4.4 MYSQL

MySQL is an open-source relational database management system(RDBMS), in Jul y 2013, it was the world's second most widely used RDBMS, and the most widely used open-source client–server model RDBMS. It is named after co-founder Michael Widenius's daughter, My. The SQL abbreviation stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.MYSQL is a popular choice of database for use in web applications,and is a central component of widely used LAMP open-source web

application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Free-software open-source projects that require a full-featured database management system often use MySQL. Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software. MySQL is also used in many high-profile, large-scale websites including Google (though not for searches, Facebook, Twitter, Fl.

.

CHAPTER 5

SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM

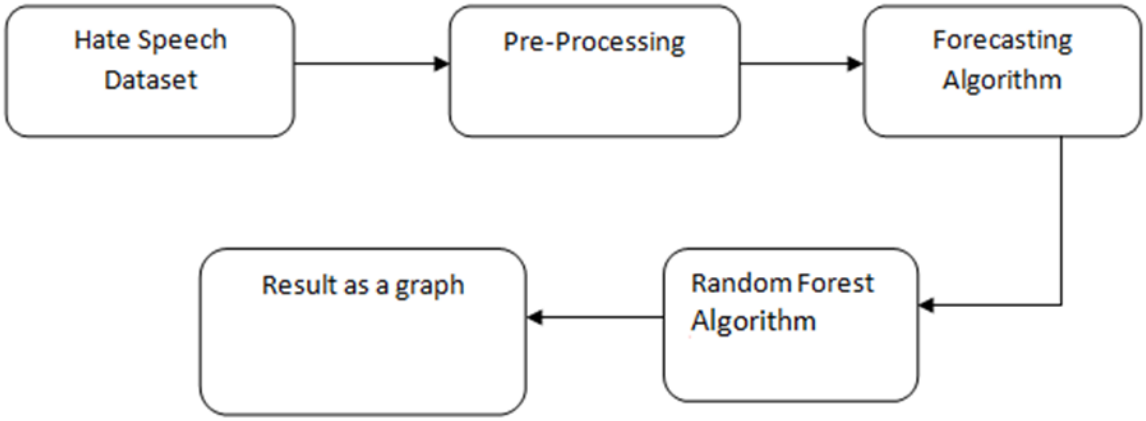


Figure:iv

5.2 UML DIAGRAM

UML is simply anther graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

ADVANTAGES

 To represent complete systems (instead of only the software portion) using object oriented concepts

 To establish an explicit coupling between concepts and executable code

 To take into account the scaling factors that are inherent to complex and critical systems

 To creating a modelling language usable by both humans and machines

UML defines several models for representing systems

 The class model captures the static structure

 The state model expresses the dynamic behavior of objects

 The use case model describes the requirements of the user

 The interaction model represents the scenarios and messages flows

 The implementation model shows the work units

5.3 USE CASE DIAGRAM

Use case diagrams overview the usage requirement for system. they are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe ―the meant‖ of the actual requirements. A use case describes a sequence of action that provide something of measurable value to an action.

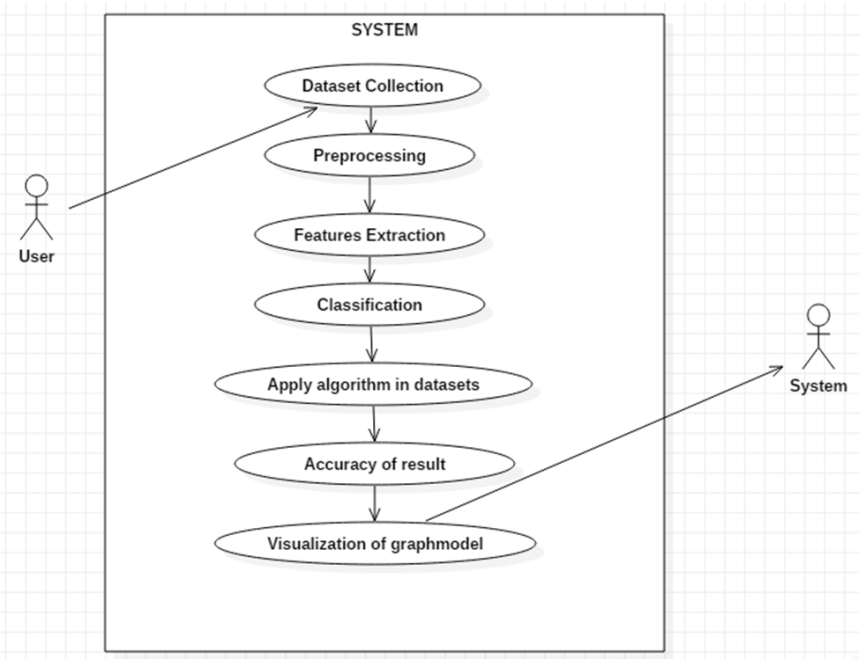


Figure:v

5.4 ACTIVITY DIAGRAM

Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Activity diagram consist of Initial node, activity final node and activities in between.

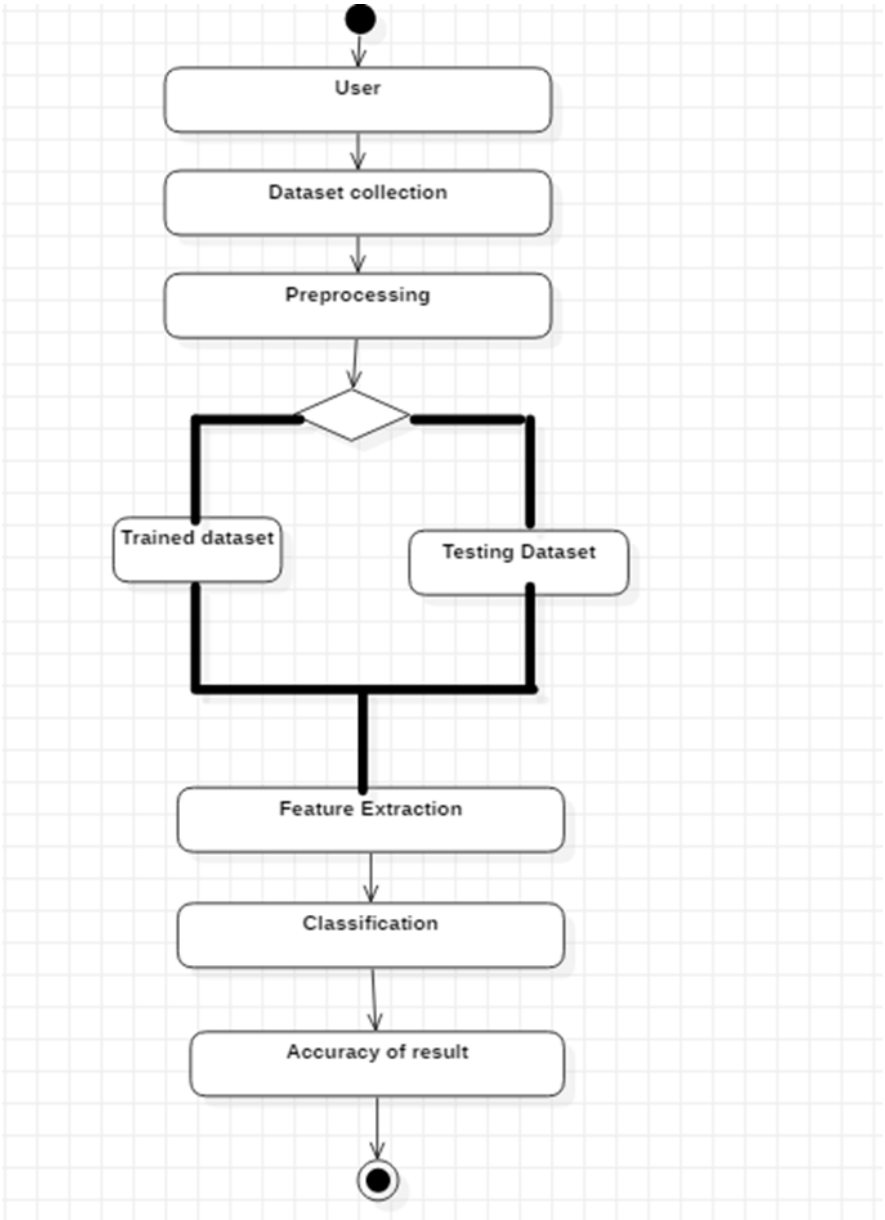


Figure:vi

5.5 SEQUENCE DIAGRAM

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and commonly used for both analysis and design purpose. Sequence diagram are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

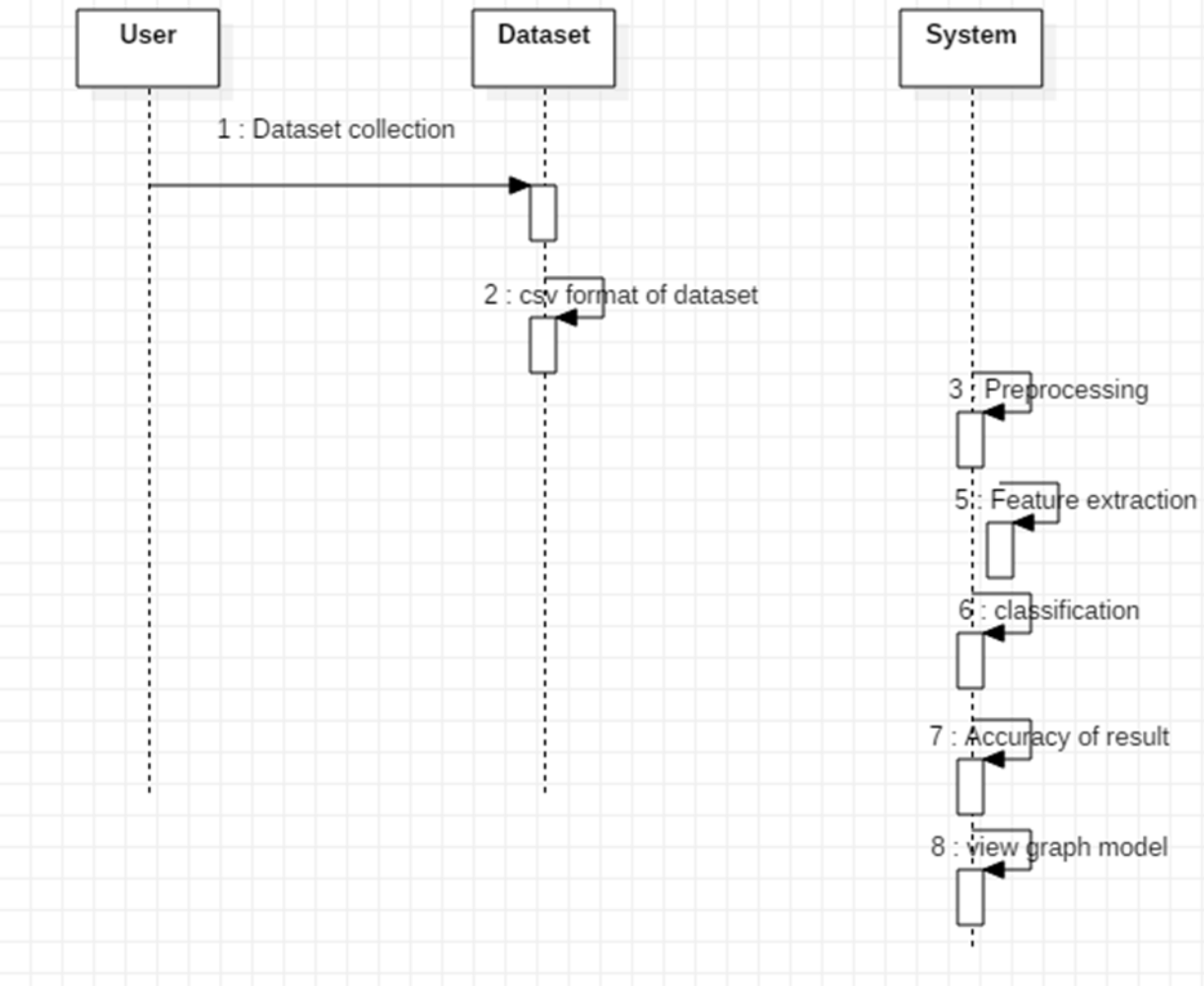


Figure:vii

5.6 CLASS DIAGRAM

In software engineering,a class diagram in the Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes,operation(or methods),and the relationships among objects.

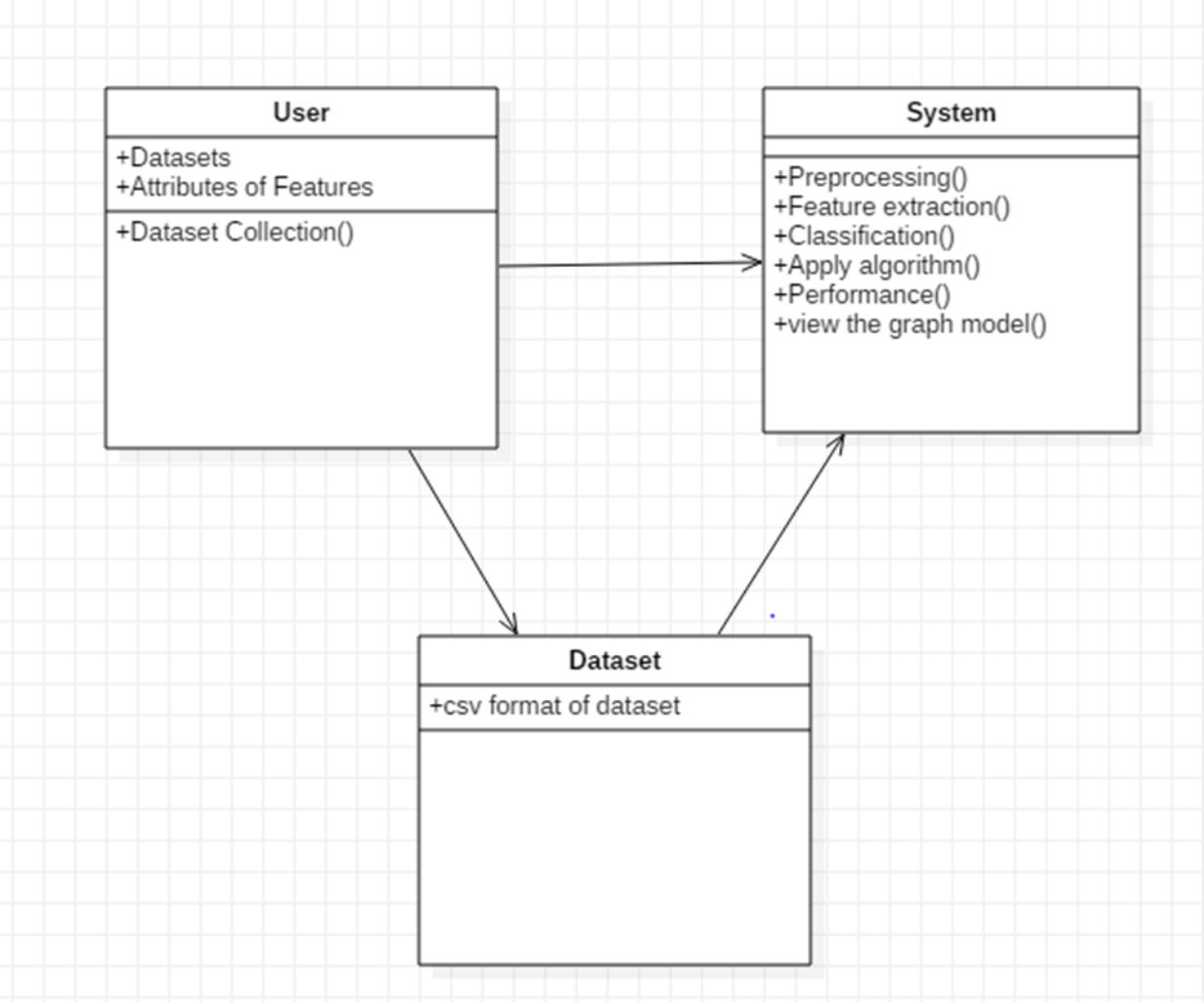
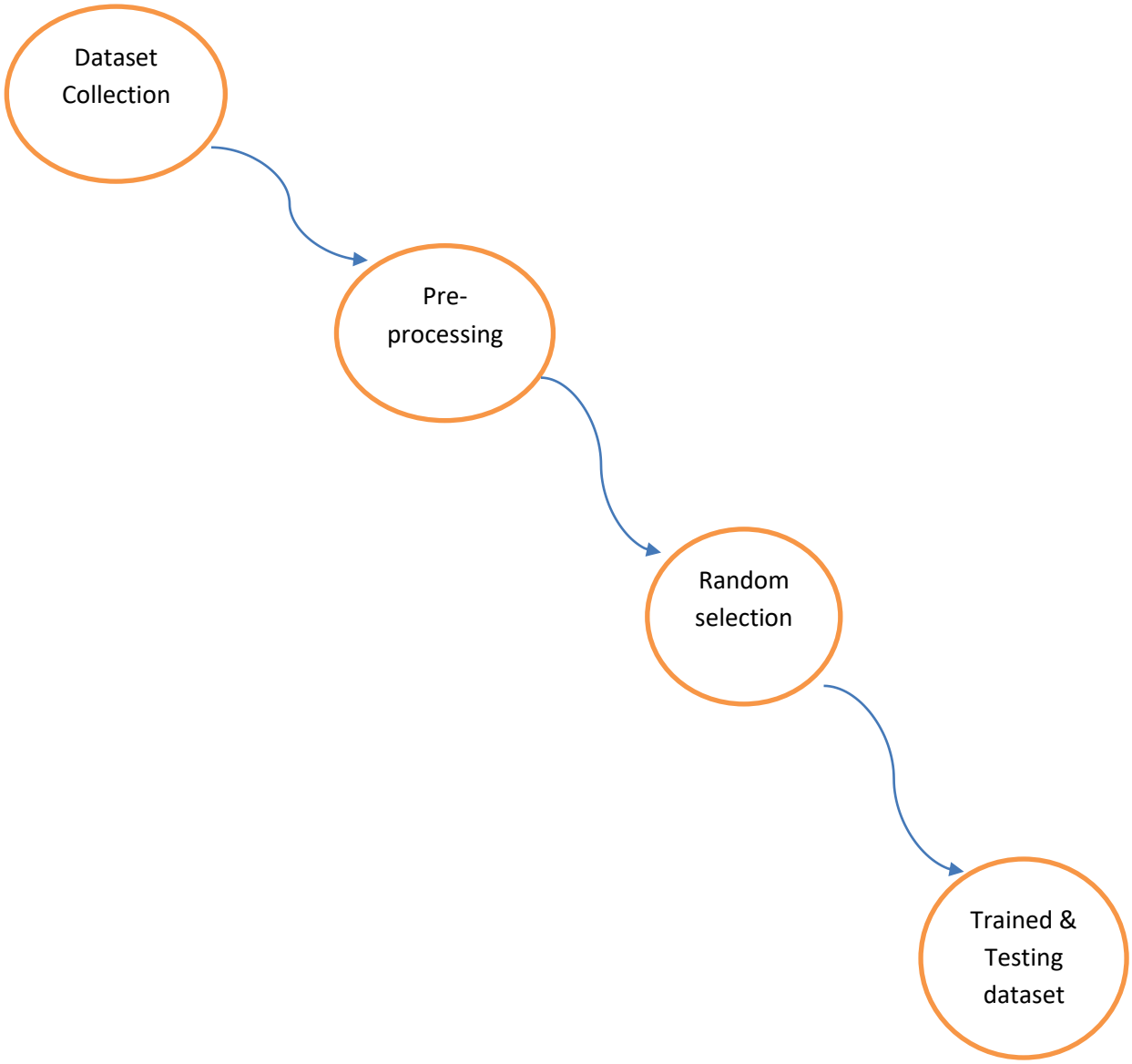


Figure:viii

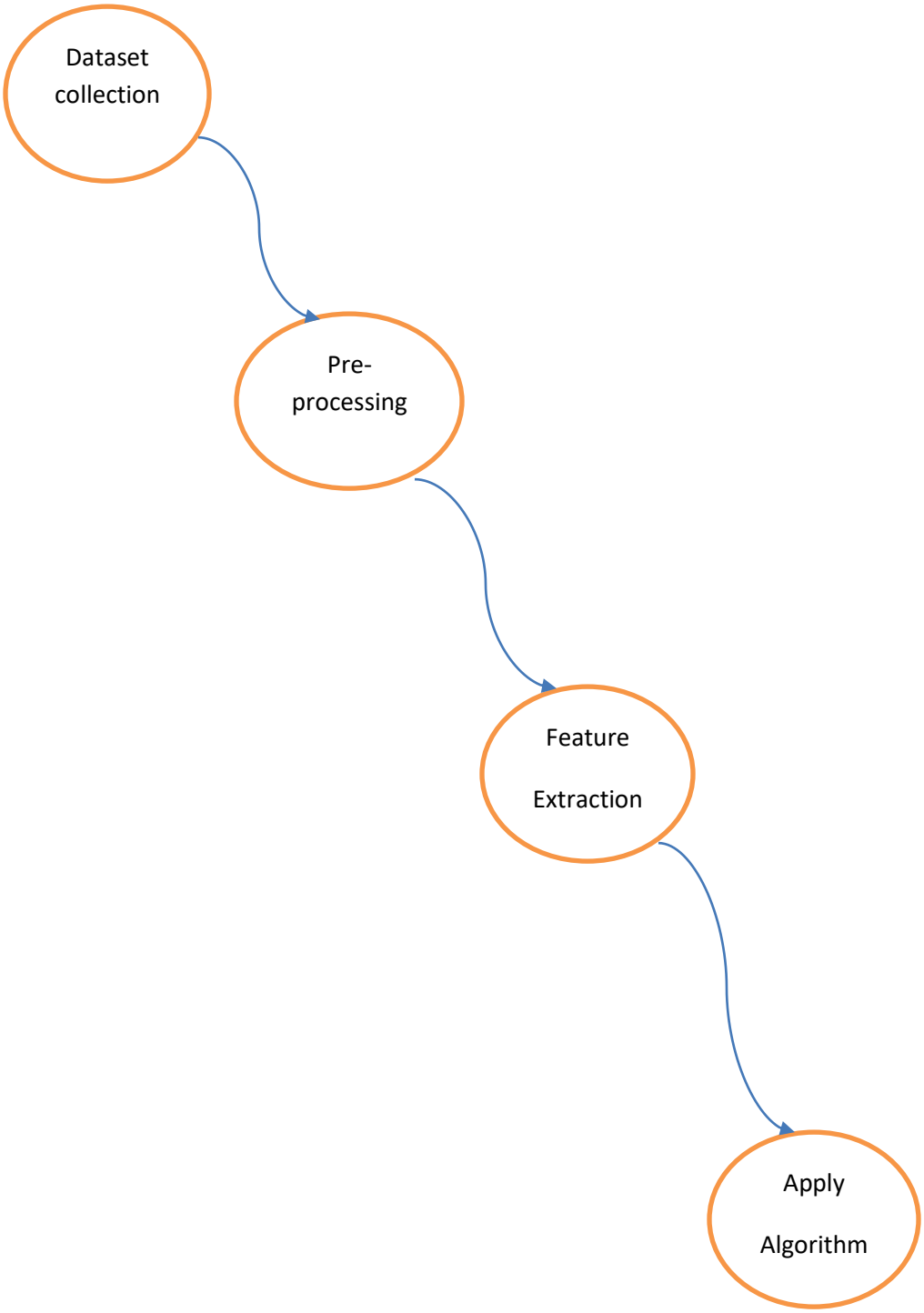
5.7 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing.

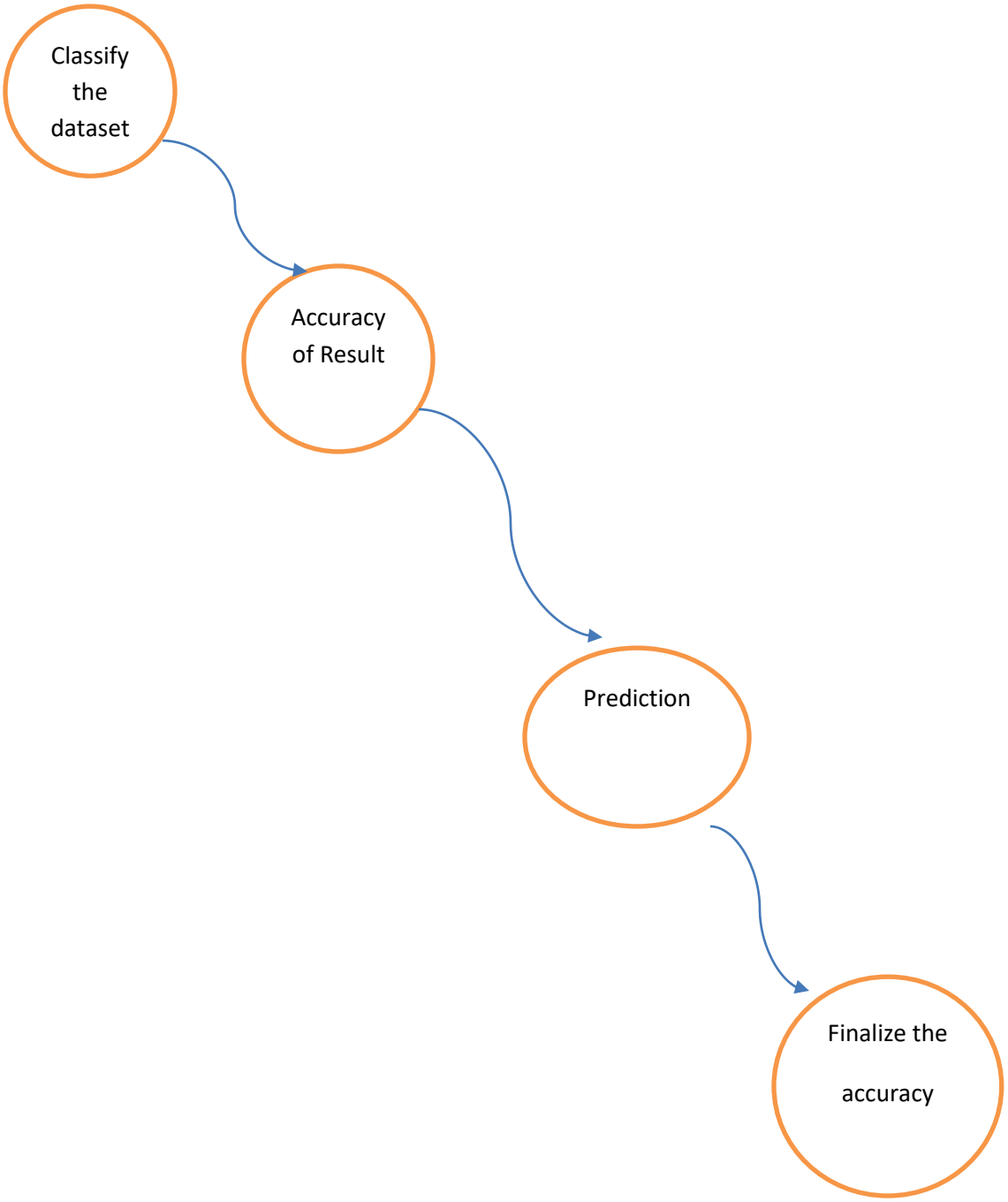
LEVEL 0



LEVEL 1


Dataset collection
Pre-processing
Feature Extraction
Apply Algorithm

LEVEL 2


Classify the dataset
Accuracy of Result
Prediction
Finalize the accuracy

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 MODULES

- Pre-processing
- Feature Extraction
- Feature Representation
- classifier

6.2 MODULE DESCRIPTION

6.2.1 Pre-processing

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as re-tweets, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We have applied an extensive number of pre-processing steps to standardize the dataset and reduce its size. We first do some general pre-processing on tweets which is as follows.Convert the tweet to lower case.Replace 2 or more dots (.) with space.Strip spaces and quotes (" and ') from the ends of tweet. Replace 2 or more spaces with a single space.

We handle special twitter features as follows.

6.2.1.1 URL

Users often share hyperlinks to other webpages in their tweets. Any particular URL is not important for text classification as it would lead to very sparse features. Therefore, we re-place all the URLs in tweets with the word URL.The regular expression used to match URLs is ((www\.[\S]+)|(https?://[\S]+)).

6.2.1.2 User Mention

Every twitter user has a handle associated with them. Users often mention other users in their tweets by @handle. We replace all user mentions with the word USER_MENTION. The regular expression used to match user mention is @[\S]+.

Emoticon(s)	Type	Regex	Replacemet
:), :), :-), (:, (:, (-:, :')	Smile	(:\s?\)|:-\)|\(\s?:|\(-:|:\'\))	EMO_POS
:D, : D, :-D, xD, x-D, XD, X-D	Laugh	(:\s?D|:-D|x-?D|X-?D)	EMO_POS
;-), ;), ;-D, ;D, (;, (-;	Wink	(:\s?\(|:-\(|\)\s?:|\)-:)	EMO_POS
<3, :*	Love	(<3|:*)	EMO_POS
:-(, : (, :(,):,)-:	Sad	(:\s?\(|:-\(|\)\s?:|\)-:)	EMO_NEG
:,(, :'(, :"(	Cry	(:,\(|:\'\(|:"\()	EMO_NEG

Table 1: List of emoticons matched by our method

6.2.1.3 Emoticon

Users often use a number of different emoticons in their tweet to convey different emotions. It is impossible to exhaustively match all the different emoticons used on social media as the number is ever increasing. However, we match some common emoticons which are used very frequently. We replace the matched emoticons with either EMO_POS or EMO_NEG depending on whether it is conveying a positive or a negative emotion. A list of all emoticons matched by our method is given in table 3.

6.2.1.4 Hashtag

Hashtags are unspaced phrases prefixed by the hash symbol (#) which is frequently used by users to mention a trending topic on twitter. We replace all the hashtags with the words with the hash symbol. For example, #hello is replaced by hello. The regular expression used to match hashtags is #(\S+).

6.2.1.5 Retweet

Retweets are tweets which have already been sent by someone else and are shared by other users. Retweets begin with the letters RT. We remove RT from the tweets as it is not an important feature for text classification. The regular expression used to match retweets is \brt\b.

After applying tweet level pre-processing, we processed individual words of tweets as follows.

Strip any punctuation [’"?!,.():;] from the word.

Convert 2 or more letter repetitions to 2 letters. Some people send tweets like I am sooooo happppppy adding multiple characters to emphasize on certain words. This is done to handle such tweets by converting them to I am soo happy.

Remove - and ’. This is done to handle words like t-shirt and their’s by converting them to the more general form tshirt and theirs.

Check if the word is valid and accept it only if it is. We define a valid word as a word which begins with an alphabet with successive characters being alphabets, numbers or one of dot

(.) and underscore(_).

Some example tweets from the training dataset and their normalized versions are shown in table 4.

6.2.2 Feature Extraction

We extract two types of features from our dataset, namely unigrams and bigrams. We create a frequency distribution of the unigrams and bigrams present in the dataset and choose top N unigrams and bigrams for our analysis.

6.2.2.1 Unigrams

Probably the simplest and the most commonly used features for text classification is the presence of single words or tokens in the the text. We extract single words from the training dataset and create a frequency distribution of these words. A total of 181232 unique words are extracted from

Raw	misses Swimming Class. http://plurk.com/p/12nt0b
Normalized	misses swimming class URL

Raw	@98PXYRochester HEYYYYYYYYY!! its Fer from Chile again
Normalized	USER_MENTION heyy its fer from chile again

Raw	Sometimes, You gotta hate #Windows updates.
Normalized	sometimes you gotta hate windows updates

Raw	@Santiago_Steph hii come talk to me i got candy :)
Normalized	USER_MENTION hii come talk to me i got candy EMO_POS

Raw	@bolly47 oh no :’(r.i.p. your bella
Normalized	USER_MENTION oh no EMO_NEG r.i.p your bella

Table 2: Example tweets from the dataset and their normalized versions.

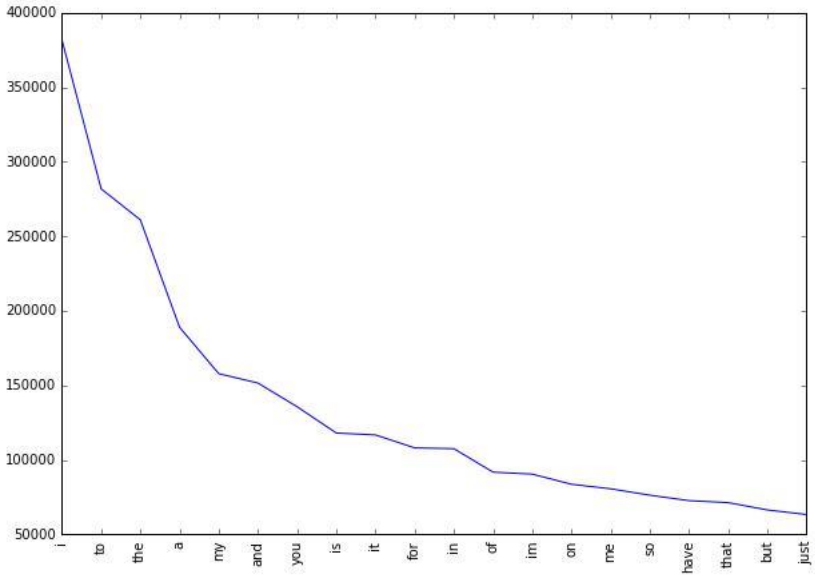


Figure ix: Frequencies of top 20 unigrams.

the dataset. Out of these words, most of the words at end of frequency spectrum are noise and occur very few times to influence classification. We, therefore, only use top N words from these to create our vocabulary where N is 15000 for sparse vector classification and 90000 for dense vector classification. The frequency distribution of top 20 words in our vocabulary is shown in figure 1. We can observe in figure 2 that the frequency distribution follows Zipf's law which states that in a large sample of words, the frequency of a word is inversely proportional to its rank in the frequency table. This can be seen by the fact that a linear trendline with a negative slope fits the plot of log (F requency) vs. log (Rank). The equation of the trendline shown in figure 2 is log(F requency) = 0:78 log(Rank) + 13:31.

6.2.2.2 Bigrams

Bigrams are word pairs in the dataset which occur in succession in the corpus. These features are a good way to model negation in natural language like in the phrase – This is not good. A total of 1954953 unique bigrams were extracted from the dataset. Out of these, most of the bigrams at end of frequency spectrum are noise and occur very few times to influence classification. We therefore use only top 10000 bigrams from these to create our vocabulary. The frequency distribution of top 20 bigrams in our vocabulary ~~is shown~~ in figure 3.

6.2.3 Feature Representation

After extracting the unigrams and bigrams, we represent each tweet as a feature vector in either sparse vector representation or dense vector representation depending on the classification method.

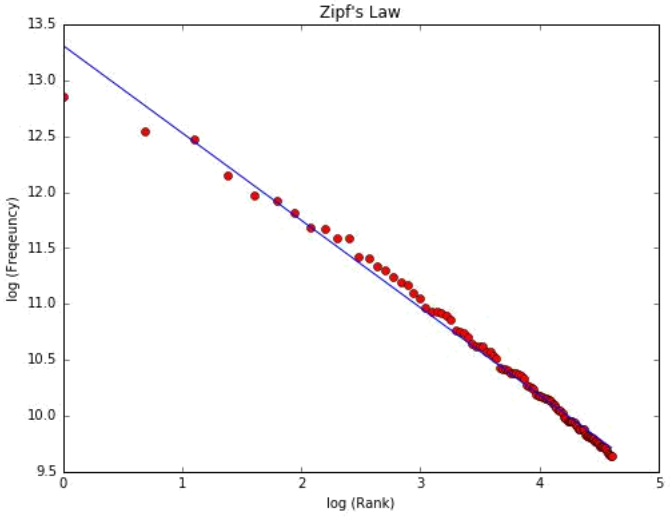


Figure x: Unigrams frequencies follow Zipf's Law.

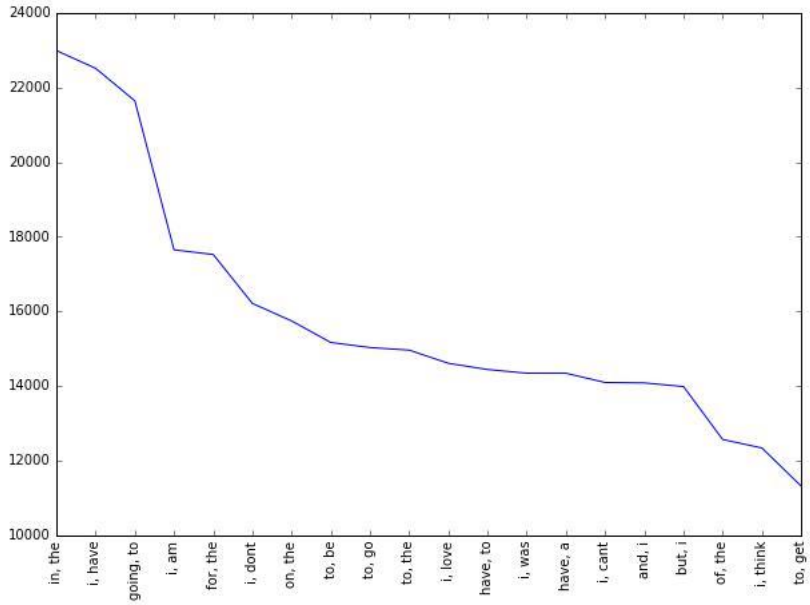


Figure xi: Frequencies of top 20 bigrams.

6.2.3.1 Dense Vector Representation

For dense vector representation we use a vocabulary of unigrams of size 90000 i.e. the top 90000 words in the dataset. We assign an integer index to each word depending on its rank (starting from

1)which means that the most common word is assigned the number 1, the second most common word is assigned the number 2 and so on. Each tweet is then represented by a vector of these indices which is a dense vector.

6.2.4 Classifiers

6.2.4.1 Naive Bayes

Naive Bayes is a simple model which can be used for text classification. In this model, the class c^ is assigned to a tweet t, where

$$c^{\wedge} = \text{argmax } P(c|t)$$

$$P(c|t) / P(c) \quad P(f_i|c)$$

In the formula above, f_i represents the i-th feature of total n features. P(c) and $P(f_i|c)$ can be obtained through maximum likelihood estimates.

6.2.4.2 Decision Tree

Decision trees are a classifier model in which each node of the tree represents a test on the attribute of the data set, and its children represent the outcomes. The leaf nodes represents the final classes of the data points. It is a supervised classifier model which uses data with known labels to form the decision tree and then the model is applied on the test data. For each node in the tree the best test condition or decision has to be taken. We use the GINI factor to decide the best split. For a given node t, GIN I(t) = 1 $^{P}_{j}$[p(jjt)]^2, where p(jjt) is the relative frequency of class j at node t, and GIN I_{split} = $^{Pk}\underline{ni}$—GIN I(i)_(n$_i$_=_number of records at child i, n = number of records at i=1to node p)indicates the quality of the split. We choose a split that minimizes the GINI factor.

6.2.4.3 Random Forest

Random Forest is an ensemble learning algorithm for classification and regression. Random Forest generates a multitude of decision trees classifies based on the aggregated decision of those trees. For a set of tweets x_1; x_2; : : : x_n and their respective sentiment labels y_1; y_2; : : :$_n$ bagging repeatedly selects a random sample (X_b, Y_b) with replacement. Each classification tree f_b is trained using a different random sample (X_b, Y_b) where b ranges from 1 : : : B. Finally, a majority vote is taken of predictions of these B trees.

6.2.4.4 SVM

SVM, also known as support vector machines, is a non-probabilistic binary linear classifier. For a training set of points (x_i; y_i) where x is the feature vector and y is the class, we want to find themaximum-margin hyperplane that divides the points with $y_i = 1$ and $y_i = 1$.The equation of the hyperplane is as follow ,w x b = 0,We want to maximize the margin, denoted by, as follows max ; s:t:8i;y_i(w x_i + b) , w; in order to separate the points well.

6.2.4.5 Convolutional Neural Networks

Convolutional Neural Networks or CNNs are a type of neural networks which involve layers called convolution layers which can interpret spacial data. A convolution layers has a number of filters or kernels which it learns to extract specific types of features from the data. The kernel is a 2D window which is slided over the input data performing the convolution operation. We use temporal convolution in our experiments which is suitable for analyzing sequential data like tweets.

CHAPTER 7

TESTING

7.1 INTRODUCTION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 TESTING METHODOLOGIES

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

7.3 TYPES OF TESTING

7.3.1 UNIT TESTING

Entire project is divided into modules and the testing performed on each of these modules is called unit testing. It concentrates on the particular module to find errors and to improve performance. Unit testing refers to numerically asserting the expected behavior of each function.

In the first phase, the images are pre-processed and tested accordingly.

In the second phase, they are segmented and the feature extractions are done.

In the third phase, training is done and the Image recognition is being tested.

Finally, the age range is estimated

7.3.2 INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between the components against a software design. Software components may be integrated in an iterative way or all together. Normally the former is considered a better practice since it allows interface issues to be localized quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components. Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

7.3.3 PERFORMANCE TESTING

In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

7.3.4 COMPATIBILITY TESTING

Compatibility testing, part of software non-functional tests, is testing conducted on the application to evaluate the application's compatibility with the computing environment.

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

As hate speech continues to be a societal problem, the need for automatic hate speech detection systems becomes more apparent. In this report, we proposed a solution to the detection of hate speech and offensive language on Twitter through machine learning using Bag of Words and TF IDF values. We performed comparative analysis of Logistic Regression, Naive Bayes, Decision Tree, Random Forest and Gradient Boosting on various sets of feature values and model parameters. The results showed that **Logistic Regression performs comparatively better with the TF IDF approach.** We presented the current problems for this task and our system that achieves reasonable accuracy (89%) as well as recall (84%). Given all the challenges that remain, there is a need for more research on this problem, including both technical and practical matters.

8.2 FUTURE WORK

In the future, this system can be expanded by introducing it to online social media content such as facebook and Instagram. users who desire to spread the hateful messages quickly find ways to circumvent these measures by for instance, posting the content as images containing the text ,rather than the text itself. although optical character recognition can be employed to solve the particular problem, this further demonstrates the difficulty of hate speech detection going forward . It will be a constant battle between those trying to spread hateful content and those trying to block it.

APPENDIX
SOURCE CODE AND SNAPSHOTS

In [6]:
```
import pandas as panda
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import *
import string
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix
import seaborn
from textstat.textstat import *
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.svm import LinearSVC
import numpy as np
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer as VS
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
```

Figure 4: importing packages

Loading the Data

In [2]:
```
dataset = panda.read_csv("labeled_data.csv")
dataset
```

Out[2]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
5	5	3	1	2	0	1	!!!!!!!!!!!!!!"@T_Madison_x: The shit just...
6	6	3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up ...
7	7	3	0	3	0	1	!!!!“@selfiequeenbri: cause I'm tired of...
8	8	3	0	3	0	1	" & you might not get ya bitch back & ...
9	9	3	1	2	0	1	" @rhythmixx_ :hobbies include: fighting Maria...
10	10	3	0	3	0	1	" Keeks is a bitch she curves everyone " lol I...
11	11	3	0	3	0	1	" Murda Gang bitch its Gang Land "
12	12	3	0	2	1	1	" So hoes that smoke are losers ? " yea ... go...
13	13	3	0	3	0	1	" bad bitches is the only thing that i like "
14	14	3	1	2	0	1	" bitch get up off me "
15	15	3	0	3	0	1	" bitch nigga miss me with it "
16	16	3	0	3	0	1	" bitch plz whatever "
17	17	3	1	2	0	1	" bitch who do you love "
18	18	3	0	3	0	1	" bitches get cut off everyday B "

Figure 5:loading data

24762	25275	3	1	2	0	1	you got niggas, and i got bitches.
24763	25276	3	0	2	1	1	you gotta be a new breed of retarded if you do...
24764	25277	3	0	3	0	1	you gotta understand that these bitches are ch...
24765	25278	3	0	3	0	1	you hoe spice
24766	25279	3	0	3	0	1	you just want some attention hoe
24767	25280	3	0	1	2	2	you know what they say, the early bird gets th...
24768	25281	3	0	3	0	1	you know what your doing when you favorite a t...
24769	25282	3	0	3	0	1	you lil dumb ass bitch, i ain't fuckin wit chu...
24770	25283	3	0	3	0	1	you look like AC Green...bitch don't call here...
24771	25284	3	0	3	0	1	you look like your 12 stop talking about fucki...
24772	25285	3	0	3	0	1	you might as well gone pussy pop on a stage
24773	25286	3	1	2	0	1	you niggers cheat on ya gf's? smh....
24774	25287	3	0	3	0	1	you really care bout dis bitch. my dick all in...
24775	25288	3	0	3	0	1	you worried bout other bitches, you need me for?
24776	25289	3	3	0	0	0	you're all niggers
24777	25290	3	2	1	0	0	you're such a retard i hope you get type 2 dia...
24778	25291	3	0	2	1	1	you's a muthaf***in lie “@LifeAsKing: @2...
24779	25292	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
24780	25294	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
24781	25295	6	0	6	0	1	youu got wild bitches tellin you lies
24782	25296	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

24783 rows × 7 columns

Figure 6: loading till last dataset

```
In [3]: dataset['class'].hist()
```

```
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc8aee7f60>
```

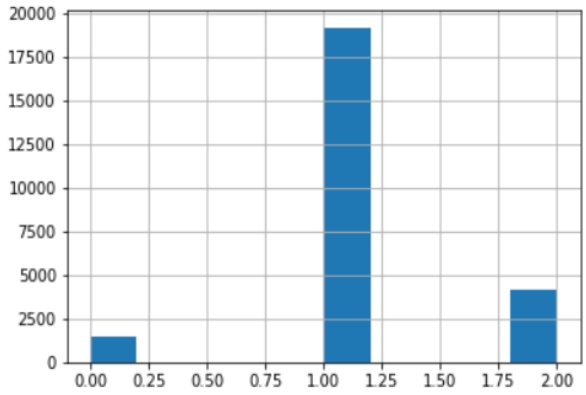


Figure 7: histogram representation of speech

In [5]:
```
# collecting only the tweets from the csv file into a variable name tweet
tweet=dataset.tweet
```

Preprocessing of the tweets

In [7]:
```
## 1. Removal of punctuation and capitlization
## 2. Tokenizing
## 3. Removal of stopwords
## 4. Stemming

stopwords = nltk.corpus.stopwords.words("english")

#extending the stopwords to include other words used in twitter such as retweet(rt) etc.
other_exclusions = ["#ff", "ff", "rt"]
stopwords.extend(other_exclusions)
stemmer = PorterStemmer()

def preprocess(tweet):

    # removal of extra spaces
    regex_pat = re.compile(r'\s+')
    tweet_space = tweet.str.replace(regex_pat, ' ')

    # removal of @name[mention]
    regex_pat = re.compile(r'@[\w\-]+')
    tweet_name = tweet_space.str.replace(regex_pat, '')

    # removal of links[https://abc.com]
    giant_url_regex =  re.compile('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
            '[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
    tweets = tweet_name.str.replace(giant_url_regex, '')

    # removal of punctuations and numbers
    punc_remove = tweets.str.replace("[^a-zA-Z]", " ")
```

Figure 8: preprocessing tweets

```
    # removal of capitalization
    tweet_lower = punc_remove.str.lower()

    # tokenizing
    tokenized_tweet = tweet_lower.apply(lambda x: x.split())

    # removal of stopwords
    tokenized_tweet=  tokenized_tweet.apply(lambda x: [item for item in x if item not in stopwords])

    # stemming of the tweets
    tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i in x])

    for i in range(len(tokenized_tweet)):
        tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
        tweets_p= tokenized_tweet

    return tweets_p

processed_tweets = preprocess(tweet)

dataset['processed_tweets'] = processed_tweets
dataset
```

Figure 9:preprcessing tweets

Out[7]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet	processed_tweets
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...	woman complain clean hous amp man alway take t...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	boy dat cold tyga dwn bad cuffin dat hoe st place
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	dawg ever fuck bitch start cri confus shit
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	look like tranni
4	4	6	0	6	0	1	!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	shit hear might true might faker bitch told ya
5	5	3	1	2	0	1	!!!!!!!!!!!!!"@T_Madison_x: The shit just...	shit blow claim faith somebodi still fuck hoe
6	6	3	0	3	0	1	!!!!!"@__BrighterDays: I can not just sit up ...	sit hate anoth bitch got much shit go
7	7	3	0	3	0	1	!!!!“@selfiequeenbri: cause I'm tired of...	caus tire big bitch come us skinni girl

Figure 10: preprocessed dataset

Visualizations

In [8]:
```
# visualizing which of the word is most commonly used in the twitter dataset

import matplotlib.pyplot as plt
from wordcloud import WordCloud

all_words = ' '.join([text for text in dataset['processed_tweets'] ])
wordcloud = WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(all_words)

plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

Figure 11: visualization

In [9]:
```
# visualizing which of the word is most commonly used for hatred speech
hatred_words = ' '.join([text for text in dataset['processed_tweets'][dataset['class'] == 1]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(hatred_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

Figure 12: visualization

In [10]:
```
# visualizing which of the word is most commonly used for offensive speech
offensive_words = ' '.join([text for text in dataset['processed_tweets'][dataset['class'] == 2]])
wordcloud = WordCloud(width=800, height=500,
random_state=21, max_font_size=110).generate(offensive_words)
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```

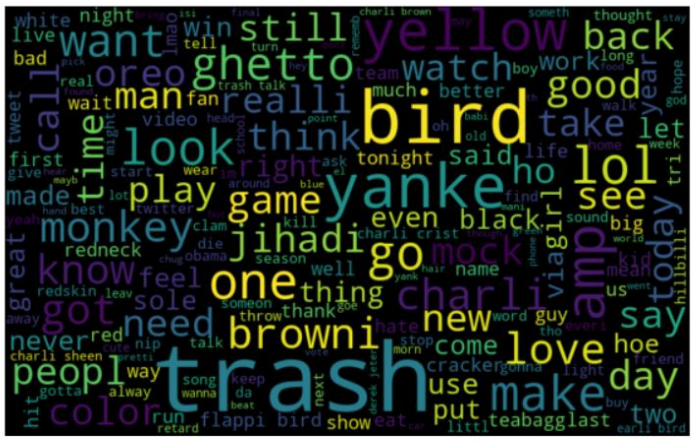


Figure 13: visualization

Feature Generation

In [11]:
```
# Bigram Features

bigram_vectorizer = CountVectorizer(ngram_range=(1,2),max_df=0.75, min_df=1, max_features=10000)
# bigram feature matrix
bigram = bigram_vectorizer.fit_transform(processed_tweets).toarray()
bigram
```

Out[11]:
```
array([[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, 0, ..., 0, 0, 0]], dtype=int64)
```

In [12]:
```
#TF-IDF Features

tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),max_df=0.75, min_df=5, max_features=10000)

# TF-IDF feature matrix
tfidf = tfidf_vectorizer.fit_transform(dataset['processed_tweets'] )
tfidf
```

Out[12]:
```
<24783x6441 sparse matrix of type '<class 'numpy.float64'>'
        with 189618 stored elements in Compressed Sparse Row format>
```

Figure 14: feature generation

Building Models using Logistic Regression

In [13]:
```
# Using Bigram Features
X = panda.DataFrame(bigram)
y = dataset['class'].astype(int)
X_train_bow, X_test_bow, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)


model = LogisticRegression(class_weight='balanced',penalty="l1", C=0.01).fit(X_train_bow,y_train)
y_preds = model.predict(X_test_bow)
report = classification_report( y_test, y_preds )
print(report)

print("Accuracy Score:" , accuracy_score(y_test,y_preds))
```

```
              precision    recall  f1-score   support

           0       0.43      0.41      0.42       164
           1       0.97      0.84      0.90      1905
           2       0.59      0.97      0.74       410

   avg / total       0.87      0.83      0.84      2479

   Accuracy Score: 0.8342073416700282
```

Figure 15: building model using logistic regression

In [65]:
```
# Running the model Using TFIDF without additional features

X = tfidf
y = dataset['class'].astype(int)
X_train_tfidf, X_test_tfidf, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)


model = LogisticRegression().fit(X_train_tfidf,y_train)
y_preds = model.predict(X_test_tfidf)
report = classification_report( y_test, y_preds )
print(report)

print("Accuracy Score:" , accuracy_score(y_test,y_preds))
```

```
                precision    recall  f1-score   support

             0       0.56      0.12      0.19       164
             1       0.90      0.97      0.93      1905
             2       0.84      0.81      0.83       410

   avg / total       0.87      0.88      0.86      2479

   Accuracy Score: 0.8846308995562727
```

Figure 16: running model with TFIDF without additional feature

In [15]:
```
# Sentiment Analysis
sentiment_analyzer = VS()
def count_tags(tweet_c):

    space_pattern = '\s+'
    giant_url_regex = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
        '[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
    mention_regex = '@[\w\-]+'
    hashtag_regex = '#[\w\-]+'
    parsed_text = re.sub(space_pattern, ' ', tweet_c)
    parsed_text = re.sub(giant_url_regex, 'URLHERE', parsed_text)
    parsed_text = re.sub(mention_regex, 'MENTIONHERE', parsed_text)
    parsed_text = re.sub(hashtag_regex, 'HASHTAGHERE', parsed_text)
    return(parsed_text.count('URLHERE'),parsed_text.count('MENTIONHERE'),parsed_text.count('HASHTAGHERE'))

def sentiment_analysis(tweet):
    sentiment = sentiment_analyzer.polarity_scores(tweet)
    twitter_objs = count_tags(tweet)
    features = [sentiment['neg'], sentiment['pos'], sentiment['neu'], sentiment['compound'],twitter_objs[0], twitter_objs[1],
                twitter_objs[2]]
    #features = pandas.DataFrame(features)
    return features

def sentiment_analysis_array(tweets):
    features=[]
    for t in tweets:
        features.append(sentiment_analysis(t))
    return np.array(features)

final_features = sentiment_analysis_array(tweet)
#final_features

new_features = panda.DataFrame({'Neg':final_features[:,0],'Pos':final_features[:,1],'Neu':final_features[:,2],'Compound':final_fea
                            'url_tag':final_features[:,4],'mention_tag':final_features[:,5],'hash_tag':final_features[:,6]})
```

Figure 17: sentimental analysis

```
new_features
```

Out[15]:

	Neg	Pos	Neu	Compound	url_tag	mention_tag	hash_tag
0	0.000	0.120	0.880	0.4563	0.0	1.0	0.0
1	0.237	0.000	0.763	-0.6876	0.0	1.0	0.0
2	0.538	0.000	0.462	-0.9550	0.0	2.0	0.0
3	0.000	0.344	0.656	0.5673	0.0	2.0	0.0
4	0.109	0.229	0.662	0.6331	0.0	1.0	1.0
5	0.101	0.155	0.744	0.3046	0.0	1.0	3.0
6	0.464	0.000	0.536	-0.9325	0.0	1.0	0.0
7	0.420	0.000	0.580	-0.8388	0.0	1.0	2.0
8	0.000	0.235	0.765	0.4717	0.0	0.0	0.0
9	0.612	0.000	0.388	-0.7430	0.0	1.0	0.0
10	0.310	0.256	0.434	-0.2789	0.0	0.0	0.0
11	0.432	0.000	0.568	-0.5859	0.0	0.0	0.0
12	0.254	0.000	0.746	-0.5267	0.0	0.0	0.0
13	0.497	0.168	0.336	-0.7096	0.0	0.0	0.0
14	0.487	0.000	0.513	-0.5859	0.0	0.0	0.0
15	0.574	0.000	0.426	-0.6597	0.0	0.0	0.0
16	0.623	0.213	0.164	-0.5423	0.0	0.0	0.0
17	0.345	0.382	0.273	0.1027	0.0	0.0	0.0
18	0.667	0.000	0.333	-0.7184	0.0	0.0	0.0
19	0.709	0.000	0.291	-0.8074	0.0	0.0	0.0

Figure 18: sentiment analysis

24783 rows × 7 columns

In [16]:
```
tfidf_a = tfidf.toarray()
modelling_features = np.concatenate([tfidf_a,final_features],axis=1)
modelling_features.shape
```

Out[16]:
```
(24783, 6448)
```

In [17]:
```
# Running the model Using TFIDF with some features from sentiment analysis

X = panda.DataFrame(modelling_features)
y = dataset['class'].astype(int)
X_train_bow, X_test_bow, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)

model = LogisticRegression().fit(X_train_bow,y_train)
y_preds = model.predict(X_test_bow)
report = classification_report( y_test, y_preds )
print(report)

print("Accuracy Score:" , accuracy_score(y_test,y_preds))
```

```
              precision    recall  f1-score   support

           0       0.58      0.12      0.19       164
           1       0.90      0.97      0.94      1905
           2       0.86      0.85      0.85       410

   avg / total       0.87      0.89      0.87      2479

   Accuracy Score: 0.8918918918918919
```

Figure 19: running model with TFIDF with sentiment analysis

In [18]:
```
def additional_features(tweet):

    syllables = textstat.syllable_count(tweet)
    num_chars = sum(len(w) for w in tweet)
    num_chars_total = len(tweet)
    num_terms = len(tweet.split())
    num_words = len(tweet.split())
    avg_syl = round(float((syllables+0.001))/float(num_words+0.001),4)
    num_unique_terms = len(set(tweet.split()))

    ###Modified FK grade, where avg words per sentence is just num words/1
    FKRA = round(float(0.39 * float(num_words)/1.0) + float(11.8 * avg_syl) - 15.59,1)
    ##Modified FRE score, where sentence fixed to 1
    FRE = round(206.835 - 1.015*(float(num_words)/1.0) - (84.6*float(avg_syl)),2)

    add_features=[FKRA, FRE,syllables, avg_syl, num_chars, num_chars_total, num_terms, num_words,
                  num_unique_terms]
    return add_features

def get_additonal_feature_array(tweets):
    features=[]
    for t in tweets:
        features.append(additional_features(t))
    return np.array(features)

fFeatures = get_additonal_feature_array(processed_tweets)
```

In [19]:
```
tfidf_a = tfidf.toarray()
modelling_features_enhanced = np.concatenate([tfidf_a,final_features,fFeatures],axis=1)
modelling_features_enhanced.shape
```

Out[19]:
```
(24783, 6457)
```

Figure 20: additional features

In [20]:
```
# Running the model Using TFIDF with additional features from sentiment analysis

X = panda.DataFrame(modelling_features_enhanced)
y = dataset['class'].astype(int)
X_train_features, X_test_features, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.1)

model = LogisticRegression().fit(X_train_features,y_train)
y_preds = model.predict(X_test_features)
report = classification_report( y_test, y_preds )
print(report)

print("Accuracy Score:" , accuracy_score(y_test,y_preds))
```

```
              precision    recall  f1-score   support

           0       0.62      0.13      0.21       164
           1       0.91      0.97      0.94      1905
           2       0.84      0.85      0.85       410

   avg / total       0.88      0.89      0.87      2479

   Accuracy Score: 0.8918918918918919
```

Figure 21: running model with TFIDF with additional features

In [21]:
```
#Confusion Matrix for TFIDF with additional features
confusion_matrix = confusion_matrix(y_test,y_preds)
matrix_proportions = np.zeros((3,3))
for i in range(0,3):
    matrix_proportions[i,:] = confusion_matrix[i,:]/float(confusion_matrix[i,:].sum())
names=['Hate','Offensive','Neither']
confusion_df = panda.DataFrame(matrix_proportions, index=names,columns=names)
plt.figure(figsize=(5,5))
seaborn.heatmap(confusion_df,annot=True,annot_kws={"size": 12},cmap='YlGnBu',cbar=False, square=True,fmt='.2f')
plt.ylabel(r'True Value',fontsize=14)
plt.xlabel(r'Predicted Value',fontsize=14)
plt.tick_params(labelsize=12)
```

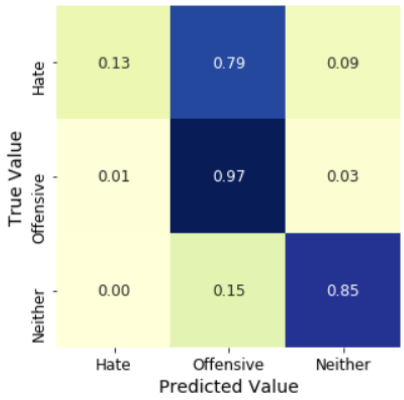


Figure 22: confusion matrix for TFIDF with additional feature

LOGIN PAGE

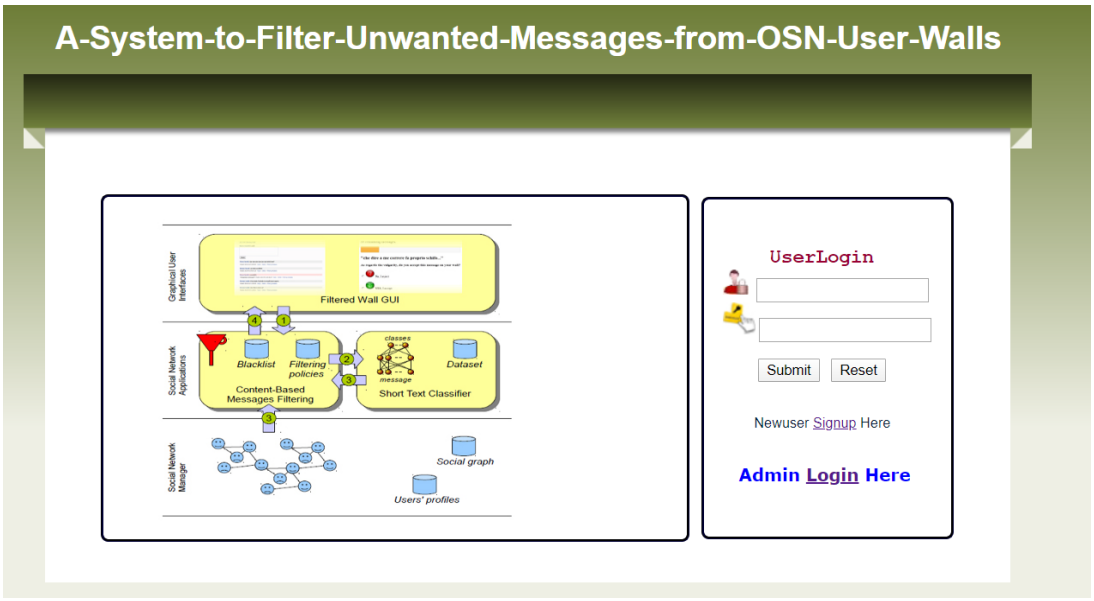


Figure 23 : user,admin login and new signin registration

COMMENTS PAGE

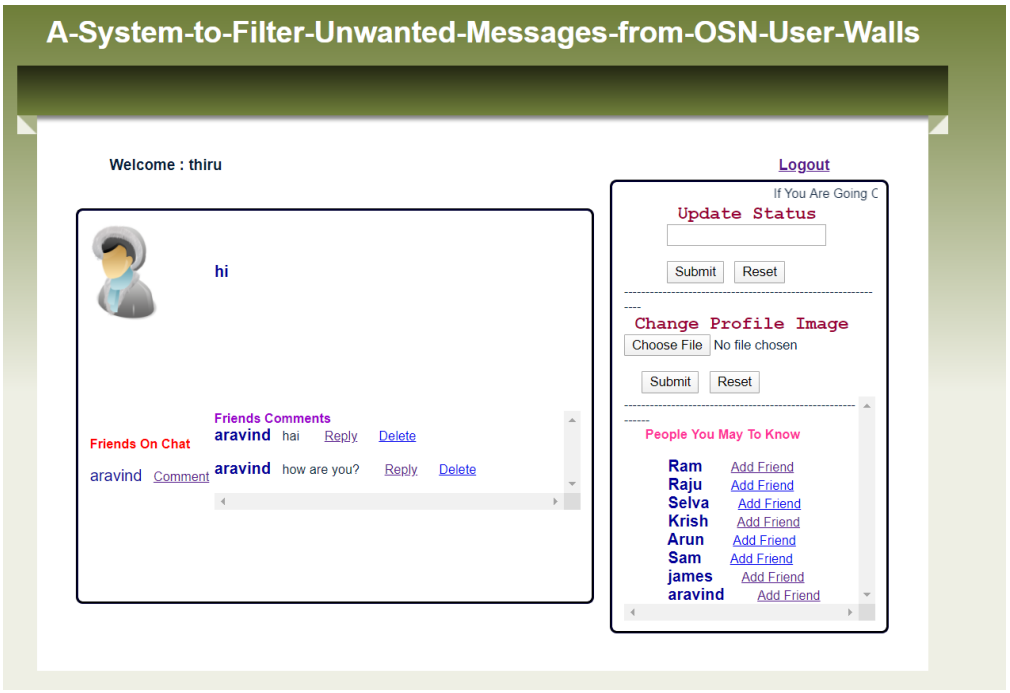


Figure 24 : comments page

FILTER PAGE

Figure 25 : filtering the hate words

REFERENCES

[1] Albert Biffet and Eibe Frank. Sentiment Knowledge Discovery in Twitter Streaming Data. Discovery Science, Lecture Notes in Computer Science, 2010, Volume 6332/2010, 1-15, DOI: 10.1007/978-3-642-16184-1_1

[2] Alec Go, Richa Bhayani and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Project Technical Report, Stanford University, 2009.

[3] Alexander Pak and Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of international conference on Language Resources and Evaluation (LREC), 2010.

[4] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner and Isabell M. Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2010.

[5] Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002.

[6] Chenhao Tan, Lilian Lee, Jie Tang, Long Jiang, Ming Zhou and Ping Li. User Level Sentiment Analysis Incorporating Social Networks. In Proceedings of ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD), 2011.

[7] Efthymios Kouloumpis, Theresa Wilson and Johanna Moore. Twitter Sentiment Analysis: The Good the Bad and the OMG! In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2011.

[8] Hatzivassiloglou, V., & McKeown, K.R.. Predicting the semantic orientation of adjectives. In Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL, 2009.

[9] Johann Bollen, Alberto Pepe and Huina Mao. Modelling Public Mood and Emotion: Twitter Sentiment and socio-economic phenomena. In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2011.

[10] Luciano Barbosa and Junlan Feng. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In Proceedings of the international conference on Computational Linguistics (COLING), 2010.

[11] Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL), 2002.

[12] Rudy Prabowo and Mike Thelwall. Sentiment Analysis: A Combined Approach. Journal of Infometrics, Volume 3, Issue 2, April 2009, Pages 143-157, 2009.

[13] Samuel Brody and Nicholas Diakopoulus. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect

Sentiment in Microblogs. In Proceedings of Empirical Methods on Natural Language Processing (EMNLP), 2011.

[14] Soo-Min Kim and Eduard Hovy. Determining the Sentiment of Opinions. In Proceedings of International Conference on Computational Linguistics (ICCL), 2004.

[15] Stefano Baccianella, Andrea Esuli, Fabrizio Sebastiani. SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In Proceedings of international conference on Language Resources and Evaluation (LREC), 2010.

[16] Theresa Wilson, Janyce Wiebe and Paul Hoffmann. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In the Annual Meeting of Association of Computational Linguistics: Human Language Technologies (ACL-HLT), 2005.

[17] Ian H. Witten, Eibe Frank & Mark A. Hall. Data Mining – Practical Machine Learning Tools and Techniques.

[19] Ricgard O. Duda, Peter E. Hart & David G. Stork: Pattern Classification.

[19] Steven Bird, Even Klein & Edward Loper. Natural Language Processing with Python.

[20] Ben Parr. Twitter Has 100 Million Monthly Active Users; 50% Log In Everyday. <http://mashable.com/2011/10/17/twitter-costolo-stats/>

[21] Google App Engine <https://developers.google.com/appengine/>

[22] Google Chart API <https://developers.google.com/chart/>

[23] Jinja2: Templating Language for Python <http://jinja.pocoo.org/>

[24] Maggie Shields, Technology Reporter, BBC News. Twitter co-founder Jack Dorsey rejoins company. <http://www.bbc.co.uk/news/business-12889048>.

[25] Multi Perspective Question Answering (MPQA) Online Lexicon <http://www.cs.pitt.edu/mpqa/subj_lexicon.html>

PUBLICATION

Thirunayan Manikantan R, Aravind Kumar D, Harish Krishna K, “Detection of hate speech and offensive in Twitter using Sentiment Analysis”, VDGOOD Journal of Computer Science and Engineering, Vol 1, Issue 2, pp 351-356, April 2020

VDGOOD Journal of Computer Science Engineering

Detection of Hate Speech and Offensive Language in Twitter Using Sentiment Analysis

Harish Krishna K[a*], Thirunayan Manikantan R[b], Aravind Kumar D[c], Amirthavalli R[d],

[a,b,c]*UG Scholar, Department of Computer Science and Engineering, Velammal Engineering College, Chennai, India.*

[d]*Assistant Professor, Department of Computer Science and Engineering, Velammal Engineering College, Chennai, India*

VDGOOD use only: Received 14-Mar-2020; revised 30-Mar-2020; accepted 20-Apr-2020

Abstract

The exponential growth of social media such as Twitter and other community forums has revolutionized communication and content publishing, but is also increasingly exploited for the propagation of hate speech and the organization of hate-based activities. The anonymity and mobility standard afforded by such media has made the breeding and spread of hate speech – eventually leading to hate crime – effortless in a virtual world that are beyond the realms of law enforcement. Existing methods in the detection of hate speech primarily cast the problem as a supervised document classification task. These can be divided into two categories: one relies on manual feature engineering that are then consumed by algorithms such as SVM, Naive Bayes, and Logistic Regression (classic methods); the other represents the more recent deep learning paradigm that employs neural networks to automatically learn multilayers of abstract features from raw data (deep learning methods). In this method we show that it is a much more challenging task, as our analysis of the language in the typical datasets shows that hate speech lacks unique, discriminative effects and therefore is found in the 'long tail' in a dataset that is difficult to discover. We then propose Deep Neural Network (DNN) structure as a unique feature extractors that are particularly effective for capturing the semantics of hate speech. Our methods are tested on the largest collection of hate speech datasets based on Twitter, and are shown to be able to outperform state of the art by up to 6 percentage points in macro-average F1, or 9 percentage points in the more challenging case of identifying hateful content. As a proxy to quantify and compare the linguistic characteristics of hate and non-hate Tweets, we also propose to study the 'uniqueness' of the vocabulary for each class.

Keywords: Classic Methods; DNN; Detection of hate speech and offensive language in Twitter; Sentimental Analysis.

1. Introduction

We have chosen to work with twitter as we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompting and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly difficult in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could

be done by analyzing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm's stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favorable response and in which a negative response (since twitter allows us to download stream of geo-tagged tweets from a particular locations. If firms can able to get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions by establishing suitable market segments. Predicting the results of the political elections and polls is also an emerging application to sentiment analysis. One of a study was conducted by Tumasjan et al. In Germany for predicting the result of the federal elections in which concluded that twitter is such a good reflection of offline sentiment analysis .

2. Data Description

The data given is in the form of a comma-separated values in other words CSV files that has tweets and their corresponding sentiments. The training datasets in the csv file is of the type format tweet_id, sentiment, tweet where the tweet_id is a unique integer identifying the tweet, sentiment is either 1(positive) or 0(negative), and tweet is put inside in " ". Similarly, the training dataset is a csv file of type tweet_id, tweet.

The dataset is a collection of words, emoticons, symbols, URLs and references to people. Words and emoticons provide to predicting the sentiment, but URLs and references to people don't. Therefore, URLs and references can be ignored. The words are also a mixture of misspelled words,words with many repeated letters and extra punctuations. The tweets, thereby, need to be pre-processed to standardize the dataset .The provided training and test dataset have 5000 and 1000 tweets respectively. Preliminary and the statistical analysis of the contents of datasets, after preprocessing might change.

2.1. Pre-processing

Raw tweets taken from the twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as re-tweets, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, the raw twitter datasets has to be normalized to create a dataset in which it can be easily learned by various classifiers. We have applied an extensive number of pre-processing procedures for standardizing the dataset and reduce its size. We first do some general pre- processing of data set tweets which are.

Convert the tweet to lower case. Replace 2 or more dots (.) with space.

Remove the spaces and quotations from the ends of tweet datasets. Change two or more spaces with a single space.

2.2. URL

Most of the users often share hyperlinks to other webpages in their tweets and comments . Any particular URL is not important for text divisions as this would lead to very sparse features. Therefore, we replace all the URLs in tweets with the word URL. The regular expression used to match URLs is ((www\.[\S]+)|(https?://[\S]+)).

2.3. User Mention

Every twitter account users will have a handle related with them. Users are most often mentioning the

other users in their tweets by a at symbol eg:@handle. We will remove all account with the word USER_MENTION. The regex used to match account mention with @[\S]+. Make sure that the symbols you use in the equation has been defined previously or after the equation.

2.4. Emoticons

Users often use emotions to express. It is impossible to exhaustively match all emoticons used on social media. However, we try match some most commonly used emoticons which are used very frequently. We replace the matched emoticons to EMO_POS or EMO_NEG depending upon on it's a positive or a negative emotion.

2.5. Hashtag

Hashtags are unspaced phrases prefixed by the hash symbol (#) which is frequently used by users to mention a trending topic on twitter. We replace all the hashtags with the words with # symbol . For example, #hello is replaced by hello. The regex used to match hashtags is #(\S+).

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet	processed_tweets
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...	woman complain clean hous amp man alway take t...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold.. tyga dwn ba...	boy dat cold tyga dwn bad cuffin dat hoe st place
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @RobiRo...	dawg ever fuck bitch start cri confus shit
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	look like tranni
4	4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	shit hear might true might faker bitch told ya
5	5	3	1	2	0	1	!!!!!!!!!!!!! "@T_Madison_x: The shit just...	shit blow claim faith somebodi still fuck hoe
6	6	3	0	3	0	1	!!!!!"@__BrighterDays: I can not just sit up ...	sit hate anoth bitch got much shit go
7	7	3	0	3	0	1	!!!“@selfequeenbri: cause I'm tired of...	caus tire big bitch come us skinni girl

Fig.1. Data set of twiter hate speech

3. Feature Extraction

We extract 2 types of features, namely unigrams and bigrams. We produce a frequency distribution among unigrams and bigrams present in the dataset and choose top N unigrams and bigrams for our analysis.

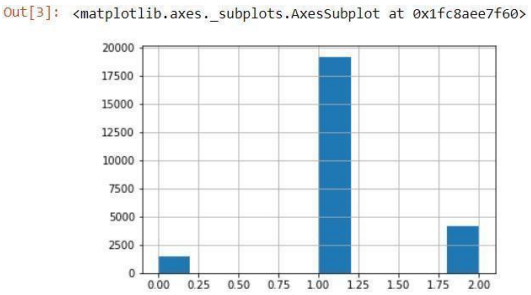


Fig.2. Insight of the data set in offencive language

A. Unigrams

Probably the simplest and the most commonly used features for text classification is the presence of single words or tokens in the the text. We, therefore, use top N words from these to create our own word sets where N is 15000 for sparse vector classification and 90000 for dense vector classification. The frequency distribution follows Zipf's law which states that in a large sample of words, the frequency of a word is inversely proportional to its rank in the frequency table. This can be seen by the fact that a linear trendline with a negative slope fits the plot of log(Frequency) vs. log(Rank).

B. Bigrams

Bigrams are word pairs in the dataset which occur in succession in the corpus. These features are a good way to model negation in natural language like in the phrase – This is not good. We therefore use only top 100000 bigrams from these to create our words list.

4. Feature Representation

After extracting the unigrams and bigrams, we represent each tweet as a feature vector in either sparse vector representation or dense vector representation depending on the classification method.

A. Dense Vector Representation

For dense vector representation we use a word sets of unigrams of size 90000 i.e. the top 90000 words in the dataset. We assign an corresponding integer index to each word depending on its rank (starting from the index 0) which means that the most common word is

assigned the number 10, the second most common word is assigned the number 11 and so on. Each tweet is then represented by a vector of these indices which is a dense vector.

B. Naive Bayes

Naive Bayes is a simple that can be used for text classification. In this model, the class c^ is assigned to a tweet t, where In the formula above, fi represents the i-th feature of total n features. P(c) and P(fi|c) can be obtained through

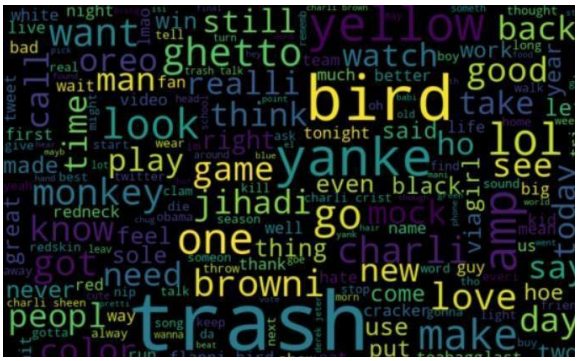


Fig.3. Representation of the hate words in bigraph

maximum likelihood estimates.We used MultinomialNB from sklearn.naive_bayes package of scikit-learn for Naive Bayes classification. We used Laplace smoothed version of Naive Bayes with the smoothing parameter set to its default value of 1. We found that presence features outperform frequency features because Naive Bayes is built to work best on integer features in data sets rather than traditional features.

C. Decision Tree

Decision trees are a classifier model that compares each node of the tree is a test on the data set, and its children represent the outcomes. The leaf nodes represents the final classes of the data points.For each node in the tree the best test condition or decision has to be taken. We use the GINI factor to decide the best split. We use the Decision Tree Classifier in sklearn.tree package given by scikit-learn to build our representation . GINI can be used to evaluate the split of every node and the best split is considered . The model has been performing a bit better using the presence this feature compared to frequency. Also using unigrams didn't make any significant improvements. The best accuracy achieved using decision trees was 68.1%.

D. Random Forest

Random Forest is the most popular learning algorithm for classification and regression. Random Forest creates a multiple and multitude of decision trees classifies based on the aggregated decision corresponding to the trees. For a set of tweets x1; x2; : : : xn and their respective sentiment labels y1; y2; : : :n bagging repeatedly selects a random sample (Xb, Yb) with replacement. Each divisionof tree fb is trained by random sample (Xb, Yb) where b ranges from 1 : : : B. Finally, the decisions are based on the result of the B trees.We implemented random forest algorithm by using Random Forest Classifier in sklearn.ensemble given by scikit-learn. We tested using 10 estimators (trees based) using both presence system and frequency features. Presence features updates performed better than frequency but the improvement was not substantial.

E. SVM

SVM, aka support vector machines, is a non-probabilistic binary linear classifier in the data set . For a training set of points (xi; yi) where x is the feature vector and y is the class, we want to find the maximum-margin hyperplane that divides the points with yi = 1 .

F. Convolutional Neural Network

We used keras with TensorFlow in the backend to implement the (CNN) Convolutional Neural Network model. We used the Dense Vector Representation (DVR) of the tweets to train our CNN models. We used a vocabulary of top 5000 words from the training dataset. We represent each word in our vocabulary with an integer index from 1 : : : 5000 where the integer index represents the rank of the word in the dataset. The integer index 0 is reserved for the special padding word. Further each of these 5000+1 words is represented by a 200 dimensional vector. The first layer of our models is the Embedding layer which is a matrix of shape (v + 1) d where v is vocabulary size (=5000) and d is the dimension of each word vector (=200). We initialize the embedding layer with random weights from N (0; 0:01). Each row of this embedding matrix represents represents the 200 dimensional word vector for a word in the vocabulary. For words in our vocabulary which match GloVe word vectors provided by the StanfordNLP group, we seed

the corresponding row of the embedding matrix from GloVe vectors. Each tweet i.e. its Dense Vector Representation (DVR) is padded with 0s at the end until its length matches to max_length which is a parameter we tweak in our experiments. We also conducted experiments using SGD+ Momentum weight updates and found out that it takes longer (100 epochs) to converge compared to validation accuracy equivalent to Adam. We ran our model upto 10 epochs. Using the Adam weight update scheme, the model converges very fast (4 epochs) and begins to overfit badly after that. We, therefore, use models from 3rd or 4th epoch for our results.

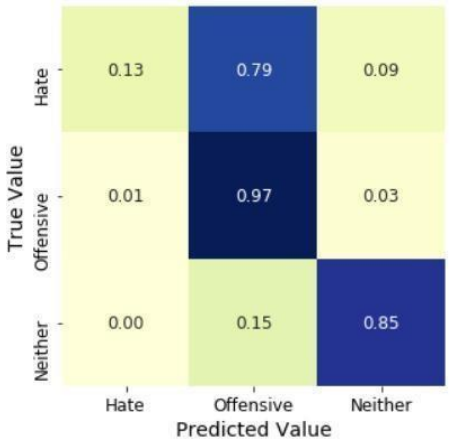


Fig.4. prediction ratio of the hate speech

5. Conclusion

A. Summary of achievements

The provided tweets were a mixture of words, emoticons, URLs, hastags, user mentions, and symbols. Before training the we pre-process the tweets to make it suitable for feeding into models. We implemented several machine learning algorithms like Naive Bayes, Maximum Entropy, Decision Tree, Random Forest, SVM, Recurrent Neural networks and Convolutional Neural Networks to classify the polarity of the tweet. We used two types of features namely unigrams and bigrams for classification and observes that augmenting the feature vector with bigrams improved the accuracy. Once the feature has been extracted it was represented as either a sparse vector or a dense vector. It has been observed that presence in the sparse vector representation recorded a better performance than frequency. Neural methods performed better than other classifiers in general. Our best LSTM model achieved an accuracy of 83.0% on Kaggle while the best CNN model achieved 83.34%. The model which used features from our best CNN model and classifies using SVM performed slightly better than only CNN. We finally used an ensemble method taking a majority vote over the predictions of 5 of our best models achieving an accuracy of 83.58%.

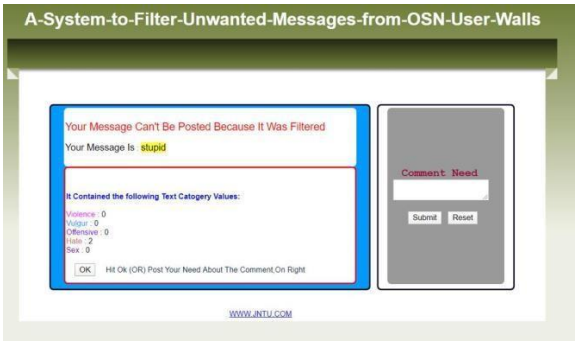


Fig. 5. The detection of hate speech

References

[1] Albert Biffet and Eibe Frank. Sentiment Knowledge Discovery in Twitter Streaming Data. Discovery Science, Lecture Notes in Computer Science, 2010, Volume 6332/2010, 1-15, DOI: 10.1007/978-3-642-16184-1_1

[2] Alec Go, Richa Bhayani and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Project Technical Report, Stanford University, 2009.

[3] Alexander Pak and Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of international conference on Language Resources and Evaluation (LREC), 2010.

[4] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner and Isabell M. Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about M. Young, The Technical Writer's Handbook. Mill Valley, CA: University

[8] Hatzivassiloglou, V, & McKeown, K.R. Predicting the semantic orientation of adjectives. In Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL, 2009.

[9] Johann Bollen, Alberto Pepe and Huina Mao. Modelling Public Mood and Emotion: Twitter Sentiment and socio- economic phenomena. In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2011.

[10] Luciano Barbosa and Junlan Feng. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In Proceedings of the international conference on Computational Linguistics (COLING), 2010.

[11] Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL), 2002.

[12] Rudy Prabowo and Mike Thelwall. Sentiment Analysis: A Combined Approach. Journal of Infometrics, Volume 3, Issue 2, April 2009, Pages 143-157, 2009.
Sentiment in Microblogs. In Proceedings of Empirical Methods on Natural Language Processing (EMNLP), 2011.

PROJECT OUTCOME

After successful completion of project work student will be able to:

TPO1: Analyze, Design and Implement projects with a comprehensive, systematic and ethical approach.

TPO2: Apply modern tools to execute and integrate modules in the project.

TPO3: Apply techniques for societal, health care, and real time sustainable research projects

TPO4: Develop communication skills by the technical presentation activities.

TPO5: Contribute as a team and lead the team in managing technical projects.

Mapping of Technical Project Outcomes with the Project titled "PROTECTED BANKING VERIFICATION SYSTEM USING PIXEL COMPARISON".

	TP1	**TP2**	**TP3**	**TP4**	**TP5**
DETECTION OF HATE SPEECH AND OFFENSIVE LANGUAGE IN TWITTER USING SENTIMENT ANALYSIS	3	3	2	3	3

(Indicate as 1 – Less than 30%; 2 – 30.1 -60%; 3 – Above 60.1%)