

B.E. PROJECT ON

**SOCIAL MEDIA ANALYSIS USING DEEP LEARNING
AND NATURAL LANGUAGE PROCESSING
TECHNIQUES**

Submitted By

Samarth Sehgal 343/CO/15
Shantanu Agrawal 354/CO/15
Shikhar Pahadia 359/CO/15

Under the Guidance of
Dr. Sushama Nagpal

A Project in partial fulfillment of requirement for the award of
B.E. in
Computer Engineering



**Department of Computer Engineering,
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
UNIVERSITY OF DELHI, DELHI
NEW DELHI-110078
2019**

DECLARATION

This is to certify that the work which is being hereby presented by us in this project titled “**Social Media Analysis Using Deep Learning and Natural Language Processing Techniques**” in partial fulfilment of the award of the Bachelor of Engineering submitted at the Department of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi, is a genuine account of our work carried out during the academic year 2018-2019 under the guidance of **Dr. Sushama Nagpal**, Department of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi.

The matter embodied in the project report, to the best of our knowledge, has not been submitted for the award of any other degree elsewhere.

Dated: 28 May 2019

Samarth Sehgal

343/CO/15

Shantanu Agrawal

354/CO/15

Shikhar Pahadia

359/CO/15

ACKNOWLEDGEMENTS

We would like to take this opportunity to acknowledge the support of all those without whom the project wouldn't have been possible.

We would like to express our deep gratitude towards our mentor Dr. Sushama Nagpal, Professor, Department of Computer Engineering, Netaji Subhas Institute of Technology, New Delhi for her constant support, guidance, constructive criticism, and encouragement which led to the completion of this work. The regular brainstorming sessions with her gave us a deep insight into the topic and evolved our ideas. We thank her for giving us this opportunity and supporting us throughout.

Our experience in working together has been wonderful. We hope that the knowledge, practical and theoretical, that we have gained through this B.E. Project will help us in our future endeavours in the field.

Last, but by no means the least, we would like to thank our parents who supported us through our thick and thin. Their trust in our capabilities helped us in giving our best.

We regret any inadvertent omissions.

CERTIFICATE

This is to certify that the report entitled “**Social Media Analysis Using Deep Learning and Natural Language Processing Techniques**” being submitted by **Samarth Sehgal (Roll No. 343CO15)**, **Shantanu Agrawal (Roll No. 354CO15)**, and **Shikhar Pahadia (Roll No. 359CO15)** to the Department of Computer Engineering, NSIT, for the award of bachelor’s degree of engineering, is the record of the bona fide work carried out by them under our supervision and guidance. The results contained in this report have not been submitted either in part or in full to any other university or institute for the award of any degree or diploma.

Supervisors

SUPERVISOR-1

Department of COE

ABSTRACT

Social Media can be both an outlet for an individual to express hateful ideas or a forum for the propagation of such hateful thoughts via online collective contagion. The increasing cases of hate crime and its high correlation with proliferation of misinformation and hateful ideas on social media warrants a deeper investigation into models for the detection of such hateful ideas in text such as tweets to enable prevention.

However, the complexity of the natural language constructs makes this task very challenging. Another challenge is to incorporate the ability to process tweets which are a mixture of Hindi and English languages. We begin by generating a data set comprising of text in Hindi and English languages as well as a mixture of the two. We collect this data from past papers in addition to manually scraping data on recent topics which have potential for Hate Speech use. These tweets are then manually annotated, after which the data is cleansed. GloVe Embedding is then used for obtaining vector representation of our data for further analysis. We proposed a novel Deep Learning approach for detecting hate speech in content on Twitter. A set of features is proposed for training both linear and ensemble classifiers over a dataset of manually annotated tweets. The performance of the proposed methodology is compared against previous approaches which used Machine Learning to analyse hate speech in social media text, namely logistic regression, which could achieve a maximum accuracy of 70%. The results are finally summarized to identify the accuracy of Hate Speech Detection using a CNN based Deep Learning architecture. An accuracy 88.79%, is obtained using this classifier with the proposed set of features, that is higher in comparison to the other models as well as the baselines with a data set that comprises a mix of Hindi and English Language text. Deep Learning architectures such as LSTMs, CNNs, and SVMs show promise in sentence level classification problems, however previous models haven't been able to breach the 77% accuracy mark. This work investigates the ability of deep learning architectures to build an accurate and robust model for Hate Speech detection and compares their performance with standard baselines in text classification problems. The experimental results reveal the merit in optimised CNN based models as compared to other deep learning and machine learning based classification models for hate speech detection.

LIST OF TABLES

Table Title	Page No.
Table 1 : Classification Report for ML models	35
Table 2 : Accuracy Report for ML models	36
Table 3 : Confusion Matrix - Logistic Regression	38
Table 4 : Confusion Matrix - Random Forest	38
Table 5 : Confusion Matrix - Gradient Boosting	39
Table 6 : Confusion Matrix - XGBoost	39
Table 7 : Accuracy Comparison for Different Models	40
Table 8 : Classification Report - CNN model	42
Table 9 : Confusion Matrix - CNN model	42
Table 10 : Evaluation Results - Naseeruddin Case	43
Table 11 : Evaluation Results - Assembly Elections	44
Table 12 : Evaluation Results - Brexit	45
Table 13 : Evaluation Results - Pulwama Attacks	46

LIST OF FIGURES

Figure Title	Page No.
Fig 1 : Internet User Data	1
Fig 2 : Internet Users India: Urban-Rural Breakup	2
Fig 3 : Increase in Facebook User-base	3
Fig 4 : Basic Architecture of NLP System	7
Fig 5 : Supervised vs Unsupervised Learning	9
Fig 6 : Supervised Machine Learning pipeline that is used to create a successful classifier	9
Fig 7 : Sample Machine Learning Algorithms	10
Fig 8 : Example of CNN filter and Polling Architecture of NLP (Source: Ronan Collobert, et al, 2014)	12
Fig 9 : Convolutional Neural Network Architecture for Sentence Classification (Source: "A Sensitivity Analysis of Convolutional Neural Networks for Sentence Classification“, 2015)	13
Fig 10 : Word Cloud for data set comprising English tweets	17
Fig 11 : Stages of Data Preprocessing	22
Fig 12 : Feature vectors 3 dimensional representation	23
Fig 13 : Code Snippet for GloVe embedding	24
Fig 14 : Ensemble Classifiers	26
Fig 15 : The workflow of research methods	27
Fig 16 : Random Forest Classifier	28
Fig 17 : Gradient Boosting in Decision Trees	28
Fig 18: Artificial Neural Network	30
Fig 19 : Neural network with many convolutional layers	31
Fig 20 : Stride of two pixels	32

Fig 21 : Max Pooling	32
Fig 22 : Word Vectors	36
Fig 23 : Training CNN model	41
Fig 24 : Training Curve - CNN	41
Fig 25 : Evaluation Results - Naseeruddin Case	44
Fig 26 : Evaluation Results - Assembly Elections	45
Fig 27 : Evaluation Results - Brexit	46
Fig 28 : Evaluation Results - Pulwama Attacks	47
Fig 29 : Accuracy Comparison of Different Models	49
Fig 30 : Analysis of Twitter Data	50
Fig 31 : CNN code snippet showing different layers	53
Fig 32 : Logistic Regression Code Snippet	54
Fig 33 : XGBoost Classifier Code Snippet	55
Fig 34 : Random Forest Classifier Code Snippet	56

TABLE OF CONTENTS

1. INTRODUCTION.....	1
1.1 Problem Statement.....	1
1.2 Background.....	1
1.3 Motivation.....	2
1.4 Advantages and Challenges.....	3
2. REVIEW OF LITERATURE.....	5
3. CONCEPTS AND TECHNOLOGIES USED.....	7
3.1 Natural Language Processing.....	7
3.2 Machine Learning.....	8
3.4 Deep Learning.....	10
3.4.1 Feature Extraction	
3.4.2 Word Embeddings and CNN for Text Classification	
3.4.3 Using CNN Hyperparameters	
3.5 Technology Stack.....	13
3.5.1 Scripting Languages	
3.5.2 Python	
3.5.3 Google Colab	
3.5.4 Keras	
3.5.5 TensorFlow	
3.5.6 NumPy	
4. DATA.....	17
4.1 Data Collection.....	17
4.2 Data Annotation.....	19
5. METHODOLOGY.....	21
5.1 Preprocessing.....	7
5.2 Feature Extraction.....	8
5.3 Machine Learning Techniques	10
5.3.1 Logistic Regression	
5.3.2 Ensemble Classifiers	
5.3.2.1 Random Forest	
5.3.2.2 Gradient Boosting	

5.3.2.3 XGBoost	
5.4 Deep Learning Techniques	10
5.4.1 Convolution Neural Networks	
5.4.1.1 Stride	
5.4.1.2 Pooling Layer	
5.4.1.3 Activation Function - ReLU	
5.4.1.4 Optimizer - Adam	
5.4.1.5 Loss function - Categorical Crossentropy	
6. RESULTS AND DISCUSSION.....	17
6.1 Analysis of Machine Learning Model.....	17
6.1.1 Accuracy, Precision, Recall and F1-Score	
6.1.2 Confusion Matrix	
6.1.2.1 Logistic Regression	
6.1.2.2 Random Forest	
6.1.2.3 Gradient Boosting Tree	
6.1.2.4 XGBoost	
6.2 Analysis of Deep Learning Model.....	19
6.2.1 Comparison with other models	
6.2.2 Accuracy with different number of epochs	
6.2.3 Recall, Precision and F1-Score	
6.2.4 Confusion Matrix	
6.3 Evaluation of Tweets.....	19
6.3.1 The Naseeruddin Shah Case	
6.3.2 Assembly Elections 2018	
6.3.3 Brexit	
6.3.4 Pulwama Attacks	
7. CONCLUSION AND FUTURE WORK.....	17
REFERENCES.....	17
APPENDIX A.....	17

1. Introduction

1.1 Problem Statement

This project aims to serve the following objectives:

1. Extracting data from various sources featuring use of Hateful Language.
2. Annotating this data manually to differentiate between inoffensive, offensive and hateful content.
3. To expand the ability of our model to process text featuring a mix of Hindi and English languages.
4. Modelling a Convolutional Neural Network, that can identify the nature of language in a given message.
5. Comparing our results with those achieved by using other analysis techniques such as Logistic Regression and XGBoost.

1.2 Background

With advances in information and communication technologies, there has been a boom in the number of internet subscribers. This boom in user base part can be attributed to the ease in accessibility coupled with a decline in data subscription rates.



Fig.1 Internet User Data

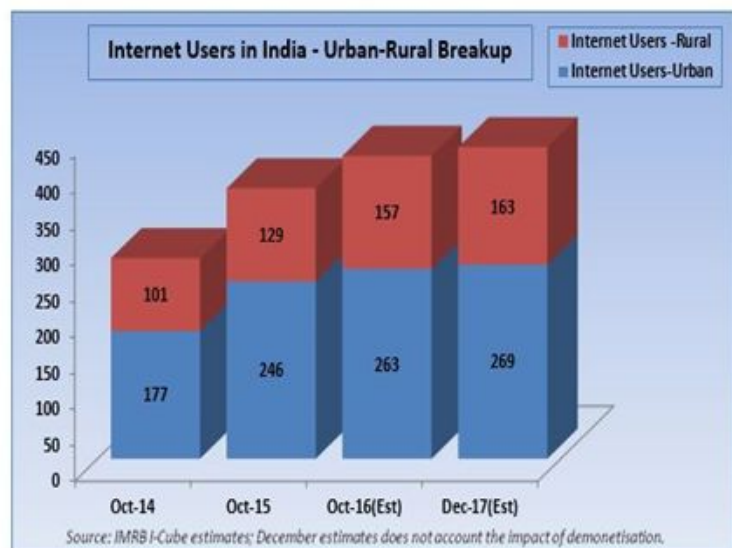
To put things in perspective, out of an estimated 7.7 billion world population, more than 4 billion people are today connected to the internet. As a result, internet has proved to be the primary channel for communication in the world today. This comprises of both personal as well as business related communication.

The interesting trend to be observed is that this rise in userbase has not been limited to Developed countries or the urban centres of Developing countries such as India. Its utility and accessibility have encouraged Internet services to penetrate to the rural grass-root, thus making the internet a universal entity rather than one with limited reach.

Fig.2 Internet Users India: Urban-Rural Breakup

With almost 3.2 billion active users, social media has emerged as a popular means of communication between people. This number of active users has rapidly grown in the past with the increase in internet subscriptions. An average person spends 15% of his/her time online

using social media and messaging services provided by the internet. This translates to nearly 1.5 hours daily using social media services.



1.3 Motivation

With such a huge userbase and a substantial amount of time being spent by each active social media user, it is natural that the volume of data being exchanged will be extremely large.

While this massive increase in data being transferred presents huge possibilities in the areas of open communication and encourages the free exchange of ideas it also poses a few very serious problems. The large audience and the practically unrestricted nature of social media make it prone to misuse. One such major area of concern is the use of social media to proliferate hate

against certain vulnerable sections of society.

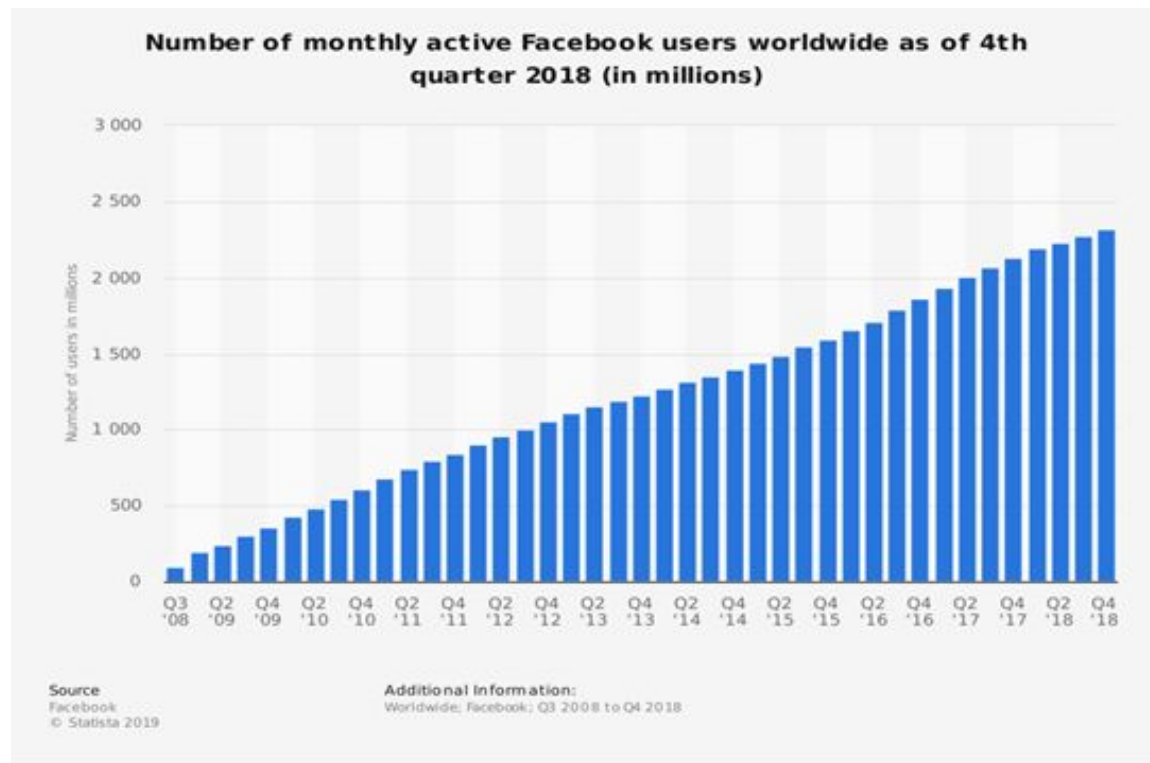


Fig.3 Increase in Facebook User-base

With the high volume of data being broadcasted unsupervised over social media sites, it has become virtually impossible to monitor this data for such hateful content. Such hateful content has potential to incite people in the real world and in turn leads to harassment of vulnerable groups. Such hateful discourse stems from deep rooted prejudice in the author of the message and targets a certain group of people based on their race, religion caste or sex. Further such content has the potential to spread misinformation in the public domain in order to vilify certain sections of society.

While there is a developing body of literature on the topic of identifying patterns in the language used on social media that expresses hateful ideation (De Choudhury et al., 2016), very few attempts have been made to employ feature extraction methods for binary classifiers that separate hateful content from offensive text.

1.4 Advantages and Challenges

- Considering the updated privacy terms of social media sites, finding the right method to extract large quantities of past data was a tedious task.
- Generating a strong dataset and its consequent annotation was a laborious process.

- Being a new brimming field, quite a lot of research and development analysis was involved.
- Understanding Convolutional Neural Networks, which do not appeal to us as a conventional operation and building upon them to perform natural language processing required great efforts.

2. Review of Literature

Previous works in the domain of Hate speech analysis have aimed to identify the targets of hate speech and various forms of online hate speech [1] and tried addressing the problem using Text classification algorithms based on Support Vector Machines (SVM) and a particular Recurrent Neural Network named Long Short Term Memory (LSTM) in order to verify their classification performances [2]. These approaches constitute the seminal literature in this subject and form the basis of our research into more advanced detection techniques. Deep-Learning based techniques such as Convolutional Neural Networks have also been utilised for classifying hate speech [3]. Logistic Regression character n-grams and NLP techniques when used [4] also managed to garner a 74% accuracy in results.

A systematic review concluded that internet may be used as a tool for furthering Hateful and Divisive ideologies. However, not all language constructs containing the word the use of foul language indicate hateful intent, specific semantic constructs may be used for predicting whether a sentence implies offensive language or hateful intent. An analysis method for automating identification of offensive speech was built using binary support vector machine classifiers.

Given the scope for research on Hate Speech and sentiment analysis, and the relatively low accuracy levels inspire us to investigate efficient methods to better understand and identify Hate Speech on Social Networking Sites. The reciprocal connectivity between authors of hateful content suggested a ripple effect in tightly-coupled virtual communities thereby concluding that twitter is an effective source for spreading hate and violence.

The field being a dynamic area with new trends and ways of expression being continuously incorporated into daily language we have tried to diversify our data set to include recent terminology and understand local languages (i.e. Hindi in our case) to make our analysis more robust. Further our aim has been to utilise GloVe Embedding which has not been used in this domain previously and has garnered highly accurate results in other NLP classification tasks. We wish to perform a comparative study to determine the optimal technique for the task.

3. Concepts and Technologies

3.1 Natural Language Processing

With large amounts of data available with organisations today, it has become an important task to be able automatically analyse understand and manipulate this data.

Natural Language Processing (NLP) is process of performing this task on data which is represented in the form of natural language text or speech.

NLP researchers aim to develop appropriate tools and techniques to make computer systems perform these tasks based on knowledge on how human beings understand and use language

The foundations of NLP can be traced back to several fields viz. computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, psychology, etc. Applications of NLP are widespread and it has been used in various fields, such as machine translation, natural language text processing and summarization, user interfaces, multilingual and cross language information retrieval (CLIR), speech recognition, artificial intelligence and expert systems, etc.

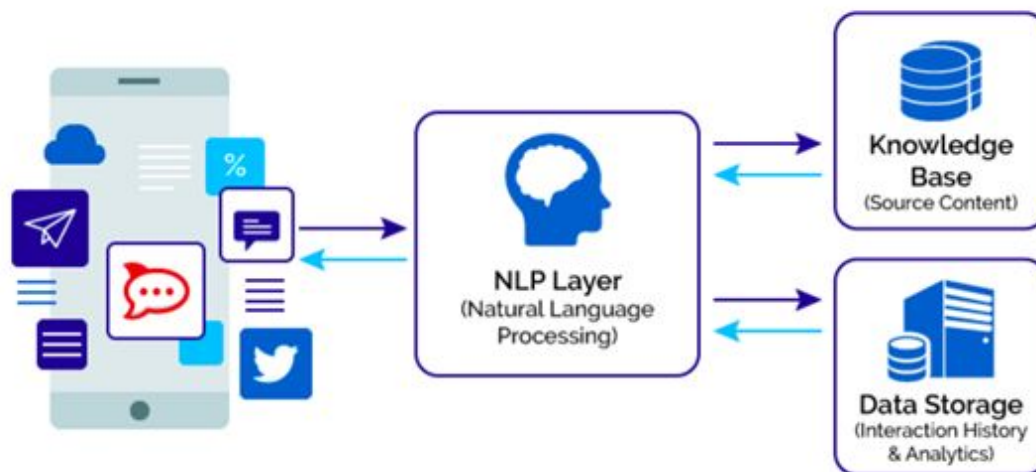


Fig.4 Basic Architecture of NLP System

Natural language understanding forms the core of any NLP task. Three major problems are encountered while building computer programs that understand natural language, these problems relate to:

- The thought process;
- The representation and meaning of the linguistic input;
- The world knowledge.

Thus, an NLP system may begin at the word level \tilde{n} to determine the morphological structure, nature (such as part-of speech, meaning) etc. of the word \tilde{n} and then may move on to the sentence level to determine the word order, grammar, meaning of the entire sentence, etc. and then to the context and the overall environment or domain.

The specific meaning or connotation of a given word or a sentence depends on the given context or domain. The given word in this scenario can be related to many other words and/or sentences in the given context.

In order to understand natural languages, a natural language processing system may involve all or some of the following seven interdependent levels of analysis, that people use to extract meaning from text or spoken languages:

- phonetic or phonological level that deals with pronunciation
- morphological level that deals with the smallest parts of words, that carry a meaning, and suffixes and prefixes
- lexical level that deals with lexical meaning of words and parts of speech analyses
- syntactic level that deals with grammar and structure of sentences
- discourse level that deals with the structure of different kinds of text using document structures and
- pragmatic level that deals with the knowledge that comes from the outside world, i.e., from outside the contents of the document.

3.2 Machine Learning

Supervised machine learning aims to find algorithms that utilise external data to produce a general hypothesis, and utilise these algorithms to make predictions regarding future occurrences. In other words, we use supervised learning to build a concise model which

represents the distribution of class labels in terms of predictor features. The classifier hence generated is then used to assign class labels to the test data set based on the value of their predictor features.

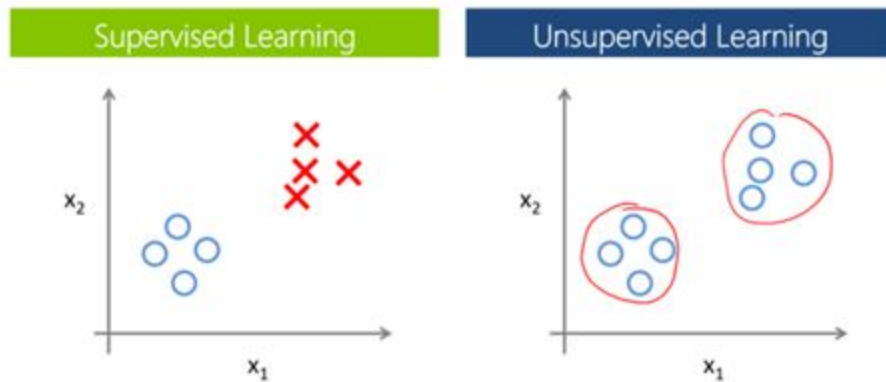


Fig.5 Supervised vs Unsupervised Learning

Machine Learning (ML) has multiple applications and a significant one of those is Data Mining. Humans are prone to making mistakes during analysis while identifying relationships between multiple features thereby making problem solving rather difficult. In such problems Machine Learning can be applied successfully, leading to improved efficiency of systems and machine designs.

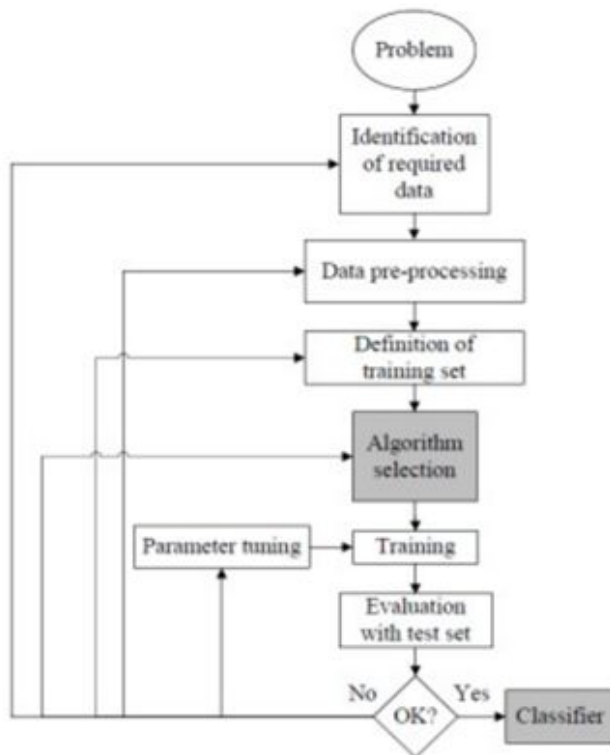


Fig.6 Supervised Machine Learning pipeline that is used to create a successful classifier

Each instance of a dataset used by machine learning algorithms is represented using the same set of features. These features can either be continuous, categorical or binary. The instances of data used for training the model may be labelled or unlabelled. In the case of supervised learning, instances are given with known labels (the corresponding correct outputs). However, in unsupervised learning, the instances are unlabelled with researchers trying to discover unknown, but useful, classes of items (Jain et al., 1999) by applying these unsupervised (clustering) algorithms.

A third type of machine learning technique is reinforcement learning (Barto & Sutton, 1997). The training information provided to the learning system by the environment (external trainer) is in the form of a scalar reinforcement signal that constitutes a measure of how well the system operates. Rather than telling the learner which actions to take, it is made to discover which actions must be taken in order to yield the best reward. This is done by in turn trying each possible action.

	Unsupervised	Supervised
Continuous	<ul style="list-style-type: none"> • Clustering & Dimensionality Reduction <ul style="list-style-type: none"> ◦ SVD ◦ PCA ◦ K-means 	<ul style="list-style-type: none"> • Regression <ul style="list-style-type: none"> ◦ Linear ◦ Polynomial • Decision Trees • Random Forests
Categorical	<ul style="list-style-type: none"> • Association Analysis <ul style="list-style-type: none"> ◦ Apriori ◦ FP-Growth • Hidden Markov Model 	<ul style="list-style-type: none"> • Classification <ul style="list-style-type: none"> ◦ KNN ◦ Trees ◦ Logistic Regression ◦ Naive-Bayes ◦ SVM

Fig.7 Sample Machine Learning Algorithms

3.3 Deep Learning

Deep learning in simple terms can be thought of as a field of Artificial Intelligence which automates predictive analysis by emulating the learning approach used by humans to gain certain specific types of knowledge. In contrast to traditional machine learning algorithms which are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

3.3.1 Feature Extraction

Each algorithm in the hierarchy applies a nonlinear transformation on its input and uses what it learns to create a statistical model as output. Iterations continue until the output has reached an acceptable level of accuracy. The number of processing layers through which data must pass is what inspired the label deep. In traditional machine learning, the learning process is supervised and the programmer has to be very, very specific when telling the computer what types of things it should be looking for when deciding if an image contains a dog or does not contain a dog.

This is a laborious process called feature extraction and the computer's success rate depends entirely upon the programmer's ability to accurately define a feature set for the problem. The advantage of deep learning is that the program builds the feature set by itself without supervision. Unsupervised learning is not only faster, but it is usually more accurate. The program uses the information it receives from the training data to create a feature set for problem and build a predictive model. Of course, the program is not aware of the labels; it will simply look for patterns of pixels in the digital data. With each iteration, the predictive model the computer creates becomes more complex and more accurate.

3.3.2 Word Embeddings and CNN for Text Classification

Text Classification involves the use of a word embedding for representing words and a Convolutional Neural Network (CNN) for learning how to discriminate documents on classification problems. When used with pre-trained word embeddings particularly, neural networks in general offer better performance than classical linear classifiers.

The architecture is therefore comprised of three key pieces:

- **Word Embedding:** A distributed representation of words where different words which have a similar meaning (based on their usage) also have a similar representation.
- **Convolutional Model:** A feature extraction model that learns to extract salient features from documents represented using a word embedding.
- **Fully Connected Model:** The interpretation of extracted features in terms of a predictive output.

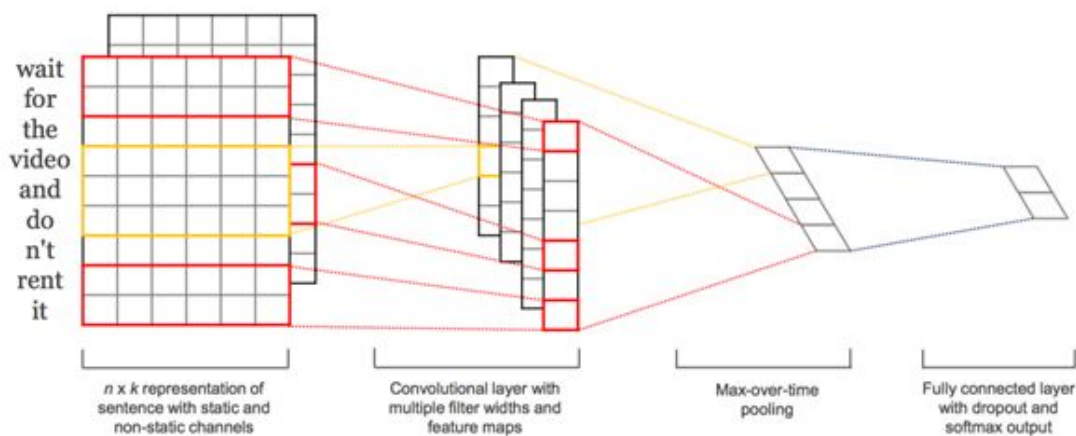


Fig.8 Example of CNN filter and Pooling Architecture of NLP (Source: Ronan Collobert, et al, 2014)

3.3.3 Using CNN Hyperparameters

Some hyperparameters matter more than others when tuning a convolutional neural network on a text classification problem.

Ye Zhang and Byron Wallace performed a sensitivity analysis into the hyperparameters needed to configure a single layer convolutional neural network for document classification. The study is motivated by their claim that the models are sensitive to their configuration. Their aim was to provide general configurations that can be used for configuring CNNs on new text classification tasks. They provide a nice depiction of the model architecture and the decision points for configuring the model.

The general findings were as follows:

- The choice of pre-trained word2vec and GloVe embeddings differ from problem to problem, and both performed better than using one-hot encoded word vectors.
- The size of the kernel is important and should be tuned for each problem.
- The number of feature maps is also important and should be tuned.
- The 1-max pooling generally outperformed other types of pooling.
- Dropout has little effect on the model performance.

They go on to provide more specific heuristics, as follows:

- Use word2vec or GloVe word embeddings as a starting point and tune them while fitting the model.
- Grid search across different kernel sizes to find the optimal configuration for your problem, in the range 1-10.

- Search the number of filters from 100-600 and explore a dropout of 0.0-0.5 as part of the same search.
- Explore using tanh, ReLU, and linear activation functions.

The depiction of the model architecture and the decision points for configuring the model is reproduced below:

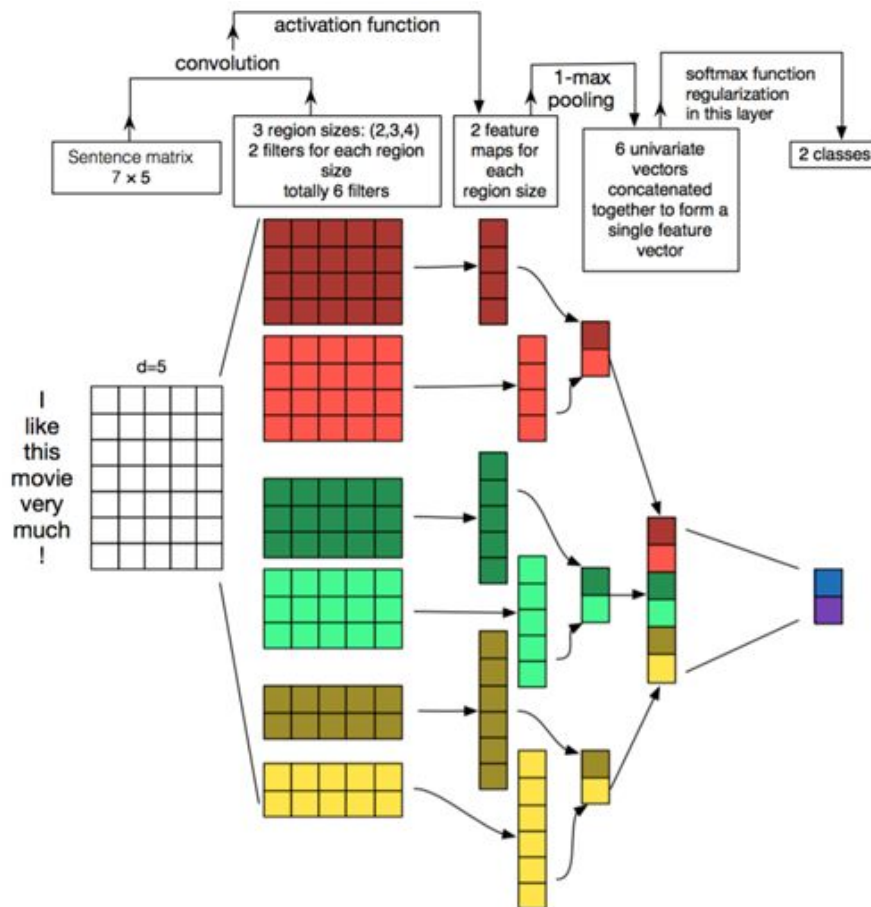


Fig.9 Convolutional Neural Network Architecture for Sentence Classification (Source: "A Sensitivity Analysis of Convolutional Neural Networks for Sentence Classification", 2015)

Better performance can be achieved with very deep convolutional neural networks. However in practice, standard and reusable architectures have not been adopted for classification tasks. Alexis Conneau, et al. comment on the relatively shallow networks used for natural language processing and the success of much deeper networks used for computer vision applications. Older architectures used for natural language are limited to 5 and 6 layers. These are contrasted with successful architectures used in computer vision with 19 or even up to 152 layers.

3.4 Technology Stack

3.4.1 Scripting Languages

A script or scripting language is a programming language that supports scripts. Scripts are programs written for a special run-time environment and are used to automate the execution of tasks which would otherwise have to be executed by a human operator one by one. Scripting languages are usually interpreted instead of being compiled.

Python is the primary language used for machine learning. It is not the fastest. Scala, Julia is faster. Also, most of the heavy lifting in Python are actually done by C or Fortran libraries backend. Neither is it the easiest to learn. R is easier for beginners. Rather, it is a general language that does a little of everything at a good enough complexity-performance trade-off along with a full suite of tools for machine learning applications.

3.4.2 Python

Python programming language is one of the top most languages used in implementing Machine Learning projects. It is broadly used all over the industries and allows easy association within development teams. Presently, most of the companies like Google, Facebook or Microsoft are choosing Python, and its various features such as simple, approachable, versatile, and complete make it choice of every developer.

Nowadays, Python programming language is gaining more popularity in Machine Learning projects due to its various features. It is a high-level, general purpose and dynamic programming language which is not new in the market. It is available for almost thirty years. Python programming language can be found practically at everyplace, like web and desktop apps, machine learning, network servers and many more. It is mainly used in small project development, but now the big firms such as Google, Facebook, Microsoft, and Netflix are also utilizing Python in implementing their projects. It is one of the fastest growing programming languages, and it is expected that Python will take over other languages such as JAVA in the upcoming year. Python programming language has several advantages which enable Machine Learning developers to use it for developing their Machine Learning projects.

3.4.3 Google Colab

Colab is an executable document that lets you write run and share code within google drive. Colab is a Jupyter Notebook stored in Google Drive. A notebook document is composed of cells each of which can contain code text images and more. Colab connects your notebook to cloud based runtime i.e. you can execute python code without any required setup on your own machine. It leads to a rich interactive coding experience allowing you to use any functionality that Python has to offer. It also allows you to enable GPU backend for your notebook.

3.4.4 Keras

Keras is a high-level deep learning library written in Python. Keras is effectively an interface which wraps multiple frameworks. It can be used as an interface for TensorFlow, Theano or CNTK. It works the same no matter what backend you use. It focuses on enabling fast experimentation. It further:

When it comes to quickly training and testing a model built from standard layers, Keras definitely serves as the fastest track. Using Keras the pipeline for building a deep network involves the following steps:

1. Define Network
2. Compile Network
3. Fit Network
4. Evaluate Network
5. Make Predictions

It is an extremely user-friendly API and gives primary importance to user experience.

3.4.5 TensorFlow

TensorFlow serves as a comprehensive platform to build and deploy Machine Learning models easily. It allows the user to build and train state-of-the-art models without sacrificing speed or performance. It is a free and open-source software library for dataflow and differentiable programming across a range of tasks.

It is a symbolic math library and is used for both research and production at Google which is widely used for machine learning applications such as neural networks. It thus serves as an ecosystem to solve challenging, real-world problems with machine learning.

TensorFlow offers multiple levels of abstraction and provides the functionality to build and train models by using the high-level Keras API. For large ML training tasks, there is a provision to use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition. At the same time TensorFlow allows for immediate iteration and intuitive debugging.

TensorFlow provides a direct path to production. TensorFlow allows the user to train and deploy their model on servers, edge devices, or the web, irrespective of the language or platform used.

3.4.6 NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

4. Data

4.1 Data Collection

For the purpose of our analysis we have built a strong data set comprising of 56 thousand tweets. The uniqueness of this data arises from the fact that it comprises of both Hindi as well as English textual data in addition to including a mix of the two languages wherein words of the Hindi language are written using English alphabets. E.g. “Aaj main free hoon.”

This data set was generated by combining twitter data accumulated from different research papers which focussed on Hate speech Detection, Analysis and Classification.

The tweets in the data set have already been annotated by the researchers for the purpose of their analysis and we use them as is for the purpose of our Analysis.

This data set of 56 thousand tweets is then divided into the training, validation, and testing sets so as to train our model and check its accuracy. The data set is divided as follows:

- **Training Set:** 60%
- **Validation Set:** 20%
- **Testing Set:** 20%



Figure 10 Word Cloud for data set comprising English tweets

In addition to the above data set comprising of 56 thousand tweets, another 30 thousand tweets were collected from microblogging website Twitter - specifically, on topics which had a good potential for Hate Speech Analysis.

We chose four relatively recent topics and extracted 7,500 tweets about each to run our predictive analysis. The topics chosen were as follows:

- **Pulwama Terror Attack;**
- **Brexit;**
- **Naseeruddin Shah Controversy;**
- **Assembly Elections 2018.**

These topics were chosen keeping in mind the potential for exchange of offensive language on social media.

Traditionally tweets have been extracted by Twitter REST API using the Beautiful Library. However due to recent privacy restrictions imposed by recently by social networking sites, there is now a limit of 7 days and also on the number of tweets that can be extracted using this method.

We thus had to use a self-designed Python Script to perform the tweet extraction. This script takes input as the keyword to be searched, the since and until dates to specify the dates within which the tweets had to be searched and gives the corresponding output.

Attributes associated with a single tweet are URLs, text, user mentions, hashtags, media files (image, audio and video), timestamp, number of retweets, likes etc. and the user information. The main focus was on text mining as storing media files would, firstly, need a lot of space and secondly, analysing media files for suicidal content would require additional machine learning.

For the purpose of searching relevant tweets related to each of the above-mentioned issues, we used the relevant hashtags related to the events which were trending at the time the event took place.

The texts were collected without knowing the sentiment. For example, when collecting tweets on hashtag #AssemblyElections2018, it is not known initially whether:

- the tweet is posted for spreading hate;
- the person is intending to stoke violence for electoral gains;
- the tweet reports a third person's use of Hate Speech e.g.: news report;
- the tweet uses Hate in a casual context e.g.: I hate Mondays.

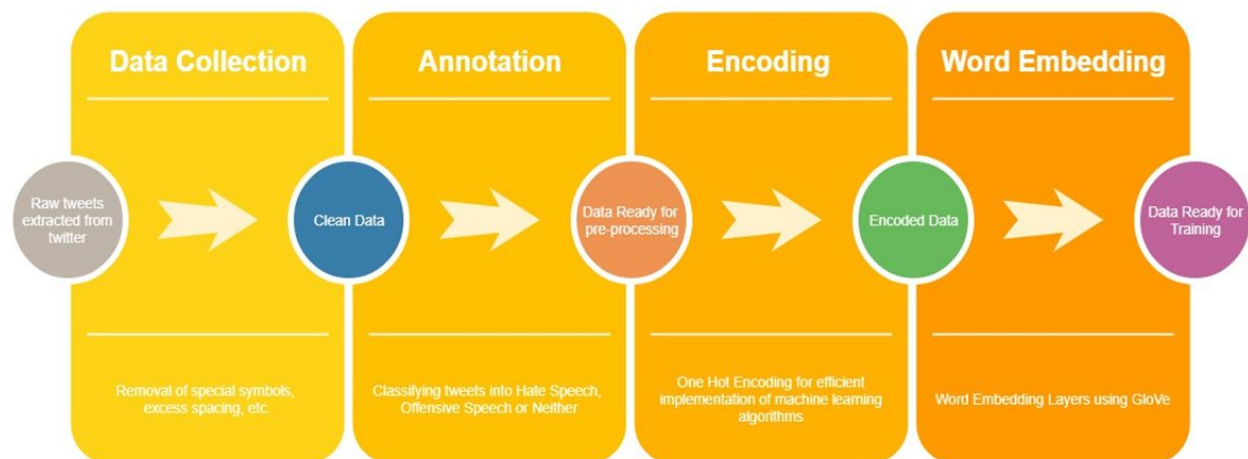
5. Methodology

The overall methodology is divided into four phases. The initial phase consists of Data extraction, second phase involves preprocessing the text within a tweet, the third phase is the feature extraction from preprocessed tweets for the training and testing, and the final phase actually classifies and identifies tweets as hateful, offensive or none. The details of these individual phases are presented below:

5.1 Preprocessing

Preprocessing is not just one single task, but a combination of many tasks. We achieved this by application of a series of steps in the order given below for the processing of raw tweets.

- We removed non-English tweets using Kutools for Excel.
- To increase diversity in learned patterns & reduce the bias from heavy twitter users, tweets in the training set from twitter users with more than a thousand people following them were dropped.
- We cleansed the tweets by removing the URLs.
- We converted all the tweets from uppercase characters to lowercase characters.
- We designed a script to shorten the words with 3 or more repetitive letters. These repetitions are intentional e.g., hhhhiiii to hi.
- Duplicate tweets tend to create a bias on the model results so, We removed the duplicacy in the tweets via a python script.
- We condensed the words for via the same script used before. For example, yyeess will be reduced to yes, while committee remains intact.
- We cleansed the dataset from all stop words.



5.2 Feature Extraction

Machine learning algorithms are unable to work with categorical data directly. Categorical data must be converted to integers or binary numbers. This applies when you are working with a sequence classification type problem intend to use deep learning methods such as LSTM recurrent neural networks, Convolution neural networks etc. Categorical data are variables that contain label values such as text rather than numeric values. Machine learning algorithms cannot give efficient results on labelled data. So we need to convert all input and output variables to numeric form for application of algorithms. For this, we have used one hot encoding in our project to convert the categorical data to binary vectors.

Traditional feature engineering strategies for textual data involve models belonging to a family of models popularly known as the Bag of Words model. This includes N-grams, TF-IDF (term frequency-inverse document frequency), term frequencies and so on. While they are effective methods for feature extraction from texts, due to the inherent nature of the model being just a bag of unstructured words, we lose additional information like the semantics, structure, sequence and context around nearby words in each text document. This motivated us to explore more sophisticated models which can capture this information and give us features which are vector representation of words, popularly known as embeddings.

For this we used a more advanced feature engineering strategies which often leverage deep learning models.

GloVe

With the help of GloVe we can create 3 dimensional feature vectors for the words of interest. This helps us give a broader picture of the similarity between different words. Words having similarity tend to cluster together in the 3D space. GloVe uses co-occurrence matrix for training of words.

Below figure will help understand GloVe better. We applied GloVe to the following words - niece, aunt, sister, nephew, uncle, brother, women, man, heir, madam, heiress, sir, king, emperor, duke, queen, earl, empress, duchess & countess. We obtained the results which are very easy to understand. Now the words which are similar to each other like aunt, niece, sister and nephew uncle, brother formed clusters with each other. Because the former represents females and the latter represents male. So the clusters of all females are kept together and similarly the clusters of male members is kept together.

Similarly, heiress, countess ,empress, etc represents females with some authority, some power. So these words are close to niece, aunt cluster because they all are females but a little away from them and far away from the male cluster. Same is the case with the king, emperor, etc cluster. The male cluster is below and the female cluster is above on the graph given below.

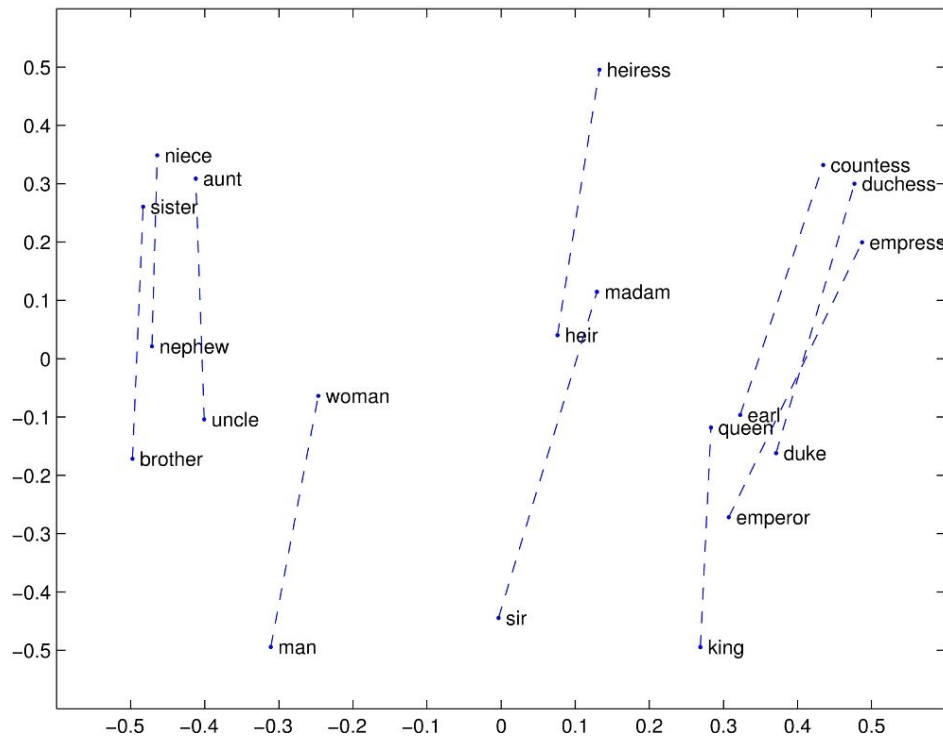


Figure 12 Feature vectors 3 dimensional representation

The code snippet of GloVe embedding used in our project is given below -

```
[ ] # prepare tokenizer
t = Tokenizer()
t.fit_on_texts(docs)
vocab_size = len(t.word_index) + 1
# integer encode the documents
encoded_docs = t.texts_to_sequences(docs)
print(encoded_docs)
# pad documents to a max length of 4 words
max_length = 4
padded_docs = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
print(padded_docs)
# load the whole embedding into memory
embeddings_index = dict()
f = open('glove.twitter.27B.100d.txt')
for line in f:
    values = line.split()
    word = values[0]
    coefs = asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
# create a weight matrix for words in training docs
embedding_matrix = zeros((vocab_size, 100))
for word, i in t.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
# define model
```

```
↳ [[17406, 2, 184, 21, 356, 19, 148, 1, 75, 123], [7, 11140, 2, 21, 8483, 12,
[[ 148    1    75   123]
 [ 1504   20    15   139]
 [ 201 17408   295 17409]
...
 [ 268    51   826 1510]
 [ 150 1954    27   639]
 [  18     9    10 52252]]
Loaded 1193514 word vectors.
```

5.3 Machine Learning Techniques

Our Analysis of Social media is formulated as a supervised ternary classification problem. For every tweet $t_i \in D$, the document set, a variable $y_i \in \{0, 1, 2\}$ is introduced, where $y_i = 0$ denotes that the tweet is hateful, $y_i = 1$ denotes that the tweet is offensive & $y_i = 2$ denotes that the tweet is neither of the above.

Both XGBoost and Gradient Boosting Decision Trees aim to boost the performance of a classifier in a stage-wise fashion by iteratively adding a new classifier to the ensemble to allow the optimization of a differentiable loss function. The Random Forest classifier is one of the most popular ensemble machine learning algorithm based on Bootstrap Aggregation (Quinlan et al., 1996) or bagging. It modifies the bagging procedure so that the learning algorithm is limited to a random sample of features of which to search, which has shown promise in text classification problems. This section is divided into two subsections describing the different feature-extraction based classification algorithms used.

5.3.1 Logistic Regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, pregnant, etc.) or 0 (FALSE, failure, non-pregnant, etc.). The goal of logistic regression is to find the best fitting (yet biologically reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a logit transformation of the probability of presence of the characteristic of interest:

$$\text{logit}(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_kX_k$$

where p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds:

$$\text{odds} = \frac{p}{1-p} = \frac{\text{probability of presence of characteristic}}{\text{probability of absence of characteristic}}$$

And

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

We got the accuracy of about 56% using the Logistic Regression model.

5.3.2 Ensemble Classifiers

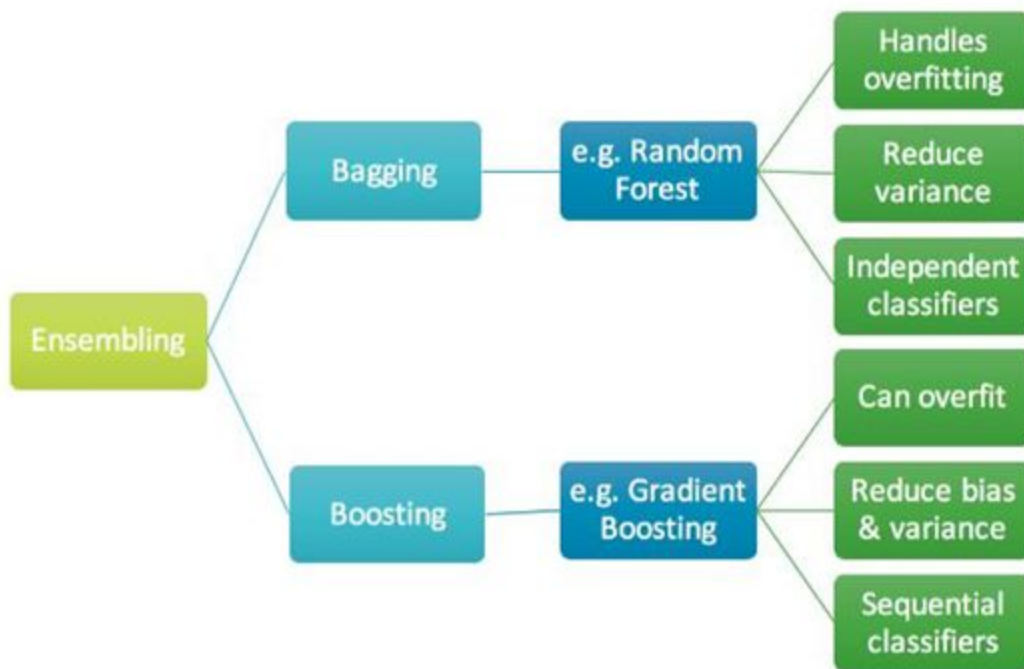


Figure 21 Ensemble Classifiers

An ensemble consists of a set of individually trained classifiers (such as Support Vector Machine and Classification Tree) whose predictions are combined by an algorithm. Ensemble methods is expected to improve the predictive performance of classifier.

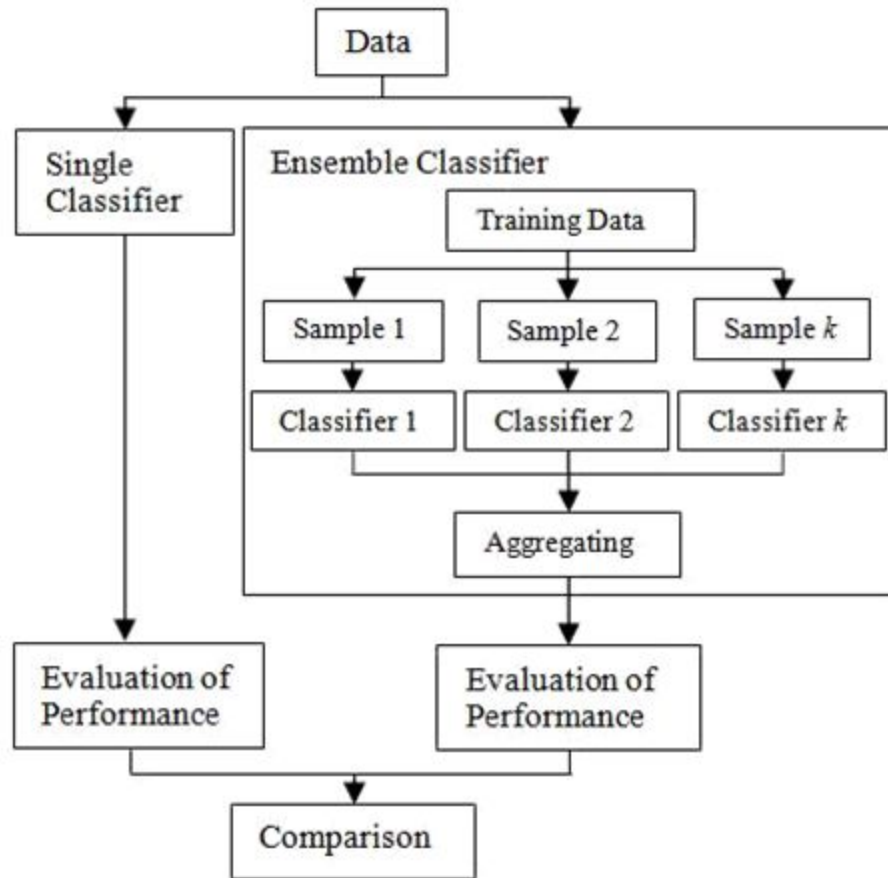


Figure 22 The workflow of research methods

Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

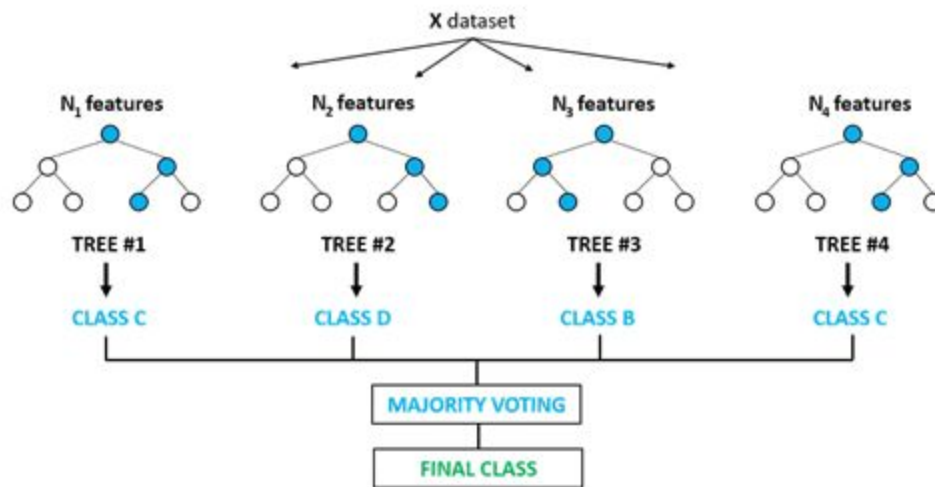


Figure 23 Random Forest Classifier

Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. Like other boosting methods, gradient boosting combines weak "learners" into a single strong learner in an iterative fashion.

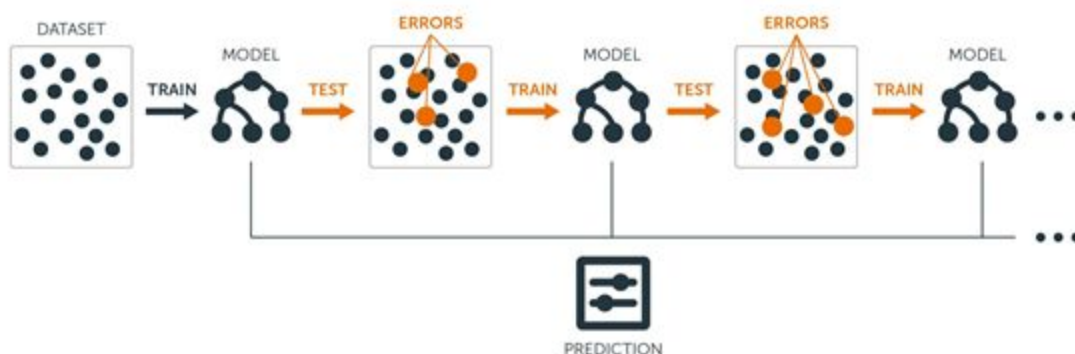


Figure 24 Gradient Boosting in Decision Trees

XGBoost

XGBoost is short for “Extreme Gradient Boosting”, where the term “Gradient Boosting” is proposed in the paper Greedy Function Approximation: A Gradient Boosting Machine, by Friedman. It is an implementation of gradient boosted decision trees designed for speed and performance. The name “XGBoost” refers to the engineering goal to push the limit of computations resources for boosted tree algorithms.

XGBoost is a software library that can be downloaded and installed and then accessed from a variety of interfaces. Specifically, XGBoost supports the following main interfaces:

- Command Line Interface (CLI).
- C++ (the language in which the library is written).
- Python interface as well as a model in scikit-learn.
- R interface as well as a model in the caret package.
- Julia.
- Java and JVM languages like Scala and platforms like Hadoop.

XGBoost Features

The library is laser focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features.

1. Model Features - The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported:
 - a. Gradient Boosting algorithm
 - b. Stochastic Gradient Boosting with sub-sampling
 - c. Regularized Gradient Boosting with both L1 and L2 regularization.
2. System Features - The library provides a system for use in a range of computing environments, not least:
 - a. Parallelization
 - b. Distributed Computing
 - c. Out-of-Core Computing
 - d. Cache Optimization

3. Algorithm Features - The implementation of the algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:

- a. Sparse Aware
- b. Block Structure
- c. Continued Training

We got the accuracy of about 66% using the Logistic Regression model.

5.4 Deep Learning Techniques

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Deep learning architectures include deep neural networks, deep belief networks and recurrent neural networks.

An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

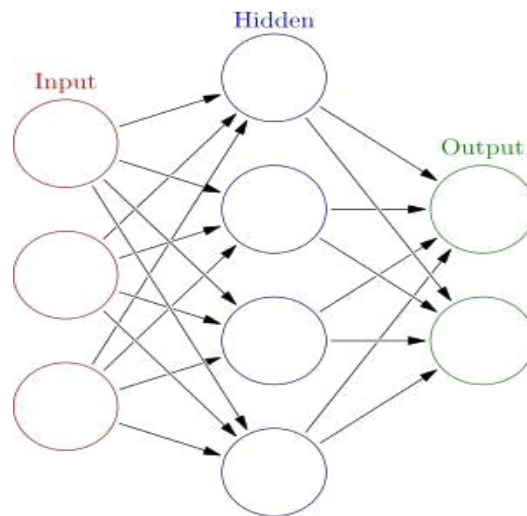


Figure 25 Artificial Neural Network

Convolution Neural Networks

We have use the CNN model in Keras with tensorflow for classification of our tweets data. A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks.

Like other kinds of artificial neural networks, a convolutional neural network has an input layer, an output layer and various hidden layers. Some of these layers are convolutional, using a mathematical model to pass on results to successive layers. This simulates some of the actions in the human visual cortex.

CNNs are a fundamental example of deep learning, where a more sophisticated model pushes the evolution of artificial intelligence by offering systems that simulate different types of biological human brain activity.

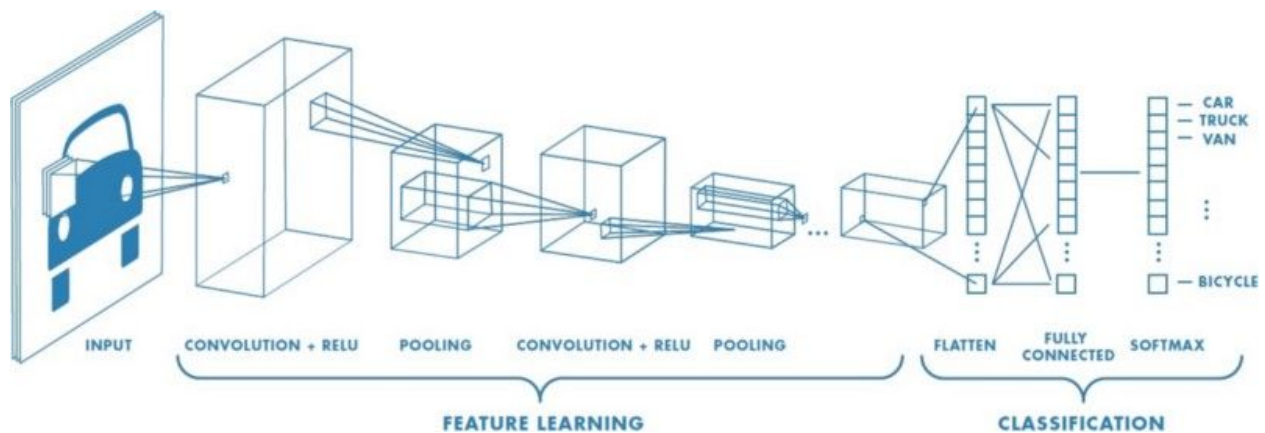
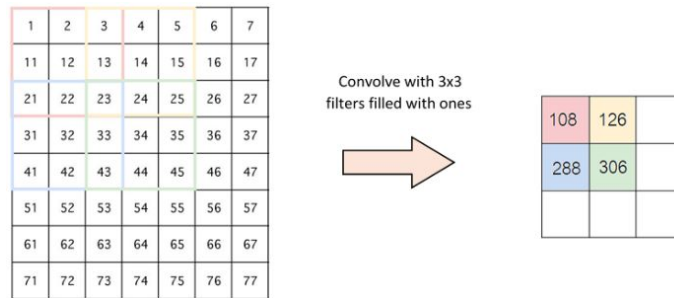


Figure 19 Neural network with many convolutional layers

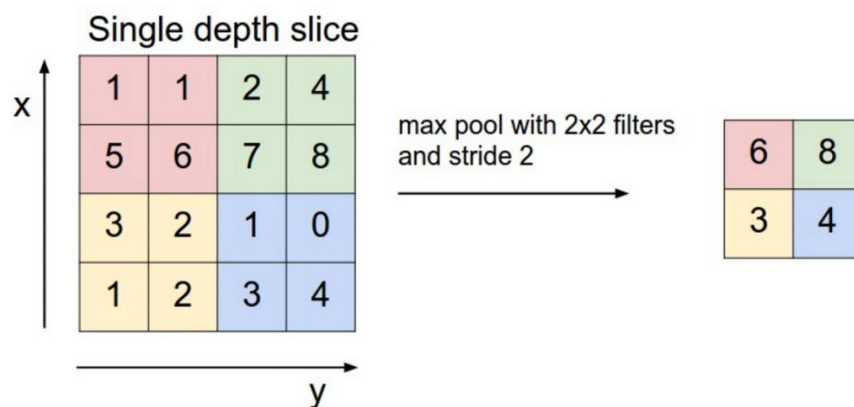
Stride

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.



Pooling Layer

A key aspect of Convolutional Neural Networks are *pooling layers*, typically applied after the convolutional layers. Pooling layers subsample their input. The most common way to do pooling is to apply a *max* operation to the result of each filter. You don't necessarily need to pool over the complete matrix, you could also pool over a window. For example, the following shows max pooling for a 2x2 window (in NLP we typically are apply pooling over the complete output, yielding just a single number for each filter):



Activation Function - ReLU

To improve the performance of the neural network for speech, we used Rectified Linear Unit (ReLU) as the activation function because it enables sparsity in hidden layer representation. ReLU also offers to disperse values of hidden unit activation probabilities thus ensuring that the information is distributed uniformly in the hidden layer. Dispersion measures whether the set of active units is different for each stimulus. It also enables the network to become invariant to input perturbations

Optimizer - Adam

We have used Adam Optimizer as it has relatively lower memory requirements and it usually works well even with a little tuning of hyperparameters.

Loss function - Categorical Crossentropy

We have used categorical crossentropy because we are dealing with multiclass classification in our project.

6. Results and Discussion

6.1 Analysis of Machine Learning Model

6.1.1 Accuracy, Precision, Recall and F1-Score

Table 1 presents the results for the Machine Learning models used proposed methodology in terms of three evaluation metrics: Precision, Recall and F1 Score. The models are trained after the tweets were embedded using GloVe embedding, which as described in the methodology proved out to give better results than applying other features.

Class	Model	Precision	Recall	F1-Score
0	Logistic Regression	0	0	0
	Random Forest	0.84	0.2	0.32
	Gradient Boosting Tree	0.75	0.1	0.18
	XGBoost	0.79	0.02	0.04
1	Logistic Regression	0.58	0.79	0.67
	Random Forest	0.71	0.9	0.8
	Gradient Boosting Tree	0.54	0.8	0.64
	XGBoost	0.63	0.86	0.73
2	Logistic Regression	0.49	0.4	0.44
	Random Forest	0.71	0.63	0.67
	Gradient Boosting Tree	0.65	0.43	0.52
	XGBoost	0.58	0.48	0.62

Here the class 0 represents the data containing offensive speech, class 1 represents hateful speech and class 2 represents clear speech. It can be observed that the Logistic Regression Model, Gradient Boosting Tree Model and XGBoost model do not perform as well as the Random Forest model owing to the lack of features used in these models that are not suitable for learning how to classify tweets with Hate Speech and Offensive speech. These models are more suitable in general domain, however, Random Forest fits in perfectly in the particular problem domain of Hate Speech Classification. Though the results of Random Forest Classification is also not as as pleasing as expected, but this model proves out to be the best among all other machine learning models, followed by XGBoost model which performs Xtreme Gradient Boosting to classify data as required. These results are also proved by the accuracy results as shown below:

Model	Accuracy
Logistic Regression	56.82
Random Forest	79.41
Gradient Boosting Tree	65.43
XGBoost	66.67

The accuracy in this table is represented in percent. As discussed, Random Forest Classification holds the maximum accuracy of 79.41% followed by 66.67% accuracy of XGBoost. The Logistic Regression model gave the least accuracy, as expected, since it is the most basic model for our problem domain, though despite its simplicity, it has better precision and recall in many cases, when compared to the boosting algorithms.

The drastic improvements from the baseline machine learning models is due to the inclusion of various features by GloVe embedding. This embedding proved out to be better than word2vector and other feature inclusions as observed in our research.

A short snippet of word vectors using GloVe embedding can be seen as below:

```
[[17406, 2, 184, 21, 356, 19, 148, 1, 75, 123], [7, 11140, 2, 21,
[[ 148      1      75    123]
[ 1504     20     15    139]
[  201 17408    295 17409]
...
[  268     51    826  1510]
[  150  1954     27    639]
[   18      9     10 52252]]
Loaded 1193514 word vectors.
```

The same word vectors were used in the deep learning models as described in the report later on.

Another way of representing our result can be through the confusion matrix. It shows the number of word vectors that were true positive, false positive, true negative and false negative, each of which is described as below.

5.1.2 Confusion Matrix

TP: Data that is predicted true and is actually true.

FP: Data that is predicted true, but is actually false.

TN: Data that is predicted false and is actually false.

FN: Data that is predicted false, but is actually true.

True Positive and True Negatives are considered to be accurate predictions. The four evaluation metrics described in the above section can be calculated using confusion matrix as below:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F - measure} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Similarly, this can also be extended to a 3-class classifier.

5.1.2.1 Confusion Matrix - Logistic Regression

True Class/ Predicted Class	0	1	2
0	0	530	142
1	0	2163	555
2	0	932	678

Here, the green coloured cells represent the correctly predicted class data while the uncoloured ones are wrongly predicted values. It can be observed that out of a total of 5000 data, only 2841 were predicted correctly.

$$\begin{aligned}\text{Hence accuracy} &= 2841/5000 * 100 \\ &= 56.82\%\end{aligned}$$

5.1.2.2 Confusion Matrix - Random Forest

True Class/ Predicted Class	0	1	2
0	133	257	134
1	17	2779	211
2	8	403	1058

Here, the green coloured cells represent the correctly predicted class data while the uncoloured ones are wrongly predicted values. It can be observed that out of a total of 5000 data, 3970 were predicted correctly.

$$\begin{aligned}\text{Hence accuracy} &= 3970/5000 * 100 \\ &= 79.41\%\end{aligned}$$

5.1.2.3 Confusion Matrix - Gradient Boosting Tree

True Class/ Predicted Class	0	1	2
0	5	421	326
1	7	2383	277
2	1	697	883

Here, the green coloured cells represent the correctly predicted class data while the uncoloured ones are wrongly predicted values. It can be observed that out of a total of 5000 data, 3271 were predicted correctly.

$$\begin{aligned}\text{Hence accuracy} &= 3271/5000 * 100 \\ &= 65.43\%\end{aligned}$$

5.1.2.4 Confusion Matrix - XGBoost

True Class/ Predicted Class	0	1	2
0	45	347	245
1	2	2393	373
2	2	698	895

Here, the green coloured cells represent the correctly predicted class data while the uncoloured ones are wrongly predicted values. It can be observed that out of a total of 5000 data, 3333 were predicted correctly.

$$\begin{aligned}\text{Hence accuracy} &= 3333/5000 * 100 \\ &= 66.67\%\end{aligned}$$

5.2 Analysis of Deep Learning Model

5.2.1 Comparison with other models

Table 7 shows the accuracy of CNN model as compared with other models discussed in the previous section, namely, Logistic Regression, Random Forest, Gradient Boosting and XGBoost.

Model	Accuracy
Logistic Regression	56.82
Random Forest	79.41
Gradient Boosting Tree	65.43
XGBoost	66.67
CNN	88.98

As the table shows, CNN performs significantly better than the machine learning model with an accuracy of 88.98%. This is attributed to the role of convolution layers in CNN that act as feature extractor.

5.2.2 Accuracy with different number of epochs

An epoch is a single step in training a neural network; in other words when a neural network is trained on every training samples only in one pass we say that one epoch is finished. So training process may consist more than one epochs. With the increasing number of epochs, the accuracy of the model may increase, but when the epochs reach a certain number, accuracy becomes almost constant.

The below snippet shows the training set running on 500 epochs and giving an accuracy of 88.98%.

```
Epoch 487/500
30000/30000 [=====] - 8s 265us/step - loss: 0.3945 - acc: 0.8461
Epoch 488/500
30000/30000 [=====] - 8s 265us/step - loss: 0.4008 - acc: 0.8449
Epoch 489/500
30000/30000 [=====] - 8s 270us/step - loss: 0.3965 - acc: 0.8482
Epoch 490/500
30000/30000 [=====] - 8s 267us/step - loss: 0.3985 - acc: 0.8444
Epoch 491/500
30000/30000 [=====] - 8s 266us/step - loss: 0.4004 - acc: 0.8479
Epoch 492/500
30000/30000 [=====] - 8s 267us/step - loss: 0.3907 - acc: 0.8501
Epoch 493/500
30000/30000 [=====] - 8s 270us/step - loss: 0.3995 - acc: 0.8452
Epoch 494/500
30000/30000 [=====] - 8s 265us/step - loss: 0.4121 - acc: 0.8427
Epoch 495/500
30000/30000 [=====] - 8s 267us/step - loss: 0.4055 - acc: 0.8416
Epoch 496/500
30000/30000 [=====] - 8s 266us/step - loss: 0.3923 - acc: 0.8469
Epoch 497/500
30000/30000 [=====] - 8s 278us/step - loss: 0.4048 - acc: 0.8454
Epoch 498/500
30000/30000 [=====] - 9s 288us/step - loss: 0.3949 - acc: 0.8482
Epoch 499/500
30000/30000 [=====] - 8s 268us/step - loss: 0.3988 - acc: 0.8449
Epoch 500/500
30000/30000 [=====] - 8s 268us/step - loss: 0.3923 - acc: 0.8481

Accuracy: 88.977796
```

As we can observe, 500 epochs are run and an accuracy of approximately 88.98% is achieved.

5.2.3 Recall, Precision and F1-Score

Table 8 presents the results for the Machine Learning models used proposed methodology in terms of three evaluation metrics: Precision, Recall and F1 Score.

Class	Precision	Recall	F1-Score
0	0.95	0.56	0.71
1	0.9	0.95	0.92
2	0.86	0.92	0.89

Here the class 0 represents the data containing offensive speech, class 1 represents hateful speech and class 2 represents clear speech. It can be easily observed that CNN is giving better results for precision, recall and F1 value as compared to machine learning models discussed in the previous section. The value of precision and recall is reaching as high as 95% while that of F1-Score reaches 92%. These high values indicates the great ability of CNN model in predicting correct values in our problem domain. Different layers of CNN were added so as to get the best results.

5.2.4 Confusion Matrix

Table 9 represents the confusion matrix for CNN model used to analyse the data.

True Class/ Predicted Class	0	1	2
0	379	166	127
1	10	2529	119
2	10	119	1540

Here, the green coloured cells represent the correctly predicted class data while the uncoloured ones are wrongly predicted values. It can be observed that out of a total of 5000 data, 4448 were predicted correctly.

Hence accuracy = $4448/5000 * 100$

$$= 88.97\%$$

5.3 Evaluation of Tweets

Tweets on trending topics on twitter were extracted as described in the data collection section of the report. These tweets were extracted using a python script in which the hashtag to be extracted and the dates between which tweets are required was entered. Twitter API and beautiful soup library were not used because they were not able to extract tweets before a week's time.

Since the CNN model gave highest accuracy, it was used to evaluate the set of tweets extracted as hateful, offensive or neither of the two. The tweets extracted were categorised broadly into four major social issues:

5.3.1 The Naseeruddin Shah Case

5.3.2 Assembly Elections 2018

5.3.3 Brexit

5.3.4 Pulwana Attacks

The results of the evaluated tweets are described below:

5.3.1 The Naseeruddin Shah Case

Naseeruddin Shah is a popular Indian film and stage personality. In December 2018 there was a case of mob violence in the city of Bulandshahr, in Uttar Pradesh over the slaughtering of cows. IN the ensuing violence, a police inspector was killed. After this incident Naseeruddin made a public statement expressing his anger over the issue. He further described such events as a threat to minority communities in the country and questioned the inaction of the government on this issue. His comments were not taken well by certain right-wing organisations whose members then took to Social Media to publicly heckle the actor. Twitterattis flooded twitter with tweets containing hate speech and offensive language. The most popular hashtag used was #NaseeruddinGussaHai, which is used to extract the tweets.

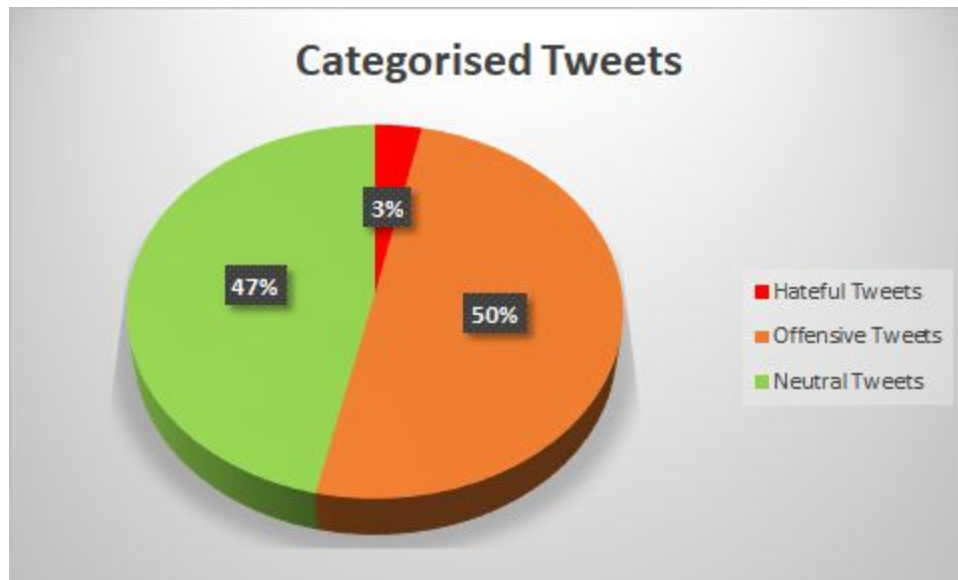
The evaluation results are as below:

Hateful Tweets	3.33%
Offensive Tweets	50.17%
Neutral Tweets	46.50%

As can be observed from the table, the percentage of offensive tweets is quite high owing to the abusive language used by Indians against Naseeruddin Shah and his family. The percentage of hateful tweets is 3.33% which can be considered as high since they target a particular community, Muslims in this case (since Naseeruddin Shah is muslim by religion). This analysis is summarised in a pie chart as below:

5.3.2 Assembly Elections 2018

IN December 2018, elections were held to elect the Legislative Assembly (known as the Vidhan Sabha) in the states of Madhya Pradesh, Chhattisgarh, Mizoram, Rajasthan and Telangana. The election was seen as a precursor to the 219 Lok Sabha elections and thus



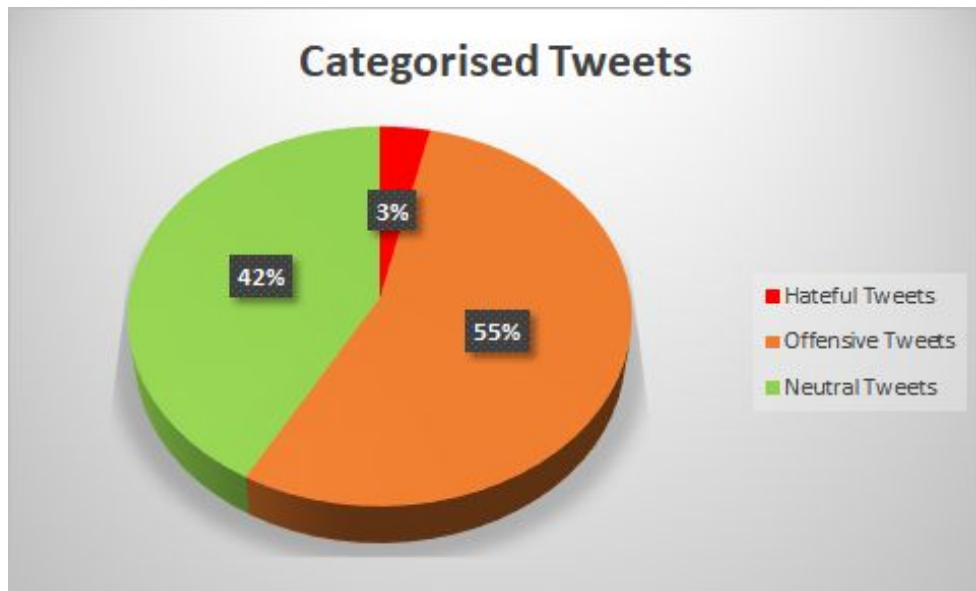
the candidates and the parties campaigned very aggressively. The campaign often turned divisive and personal with leaders and general supporters often using offensive language both in public meetings as well as online to target their opponents. The use of hateful and divisive language was rampant which is reflected clearly in the extracted twitter data.

The evaluation results are as below:

Hateful Tweets	3.53%
Offensive Tweets	54.61%
Neutral Tweets	41.86%

The results show a high content of offensive tweets as the BJP got defeated in all the constituencies, which was a major surprise for the party supporters countrywide. BJP is a

pro-Hindu party, hence breaking religious outrage in the country and on social media, which can be seen in the 3.53% of hateful tweets. This analysis is summarised in the pie chart as below:



5.3.3 Brexit

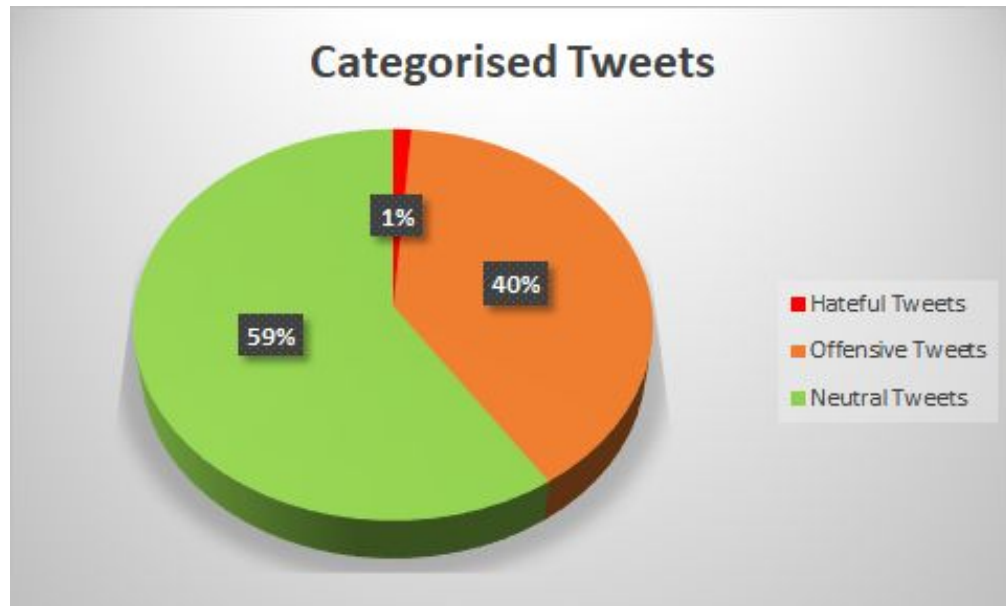
On 23rd June 2016 the United Kingdom held a referendum to determine whether it should remain a part of the European Union. The referendum resulted in a decision to leave the European Union; however the results were highly polarised. While England and Wales voted to leave the Union, Scotland and Northern Ireland voted to stay. As a result, many heated debates took place over the issue, particularly online. Some of these exchanges tended to get offensive with people often using strong language to disparage those with differing views, thereby making the topic a good sample for our analysis.

The evaluation results are as below:

Hateful Tweets	1.23%
Offensive Tweets	39.43%
Neutral Tweets	59.34%

The table shows a fairly less amount of offensive tweets and a highly less amount of hateful tweets, reason being this being a widespread issue rarely grew personal and heated arguments that often lead to use of offensive language. Also, this issue didn't

target a particular group or religion hence the percentage of hateful tweets is only 1.23%.



This analysis is summarised in the pie chart as below:

5.3.4 Pulwana Attacks

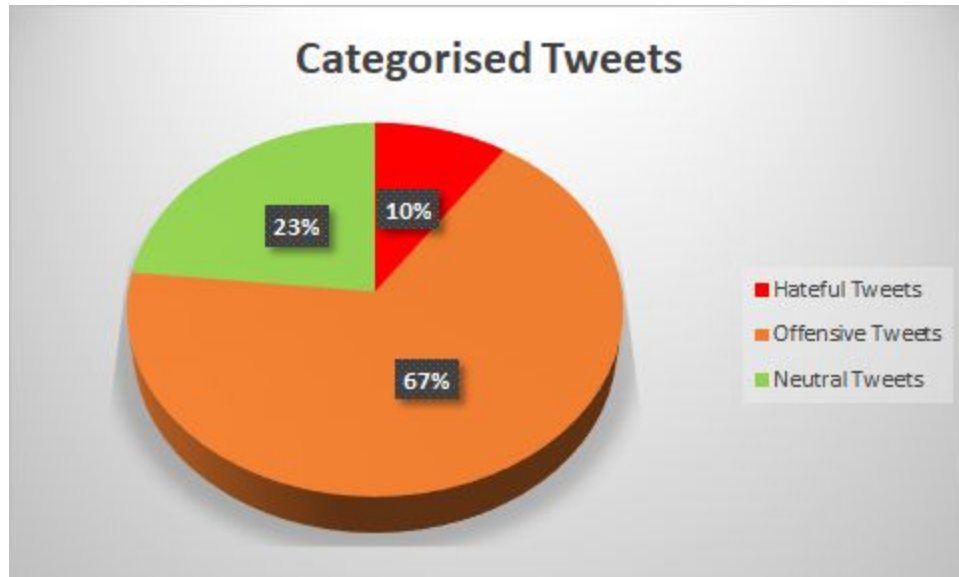
On the 14th of February 2019, a convoy of vehicles carrying Central Reserve Police Force (CRPF) personnel on the Jammu Srinagar National Highway was attacked by a suicide bomber in the Pulwama district of Jammu and Kashmir. A Pakistan-based militant group by the name of Jaish-e-Mohammed claimed responsibility for these attacks which resulted in the deaths of 40 Central Reserve Police Force (CRPF) personnel. As a result, large sections of the Indian population took to social media to voice their anger against the continuous acts of state sponsored terrorism by Pakistan and its unwillingness to prosecute internationally designated terror organisations operating from its soil.

The evaluation results are as below:

Hateful Tweets	9.56%
Offensive Tweets	67.11%
Neutral Tweets	23.33%

The results of this analysis proved to be completely different from the previous ones. There is a whopping 9.56% of offensive tweets, showing the outrage of Indians against the Pulwana attack, and a similar comeback expected from Pakistan. Similarly 67.11% of the tweets are offensive owing to the social media fights and comments of Indians and

Pakistanis. The number of neutral tweets is very less which is an alarming situation as this spreads a negative energy over social media and should be controlled by taking appropriate measures. This analysis is summarised as below:



7. Conclusion and Future Work

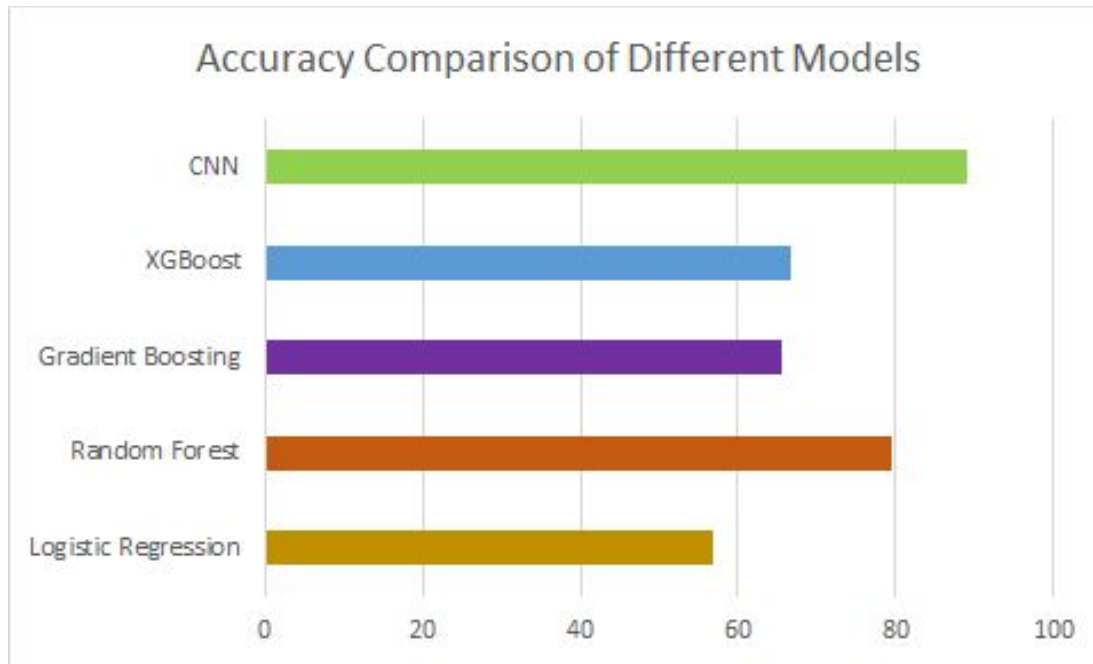


Figure 29 Accuracy Comparison of Different Models

In our work, for the purpose of identifying hate speech and offensive language on social networks, we built hateful and offensive domain lexicons and developed hybrid approaches combining both contextual and meta information involving hate speech. Our work proposes a model to analyze tweets, by developing a set of features using GloVe to be fed into classifiers for identification of hate speech using Machine Learning and Deep Learning Techniques. When annotated by humans, 12.51% of the total dataset of 57k tweets was found to be hateful and 54.76% was found to be offensive. The major contribution of this work is the improved performance of the Random forest classifier as compared to other machine learning classifiers as well as the baselines. This indicates the promise of the proposed set of features with a bagging based approach with minimal correlation show as compared to other classifiers.

Further, Deep Learning technique was deployed for the task of Hateful Speech Detection in tweets. Using different number of layers and an extensive training set, a high level of accuracy was achieved. Also, Adam optimiser and batch normalisation was employed to get better results.

A quantitative comparison between the various models revealed the effectiveness of a CNN based model in offensive speech detection in tweets. This was attributed to the ability of CNNs to spatially encode the tweets into a one dimensional structure to be fed into the model for training.

In addition, we analysed tweets that were trending on social media, namely twitter. Four social issues were picked, Pulwama attack, Naseeruddin Shah Case, Brexit case and Assembly Elections 2018. Results of the analysis are summarised below:

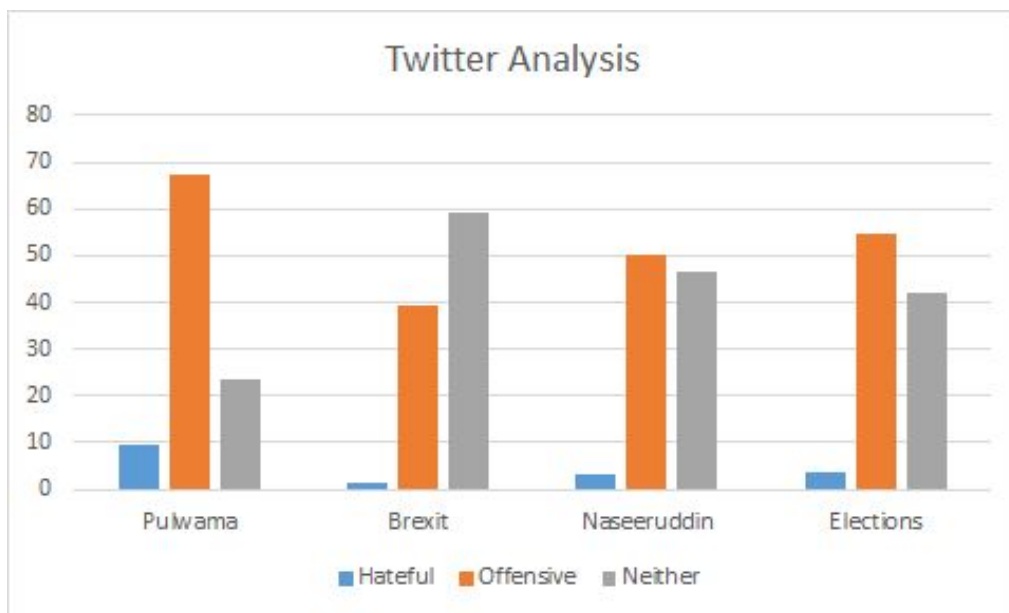


Figure 30 Analysis of Twitter Data

Using the results generated from our model on hateful speech detection, the work can be further extended in future on the following lines:

- The trained model can be used in creating a chrome app such that all hateful tweets could be hidden on the twitter webpage, and the offensive tweets can be displayed with a disclaimer.
- Furthermore, a twitteratti posting too much offensive content can be barred from using twitter. The ban can be a temporary ban or a permanent termination of account.
- In the trending hashtags section, a meter can be added that displays the percentage of offensive and hateful content with a disclaimer.
- Some tweets that are highly offensive can be hidden from a certain age or group of users, using the results generated by our model.

References:

- [1] L. Silva, M. Mondal, D. Correa, F. Benevenuto and I. Weber, “Analyzing the Targets of Hate in Online Social Media”, Tenth International AAAI Conference on Web and Social Media [2016]
- [2] F. Vigna, A. Cimino, F. Dell’Orletta, M. Petrocchi, and M. Tesconi, “Hate me, hate me not: Hate speech detection on Facebook”, First Italian Conference on Cybersecurity [2017]
- [3] B. Gambäck and U. Sikdar, ” Using Convolutional Neural Networks to Classify Hate-Speech”, First Workshop on Abusive Language Online, p. 85–90, [2017]
- [4] Z. Waseem and D. Hovy “Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter”, NAACL-HLT [2016]
- [5] A. Schmidt and M. Wiegand, “A Survey on Hate Speech Detection using Natural Language Processing ”, Fifth International Workshop on Natural Language Processing for Social Media [2017]
- [6] I. Kwok and Y. Wang, “Locate the Hate: Detecting Tweets against Blacks”, Twenty-Seventh AAAI Conference on Artificial Intelligence [2013]
- [7] T. Davidson, D. Warmusley, M. Macy and I. Weber “Automated Hate Speech Detection and the Problem of Offensive Language”, arXiv:151108630 [2015]
- [8] A. Bohra, D. Vijay, V. Singh, S. Akhtar and M. Shrivastava, “A Dataset of Hindi-English Code-Mixed Social Media Text for Hate Speech Detection”, 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics [2018]
- [9] S. Kamble and A. Joshi, “Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models”, 15th ICON [2018]
- [10] J. Bartlett, J. Reffin, N. Rumball and S. Williamson, “Antisocial media.” DEMOS [2014]
- [11] P. Burnap and M. L. Williams, “Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making.” Policy & Internet 7(2):223–242 [2015]
- [12] I. Chaudhry, “#hashtagging hate: Using twitter to track racism online.” First Monday 20(2) [2015]
- [13] D. Correa, L. Silva, M. Mondal, F. Benevenuto, and K. P. Gummadi, “The many shades of anonymity: Characterizing anonymous social media content.” In Proc. of ICWSM [2015]
- [14] N. Djuric, J. Zhou, R. Morris, M. V. Grbovic, Radosavljevic and N. Bhamidipati, “Hate speech detection with comment embeddings.” In WWW. [2015]
- [15] Y. Kim, “Convolutional neural networks for sentence classification.” arXiv:14085882 [2014]
- [16] C. Zhou, C. Sun, Z. Liu, F. Lau, “A C-LSTM neural network for text classification.” arXiv:151108630 [2015]

- [17] S. Li, B. Xu, T.L. Chung, “Definition extraction with LSTM recurrent neural networks. In: Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data.” Springer, p. 177–189 [2016]
- [18] J.W. Pennebaker, M.E. Francis, R.J. Booth, “Linguistic inquiry and word count” Liwc 2001. Mahway: Lawrence Erlbaum Associates [2001]
- [19] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard and D. McClosky, “The stanford corenlp natural language processing toolkit.” In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. p. 55–60 [2014]
- [20] A. Aizawa, “An information-theoretic perspective of tf–idf measures.” Information Processing & Management. p. 45–65 [2013].
- [21] D.M. Blei, A.Y. Ng and M.I. Jordan, “Latent dirichlet allocation.” Journal of machine Learning research. p. 993–1022 [2013].
- [22] A. Liaw, M. Wiener, et al. “Classification and regression by random forest.” R news p. 18–22. [2002]
- [23] J.H. Friedman, “Stochastic gradient boosting.” Computational Statistics & Data Analysis. p. 367–378. [2002]
- [24] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system.” In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM. p. 785–794 [2016]
- [25] P. Liu, X. Qiu and X. Huang, “Recurrent neural network for text classification with multi-task learning.” arXiv preprint arXiv:160505101 [2016]
- [26] A. Pak and P. Paroubek, “Twitter as a corpus for sentiment analysis and opinion mining.” In: LREc; vol. 10. [2010]
- [27] M. De Choudhury, M. Gamon, S. Counts and E. Horvitz, “Predicting depression via social media.” ICWSM [2013]

APPENDIX A:

A.1 Layers of CNN

```
model = Sequential()
e = Embedding(vocab_size, 100, weights=[embedding_matrix], input_length=4,
              trainable=False)

model.add(e)
model.add(Dropout(0.2))
model.add(Conv1D(256, #!!!!!!!!!!!!!!!!!!!!!!
                 3,
                 padding='valid',
                 activation='relu',
                 strides=1))
model.add(GlobalMaxPooling1D())
model.add(Dense(256))
model.add(Dropout(0.5))
model.add(Activation('relu'))

model.add(Dense(128))
model.add(Dropout(0.5))
model.add(Activation('relu'))

model.add(Dense(32))
model.add(Dropout(0.5))
model.add(Activation('relu'))

model.add(Dense(16))
model.add(Dropout(0.5))
model.add(Activation('relu'))

model.add(Dense(3))
model.add(Activation('softmax'))
model.summary()
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['acc', metrics.binary_accuracy])

# compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['acc'])
```

Figure.31 CNN code snippet showing different layers


A.2 Logistic Regression Model

LOGISTIC REGRESSION

```
[ ] from sklearn.linear_model import LogisticRegression
```

```
[ ] logisticRegr = LogisticRegression()
```

```
[ ] logisticRegr.fit(padded_docs_train, labels_np_train)
```

```
 /usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432:  
FutureWarning)  
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:469:  
"this warning.", FutureWarning)  
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,  
intercept_scaling=1, l1_ratio=None, max_iter=100,  
multi_class='warn', n_jobs=None, penalty='l2',  
random_state=None, solver='warn', tol=0.0001, verbose=0,  
warm_start=False)
```

```
[ ] score = logisticRegr.score(padded_docs_test, labels_np_test)  
print(score*100)  
y_pred = logisticRegr.predict(padded_docs_test)  
predictions = [round(value) for value in y_pred]
```


```
 55.49109821964393
```

Figure.32 Logistic Regression Code Snippet

A.3 XGBoost Model

XGB CLASSIFIER

```
[ ] !pip install xgboost
```

```
➤ Requirement already satisfied: xgboost in /usr/local/lib/python3.6/dist-packages  
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages  
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages
```

```
[ ] from xgboost import XGBClassifier  
    from sklearn.metrics import accuracy_score  
    from sklearn.metrics import precision_score  
    from sklearn.metrics import recall_score  
    from sklearn.metrics import f1_score
```

```
[ ] xgb = XGBClassifier()  
    xgb.fit(padded_docs_train, labels_np_train)  
    # make predictions for test data  
    y_pred = xgb.predict(padded_docs_test)  
    predictions = [round(value) for value in y_pred]  
    # evaluate predictions  
    accuracy = accuracy_score(labels_np_test, predictions)  
    print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
➤ Accuracy: 61.87%
```

Figure.33 XGBoost Classifier Code Snippet

A.4 Random Forest Model

RANDOM FOREST

```
[ ] from sklearn import tree, ensemble
    from imblearn.under_sampling import RandomUnderSampler
    from imblearn.pipeline import make_pipeline

    cart = tree.DecisionTreeClassifier(criterion='entropy', max_depth=8,
                                      min_samples_leaf=5)
    rus = make_pipeline(RandomUnderSampler(), tree.DecisionTreeClassifier(
        criterion='entropy', max_depth=8, min_samples_leaf=5))
    forest = ensemble.RandomForestClassifier(criterion='entropy',
                                           max_depth=15, min_samples_leaf=5)
    gboost = ensemble.GradientBoostingClassifier(max_depth=15, min_samples_leaf=5)

    cart.fit(padded_docs_train, labels_np_train)
    rus.fit(padded_docs_train, labels_np_train)
    forest.fit(padded_docs_train, labels_np_train)
    y_pred = forest.predict(padded_docs_test)
    predictions = [round(value) for value in y_pred]
    # evaluate predictions
    accuracy = calc(accuracy_score(labels_np_test, predictions))
    print("Accuracy: %.2f%%" % (accuracy * 100.0))
```


 /usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py:245: FutureWarning: "10 in version 0.20 to 100 in 0.22.", FutureWarning)
Accuracy: 79.83%

Figure.34 Random Forest Classifier Code Snippet