SARCASTIC AND NON-SARCASTIC TWEET CLASSIFICATION USING DEEP LEARNING

A PROJECT REPORT

Submitted by

BENNY RICHARDS. R	**211420205028**
ALLEN HARRIS. A	**211420205013**
MAGESH. S	**211420205084**

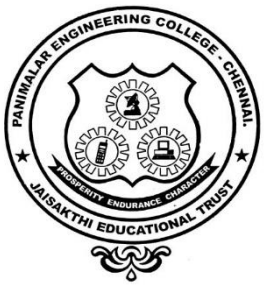
in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution,Affiliated to Anna University,Chennai)

MARCH 2024

BONAFIDE CERTIFICATE

Certified that this project report **"SARCASTIC AND NON-SARCASTIC TWEET CLASSIFICATION USING DEEP LEARNING"** is the bonafide work of **"BENNY RICHARDS. R (211420205028) , ALLEN HARRIS. A (211420205013), MAGESH. S (211420205084)"** who carried out the project under my supervision.

SIGNATURE

Dr. M. HELDA MERCY M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Department of Information Technology

Panimalar Engineering College

Chennai - 600 123

SIGNATURE

Mr. M. DILLI BABU, M.E., PhD.,

ASSOCIATE PROFESSOR

Department of Information Technology

Panimalar Engineering College

Chennai - 600 123

Submitted for the project and viva-voce examination held on _____

SIGNATURE

INTERNAL EXAMINER

SIGNATURE

EXTERNAL EXAMINER

DECLARATION

I hereby declare that the project report entitled "**SARCASTIC AND NON-SARCASTIC TWEET CLASSIFICATION USING DEEP LEARNING**" which is being submitted in partial fulfillment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology ' in **Panimalar Engineering College, Autonomous institution Affiliated to Anna University- Chennai** is the result of the project carried out by me under the guidance of **Mr. M. DILLI BABU, Associate Professor in the Department of Information Technology**. I further declare that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

Benny Richards. R Allen Harris. A Magesh. S

Date:

Place: Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: (Mr. M. DILLI BABU, M.E.,(PhD))

Place: Chennai (Associate Professor / IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Beloved Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E.,M.B.A.,Ph.D.,**and**Dr. SARANYA SREE SAKTHIKUMAR.,B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department,**Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU**, **M.E.,(Ph.D.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Mr. M. DILLI BABU, M.E.,(PhD).,** Associate Professor for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

Nowadays, social media has become a popular channel for people to exchange opinions through the user-generated text. Exploring the mechanisms about how customers' opinions towards products are influenced by friends, and further predicting their future opinions have attracted great attention from corporate administrators and researchers. Various influence models have already been proposed for the opinion prediction problem. However, they largely formulate opinions as derived sentiment categories or values but ignore the role of the content information. Besides, existing models only make use of the most recently received information without taking into consideration the long-term historical communication. To keep track of user opinion behaviors and infer user opinion influence from the historical exchanged textual information, we develop a content-based sequential opinion influence framework. Based on this framework, two opinion sentiment prediction models with alternative prediction strategies are proposed. In the experiments conducted on social media datasets, the proposed models outperform other popular influence models. An interesting finding based on a further analysis of user characteristic is that an individual's influence is correlated to her/his style of expressions. In our project we will be using Convolution Neural Network (CNN) + Long Short Term Memory (LSTM) as existing and Tree Convolution Neural Network (TCNN) as proposed system. From the result its proved that proposed Tree Convolution Neural Network (TCNN) works better than existing Convolution Neural Network (CNN) + Long Short Term Memory (LSTM) in terms of accuracy..

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
GRU	Gated Recurrent Unit
ML	Machine Learning
BILSTM	Bidirectional Long Short-Term Memory
LSA	Latent Semantic Analysis
POS	Parts Of Speech
FIS	Fuzzy Inference System
SIANN	Space Invariant Artificial Neural Networks
DFD	Data Flow Diagram
UML	Unified Modelling Language
JSON	Java Script Object Notation
DBMS	Database Management System
IDE	Integrated Development Environment
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

This research project delves into the utilization of deep learning methodologies for categorizing tweets into sarcastic and non-sarcastic classifications. Given the intricate nature of sarcasm detection within textual data, deep neural networks present an auspicious avenue for capturing nuanced linguistic subtleties. Through the employment of recurrent neural networks (RNNs), convolutional neural networks (CNNs), and attention mechanisms, our objective is to cultivate resilient models adept at discerning sarcasm within social media discourse. By conducting experiments involving diverse architectural configurations, training methodologies, and data augmentation techniques, our aim is to bolster the precision and generalizability of our models. Ultimately, this study contributes to the progression of natural language comprehension and sentiment analysis within the realm of social media platforms.

Within the landscape of social media, the ability to distinguish between sarcastic and non-sarcastic content holds significant importance in comprehending the underlying sentiments and contextual nuances of messages. Sarcasm, characterized by its ironic verbal expression, often involves conveying one message while intending another, thereby posing a considerable challenge for conventional machine learning algorithms in accurately identifying and deciphering it. However, recent advancements in deep learning methodologies have demonstrated promising capabilities in effectively categorizing tweets into sarcastic and non-sarcastic categories.

1.1 OVERVIEW

This paper explores the application of deep learning methodologies for the classification of tweets into sarcastic and non-sarcastic categories. We delve into the challenges posed by sarcasm detection, such as linguistic ambiguity, contextual nuances, and the absence of explicit markers. By leveraging the capabilities of deep neural networks, particularly recurrent neural networks (RNNs) and convolutional neural networks (CNNs), we aim to develop robust models capable of capturing the intricate patterns and cues indicative of sarcasm in text. Furthermore, we investigate the impact of different training strategies, such as data augmentation, transfer learning, and ensemble methods, to enhance the generalization capability and robustness of the

models. Additionally, we explore the incorporation of external knowledge sources, such as sentiment lexicons or contextual embeddings, to further improve sarcasm detection performance.

1.2 DOMAIN DESCRIPTION

Deep Learning:

In the statistical context, Deep Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Deep Learning involves concepts of automation, it requires human guidance. Deep Learning involves a high level of generalization in order to get a system that performs well on yet unseen data instances.

Deep learning is a relatively new discipline within Computer Science that provides a collection of data analysis techniques. Some of these techniques are based on well established statistical methods (e.g. logistic regression and principal component analysis) while many others are not.

Most statistical techniques follow the paradigm of determining a particular probabilistic model that best describes observed data among a class of related models. Similarly, most Deep learning techniques are designed to find models that best fit data (i.e. they solve certain optimization problems), except that these Deep learning models are no longer restricted to probabilistic ones.

Therefore, an advantage of Deep learning techniques over statistical ones is that the latter require underlying probabilistic models while the former do not. Even though some Deep learning techniques use probabilistic models, the classical statistical techniques are most often too stringent for the oncoming Big Data era, because data sources are increasingly complex and multi-faceted. Prescribing probabilistic models relating variables from disparate data sources that are plausible and amenable to statistical analysis might be extremely difficult if not impossible.

Deep learning might be able to provide a broader class of more flexible alternative analysis methods better suited to modern sources of data. It is imperative for statistical agencies to explore the possible use of Deep learning techniques to determine whether their future needs might be better met with such techniques than with traditional ones.

CLASSES OF DEEP LEARNING

There are two main classes of Deep learning techniques:

1. Supervised Deep learning and unsupervised Deep learning.

A. Examples of supervised learning Logistic regression (statistics) vs Support vector Deeps (Deep learning) Logistic regression, when used for prediction purposes, is an example of supervised Deep learning. In logistic regression, the values of a binary response variable (with values 0 or 1, say) as well as a number of predictor variables (covariates) are observed for a number of observation units. These are called training data in Deep learning terminology. The main hypotheses are that the response variable follows a Bernoulli distribution (a class of probabilistic models), and the link between the response and predictor variables is the relation that the logarithm of the posterior odds of the response is a linear function of the predictors. The response variables of the units are assumed to be independent of each other, and the method of maximum likelihood is applied to their joint probability distribution to find the optimal values for the coefficients (these parameterize the aforementioned joint distribution) in this linear function. The particular model with these optimal coefficient values is called the "fitted model," and can be used to "predict" the value of the response variable for a new unit (or, "classify" the new unit as 0 or 1) for which only the predictor values are known. Support Vector Deeps (SVM) are an example of a non-statistical supervised Deep learning technique; it has the same goal as the logistic regression classifier just described: Given training data, find the best-fitting SVM model, and then use the fitted SVM model to classify new units. The difference is that the underlying models for SVM are the collection of hyper planes in the space of the predictor variables. The optimization problem that needs to be solved is finding the hyper plane that best separates, in the predictor space, the units with response value 0 from those with response value 1. The logistic regression optimization problem comes from probability theory whereas that of SVM comes from geometry. Other supervised Deep learning techniques mentioned later in this briefing include decision trees, neural networks, and Bayesian networks.

B. Examples of unsupervised learning Principal component analysis (statistics) vs Cluster analysis (Deep learning). The main example of an unsupervised Deep learning technique that comes from classical statistics is principal component analysis, which seeks to "summarize" a set

of data points in high-dimensional space by finding orthogonal one-dimensional subspaces along which most of the variation in the data points is captured. The term “unsupervised” simply refers to the fact that there is no longer a response variable in the current setting. Cluster analysis and association analysis are examples of non-statistical unsupervised Deep learning techniques. The former seeks to determine inherent grouping structure in given data, whereas the latter seeks to determine co-occurrence patterns of items

1.3 PROJECT DESCRIPTION

In recent years, hate speech has been increasing in-person and online communication. The social media as well as other online platforms are playing an extensive role in the breeding and spread of hateful content – eventually which leads to hate crime. For example, according to recent surveys, the rise in online hate speech content has resulted in hate crimes including Trump's election in the US, the Manchester and London attacks in the UK, and terror attacks in New Zealand. To tackle these harmful consequences of hate speech, different steps including legislation have been taken by the European Union Commission. Recently, the European Union Commission also enforced social media networks to sign an EU hate speech code to remove hate speech content within 24 hours.

However, the manual process to identify and remove hate speech content is labor-intensive and time-consuming. Due to these concerns and widespread hate speech content on the internet, there is a strong motivation for automatic hate speech detection. The automatic detection of hate speech is a challenging task due to disagreements on different hate speech definitions. Therefore, some content might be hateful to some individuals and not to others, based on their concerned definitions. According hate speech is: “the content that promotes violence against individuals or groups based on race or ethnic origin, religion, disability, gender, age, veteran status, and sexual orientation/gender identity”. Despite these different definitions, some recent studies claimed favorable results to detect automatic hate speech in the text. The proposed solutions employed the different feature engineering techniques and ML algorithms to classify content as hate speech. Regardless of this extensive amount of work, it remains difficult to compare the performance of these approaches to classify hate speech content. To the best of our knowledge, the existing studies lack the comparative analysis of different feature engineering techniques and ML algorithms.

CHAPTER 2

LITERATURE SURVEY

2.1 Predicting the popularity of messages based on big data

Publisher: IEEE 2022

Jun Zhou; Guiping Wu; Manshu Tu; Bing Wang; Yan Zhang; Yonghong Yan

Social media has been an important way for people to get news. It is designed to make the sharing of messages very fast and easy. It also attracted the attention of a large number of researchers. There has been research concerning predicting what messages will be popular. But it lacks of in-depth study of what features play an important role in the prediction task. In the work, we systematically and comprehensively study three types of features: user features, text features and time features. Multiple comparison experiments are carried out on big data platform. Experimental results show that time features are the most valuable features, almost close to the effect of all the features, and the popularity of messages is predicted with a satisfactory accuracy.

2.2 Twitter's Hate Speech Multi-label Classification Using Bidirectional Long Short-term Memory (BiLSTM) Method

Publisher: IEEE 2022

Refa Annisatul Ilma; Setiawan Hadi; Afrida Helen

Since social media is one of the most likable products of techngetpeople get easier to express their opinions. Anyone be able to tell their opinion freely there. Unfortunately, its convenience has also become a boomerang for us, the easier every opinion conveyed the easier hate speech is expressed. This matter become the dark side of social media. Hate speech face us with a lot of dangers, such as violence, social conflict, even homicide. Therefore, preventing all of those dangers that might be occur because of hate speech is one of the prior things we need to do. This research was done as an attempt to take care of the dangers that could be done by hate speech. The attempt we tried to do is using multi-label text classification to predict hate speech with the Bidirectional Long Short-term Memory (BiLSTM) method. This multi-label text classification labelled every tweet in the dataset crawled from Twitter with 12 labels about hate speech. From

this experiment, we obtained the best hyperparameter value that could achieve great performance with 82.31% accuracy, 83.41% precision, 87.28% recall, and 85.30% F1-score.

2.3 Interest Analysis Using Semantic PageRank and Social Interaction Content

Publisher: IEEE 2022

Chung-Chi Huang; Lun-Wei Ku

Social media has long been a popular resource for sentiment analysis and data mining. In this paper, we learn to predict reader interest after article reading using social interaction content in social media. The abundant interaction content (e.g., reader feedback) aims to replace typically private reader profile and browse history. Our method involves estimating interest preferences with respect to article topics and identifying quality social content concerning informativity. During interest analysis, we combine and transform articles and their reader responses into PageRank word graph to balance author- and reader-end influence. Semantic features of words, such as their content sources (authors vs. readers), syntactic parts-of-speech, and degrees of references (i.e., significances) among authors and readers, are used to weight PageRank word graph. We present the prototype system, Interest Finder, that applies the method to reader interest prediction by calculating word interestingness scores. Two sets of evaluation show that traditional, local Page Rank can more accurately cover more span of reader interest with the help of topical interest preferences learned globally, word nodes' semantic information, and, most important of all, quality social interaction content such as reader feedback.

2.4 A Deep Learning Pipeline to Examine Political Bias with Congressional Speeches

Publisher: IEEE 2022

Prasad Hajare; Sadia Kamal; Siddharth Krishnan; Arunkumar Bagavathi

Computational methods to model political bias in social media involve several challenges due to heterogeneity, high-dimensionality, multiple modalities, and the scale of the data. Most of the current political bias detection methods rely heavily on the manually-labeled ground-truth data for the underlying political bias prediction tasks. Such methods are human-intensive labeling, labels related to only a specific problem, and the inability to determine the near future bias state of a social media conversation. In this work, we address such problems and give Deep learning

approaches to study political bias in two ideologically diverse social media forums: Gab and Twitter without the availability of human-annotated data. We propose a method to exploit the features of entities on transcripts collected from political speeches in US congress to label political bias of social media posts automatically without any human intervention. With existing Deep learning algorithms we achieve the highest accuracy of 70.5% and 65.1% to predict posts on Twitter and Gab data respectively. We also present a Deep learning approach that combines features from cascades and text to forecast cascade's political bias with an accuracy of about 85%.

2.5 Better Prevent than React: Deep Stratified Learning to Predict Hate Intensity of Twitter Reply Chains

Publisher: IEEE 2022

Dhruv Sahnan; Snehil Dahiya; Vasu Goel; Anil Bandhakavi; Tanmoy Chakraborty

Given a tweet, predicting the discussions that unfold around it is convoluted, to say the least. Most if not all of the discernibly benign tweets which seem innocuous may very well attract inflammatory posts (hate speech) from people who find them non-congenial. Therefore, building upon the aforementioned task and predicting if a tweet will incite hate speech is of critical importance. To stifle the dissemination of online hate speech is the need of the hour. Thus, there have been a handful of models for the detection of hate speech. Classical models work retrospectively by leveraging a reactive strategy – detection after the postage of hate speech, i.e., a backward trace after detection. Therefore, a benign post that may act as a surrogate to invoke toxicity in the near future, may not be flagged by the existing hate speech detection models. In this paper, we address this problem through a proactive strategy initiated to avert hate crime. We propose DRAGNET, a deep stratified learning framework which predicts the intensity of hatred that a root tweet can fetch through its subsequent replies. We extend the collection of social media discourse from our earlier work [1], comprising the entire reply chains up to ~5k root tweets catalogued into four controversial topics Similar to [1], we notice a handful of cases where despite the root tweets being non-hateful, the succeeding replies inject an enormous amount of toxicity into the discussions. DRAGNET turns out to be highly effective, significantly outperforming six state-of-the-art baselines. It beats the best baseline with an increase of 9.4% in the Pearson correlation coefficient and a decrease of 19% in Root Mean Square Error. Further,

DRAGNET'S deployment in Logically's advanced AI platform designed to monitor real-world problematic and hateful narratives has improved the aggregated insights extracted for understanding their spread, influence and thereby offering actionable intelligence to counter them

2.6 Cross-Lingual Few-Shot Hate Speech and Offensive Language Detection Using Meta Learning

Publisher: IEEE 2022

Marzieh Mozafari; Reza Farahbakhsh; Noel Crespi

Automatic detection of abusive online content such as hate speech, offensive language, threats, etc. has become prevalent in social media, with multiple efforts dedicated to detecting this phenomenon in English. However, detecting hatred and abuse in low-resource languages is a non-trivial challenge. The lack of sufficient labeled data in low-resource languages and inconsistent generalization ability of transformer-based multilingual pre-trained language models for typologically diverse languages make these models inefficient in some cases. We propose a meta learning-based approach to study the problem of few-shot hate speech and offensive language detection in low-resource languages that will allow hateful or offensive content to be predicted by only observing a few labeled data items in a specific target language. We investigate the feasibility of applying a meta learning approach in cross-lingual few-shot hate speech detection by leveraging two meta learning models based on optimization-based and metric-based (MAML and Proto-MAML) methods. To the best of our knowledge, this is the first effort of this kind. To evaluate the performance of our approach, we consider hate speech and offensive language detection as two separate tasks and make two diverse collections of different publicly available datasets comprising 15 datasets across 8 languages for hate speech and 6 datasets across 6 languages for offensive language. Our experiments show that meta learning-based models outperform transfer learning-based models in a majority of cases, and that Proto-MAML is the best performing model, as it can quickly generalize and adapt to new languages with only a few labeled data points (generally, 16 samples per class yields an effective performance) to identify hateful or offensive content.

2.7 Analyze Hate Contents on Sinhala Tweets using an Ensemble Method

Publisher: IEEE 2022

Madurangi Guruge; Supunmali Ahangama; Dinithi Amarasinghe

As social media grew in popularity among the general public, content and opinion sharing has become rapid and convenient. The majority of users rely on social media for news and trust the content shared by the network. Some individuals, both purposefully and unintentionally, disseminate hate content and instill hatred in their readers. Even in Sri Lanka, the spread of hate propaganda on social media has resulted in communal discord and a variety of concerns. Only a limited number of research studies have been conducted to analyze the hate content written in Sinhala. This work investigates a mechanism for detecting hate content typed in Sinhala language and posted on Twitter. The proposed supervised mechanism is an ensemble method that selects the most accurate result from different models. 63% of accuracy, 58% of F1 Score, 61% of Precision and 58% of Recall were achieved when predicting hate content.

2.8 Un-Compromised Credibility: Social Media Based Multi-Class Hate Speech Classification for Text

Publisher: IEEE 2022

Khubaib Ahmed Qureshi; Muhammad Sabih

There is an enormous growth of social media which fully promotes freedom of expression through its anonymity feature. Freedom of expression is a human right but hate speech towards a person or group based on race, caste, religion, ethnic or national origin, sex, disability, gender identity, etc. is an abuse of this sovereignty. It seriously promotes violence or hate crimes and creates an imbalance in society by damaging peace, credibility, and human rights, etc. Detecting hate speech in social media discourse is quite essential but a complex task. There are different challenges related to appropriate and social media-specific dataset availability and its high-performing supervised classifier for text-based hate speech detection. These issues are addressed in this study, which includes the availability of social media-specific broad and balanced dataset, with multi-class labels and its respective automatic classifier, a dataset with language subtleties, dataset labeled under a comprehensive definition and well-defined rules, dataset labeled with the strong agreement of annotators, etc. Addressing different categories of hate separately, this paper

aims to accurately predict their different forms, by exploring a group of text mining features. Two distinct groups of features are explored for problem suitability. These are baseline features and self-discovered/new features. Baseline features include the most commonly used effective features of related studies. Exploration found a few of them, like character and word n-grams, dependency tuples, sentiment scores, and count of 1st, 2nd person pronouns are more efficient than others. Due to the application of latent semantic analysis (LSA) for dimensionality reduction, this problem is benefited from the utilization of many complex and non-linear models and CAT Boost performed best. The proposed model is compared with related studies in addition to system baseline models. The results produced by the proposed model were much appreciating.

2.9 Using social networks to predict changes in health: Extended abstract
Publisher: IEEE 2022

Karen S. Jung; Ozan K. Tonguz

Social networking sites not only have billions of users but detailed content about each individual's daily life. This detailed information about a person's life could be exploited to allow individuals to learn more about themselves. In this paper, we introduce the concept of using social networks to foresee changes in an individual's health. We develop a new model that can predict if a person has recently undergone weight loss by analyzing the text from the person's tweets. Sentiment analysis, parts-of-speech (POS) tagging, and categorization are used in this model. The model is tested on Twitter users and a good statistical accuracy is observed. The success of this model suggests that this idea could be further explored to identify other patterns and create new models for a variety of health changes and health problems, particularly those that are of huge interest to individuals and businesses.

2.10 Early Prediction of Hate Speech Propagation
Publisher: IEEE 2022

Ken-Yu Lin; Roy Ka-Wei Lee; Wei Gao; Wen-Chih Peng

Online hate speech has disrupted the social connectedness in online communities and raises public safety concerns in our societies. Motivated by this rising issue, researchers have developed many Deep learning and deep learning methods to detect hate speech in social media

automatically. However, most of the existing automated solutions have focused on detecting hate speech in a single post, neglecting the network and information propagation effects of social media platforms. Ideally, the content moderators would want to identify the hateful posts and monitor posts and threads that are likely to induce hate. This paper aims to address this research gap by defining a new problem of early hate speech propagation prediction. We also propose HEAR, which is a deep learning model that utilizes a post's semantic, propagation structure, and temporal features to predict hateful propagation in social media. Through extensive experiments on two publicly available large Twitter datasets, we demonstrate HEAR's ability to outperform the state-of-the-art baselines in the early prediction of hateful propagation task. Specifically, with just 15 minutes of observation on a post's propagation, HEAR outperforms the best baselines by more than 10% (F1 score) in predicting the eventual amount of hateful posts it will induce.

2.11 Sensing, Understanding, and Shaping Social Behavior

Publisher: IEEE 2022

Erez Shmueli; Vivek K. Singh; Bruno Lepri; Alex Pentland

The ability to understand social systems through the aid of computational tools is central to the emerging field of computational social systems. Such understanding can answer epistemological questions on human behavior in a data-driven manner, and provide prescriptive guidelines for persuading humans to undertake certain actions in real-world social scenarios. The growing number of works in this subfield has the potential to impact multiple walks of human life including health, wellness, productivity, mobility, transportation, education, shopping, and sustenance. The contribution of this paper is twofold. First, we provide a functional survey of recent advances in sensing, understanding, and shaping human behavior, focusing on real-world behavior of users as measured using passive sensors. Second, we present a case study on how trust, which is an important building block of computational social systems, can be quantified, sensed, and applied to shape human behavior. Our findings suggest that:1) trust can be operationalized and predicted via computational methods (passive sensing and network analysis) and 2) trust has a significant impact on social persuasion; in fact, it was found to be significantly more effective than the closeness of ties in determining the amount of behavior change.

2.12 Opinion mining from social media using Fuzzy Inference System (FIS)

Publisher: IEEE 2022

K. Nivetha; G. Ragavi Ram; P. Ajitha

The Emotion recognition of the speaker can impact in the commercial sector to know the valuable feedback. Though many systems have appeared to perform similar task, they do not provide feasible solutions for various parameters, especially have lesser accuracy. Thus an integrated Fuzzy Inference System (FIS) with naïve Bayes classification that provided crisp outputs with greater accuracy. Fuzzy set theory is applied over the selected feature input and maps to the classified output. The rules are formulated to make suitable judgments on whether the uttered text is a negative or non-negative class of emotion. The defined set of database helps to correlate and identify similar texts. With this the opinions of the group of people can be interpreted. The bag of words are predicted with the help of wordlist database. Experimental analysis is made for comparing the existing hybrid Deep learning approach and the proposed algorithm. The results showed that the proposed system has greater accuracy and improved recognition ratio.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Convolution Neural Network (CNN):

In deep learning, a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as **shift invariant** or **space invariant artificial neural networks** (**SIANN**), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, Image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

Long Short Term Memory (LSTM):

Long short-term memory (**LSTM**) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

3.1.1 EXISTING SYSTEM DISADVANTAGES

- The current work only considers the influence from neighbors. So, to extend capture the influence of the external information sources on users opinion behaviors.
- Sentence extraction difficult to find the optimal opinions.
- Redundancy occur is more.
- Finding opinion target extraction and opinion summarization is very hard.

3.2 PROPOSED SYSTEM

Tree Convolution Neural Network (Tree CNN):

A Convolution Neural Network (CNN) is a class of artificial neural networks where connections between nodes form a graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feed forward neural networks, CNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks

such as unsegmented, connected handwriting recognition or speech recognition. The term "Enhanced neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feed forward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled.

3.2.1 ADVANTAGE OF PROPOSED SYSTEM

- Sentence extraction is not difficult to find the optimal opinions.
- Redundancy occur is less.
- Finding opinion target extraction and opinion summarization is very easy.

Working of Proposed System:

In propose a novel cross-domain sentiment classification algorithm and content based sentiment analysis algorithm based on term frequency, to analyze the sentiment polarity for short texts. It expands feature vectors based on unlabeled data from the target domain. In this way, some important sentiment indicators for the target domain are appended to feature vectors. At last, validation of algorithm on one target dataset. The project, mainly focus on positive and negative sentiment reviews. The first strategy is to identify the reviews as positive or negative by using the positive and negative words used in the review comments. Then expand features based on the co- occurrence frequency between a candidate of additional related feature and a domain-independent feature. Compared with point wise mutual information, sentiment related index considers the distributions of word occurrences instead of the co- occurrence frequency between different words, thus surmounting the challenge caused by infrequent features and words. Then calculate weightage and ranking in each opinions using content analysis algorithm.

CHAPTER 4

SYSTEM CONFIGURATION

4.1 HARDWARE SPECIFICATION:

Processor: I3 Processor.

Ram: 4 GB RAM

Hard Disk: 500 G.B Hard Disk

14 inch monitor

4.2 SOFTWARE REQUIREMENTS:

Operating System: Windows 7/8/10

Technology: Deep Learning

Languages used: Python

CHAPTER 5

SYSTEM DESIGN

5.1 DFD/ER DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

Data flow diagrams are also known as bubble charts.[5] DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data flow diagrams can be drawn up and compared with the new system's data flow diagrams to draw comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with a physical idea of

where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report. How any system is developed can be determined through a data flow diagram model.

In the course of developing a set of leveled data flow diagrams the analyst/designer is forced to address how the system may be decomposed into component sub-systems, and to identify the transaction data in the data model.

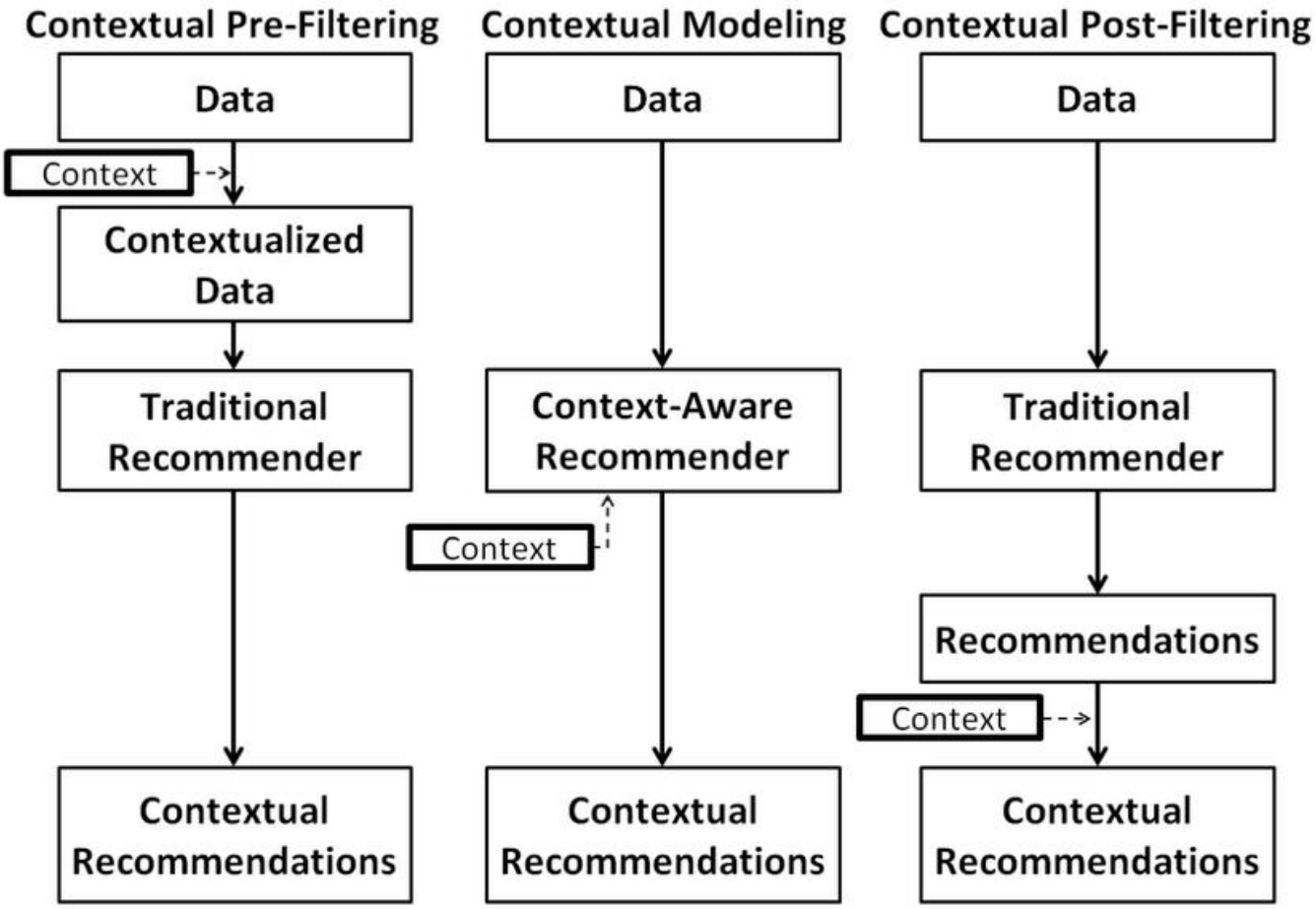


Fig. 5.1: Data Flow Diagram

5.2 UML DIAGRAMS

USECASE DIAGRAM

Use Case Diagrams describes the relationships and the dependencies between a group of Use Cases and the Actors participating in the process. Use Case Diagrams are meant to facilitate the communication with the future users of the system and with the customer and are especially helpful to determine the required features of the system is to have. Use Case Diagrams describes what the system should do but do not and cannot specify how this is to be achieved.

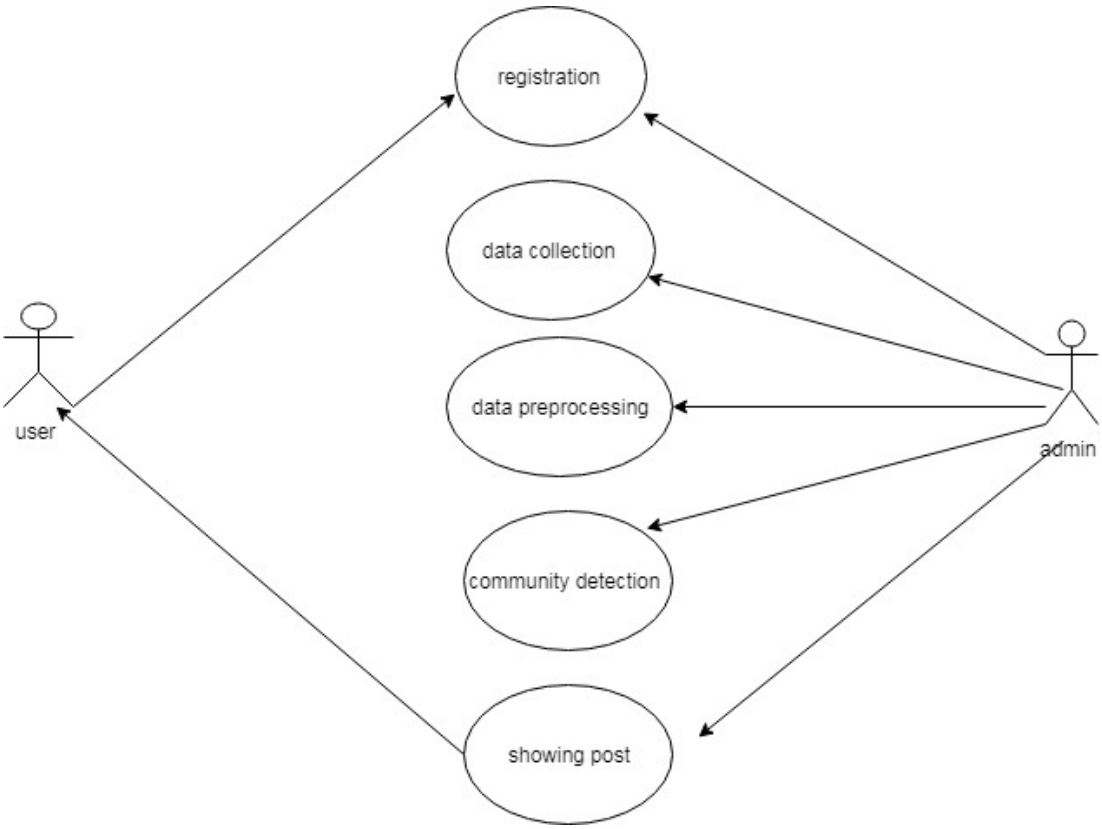


Fig. 5.2: Use Case Diagram

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 modeling language usable by both humans and Deeps.

Use cases are used in almost every project. They are helpful in exposing requirements and planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

5.3 ACTIVITY DIAGRAM

Activity Diagrams describes about the sequences of the activities in a system with the help of Activities. An Activity is a single step in a process. One Activity is one state in the system with internal activity and at least one outgoing transition. Activities can also have more than one outgoing transition if they have different conditions.

When to Use: Activity Diagrams

Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to model the workflow behind the system being designed. Activity Diagrams are also useful for: analyzing a use case by describing what actions need to take place and when they should occur; describing a complicated sequential algorithm; and modeling applications with parallel processes. 1 However, activity diagrams should not take the place of interaction diagrams and state diagrams. Activity diagrams do not give detail about how objects behave or how objects collaborate.

How to Draw: Activity Diagrams

Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities. A fork is used when multiple activities are occurring at the same time. The diagram below shows a fork after activity1. This indicates that both activity2 and activity3 are occurring at the same time. After activity2 there is a branch. The branch describes what activities will take place based on a set of conditions. All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch. After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.

Where to Use Activity Diagrams

The basic usage of activity diagram is similar to other four UML diagrams. The specific usage is to model the control flow from one activity to another. This control flow does not include messages.

Activity diagram is suitable for modeling the activity flow of the system. An application can have multiple systems. Activity diagram also captures these systems and describes the flow from one system to another. This specific usage is not available in other diagrams. These systems can be database, external queues, or any other system.

We will now look into the practical applications of the activity diagram. From the above discussion, it is clear that an activity diagram is drawn from a very high level. So it gives high level view of a system. This high level view is mainly for business users or any other person who is not a technical person.

This diagram is used to model the activities which are nothing but business requirements. The diagram has more impact on business understanding rather than on implementation details.

Activity diagram can be used for −

- Modeling work flow by using activities.
- Modeling business requirements.
- High level understanding of the system's functionalities.
- Investigating business requirements at a later stage.

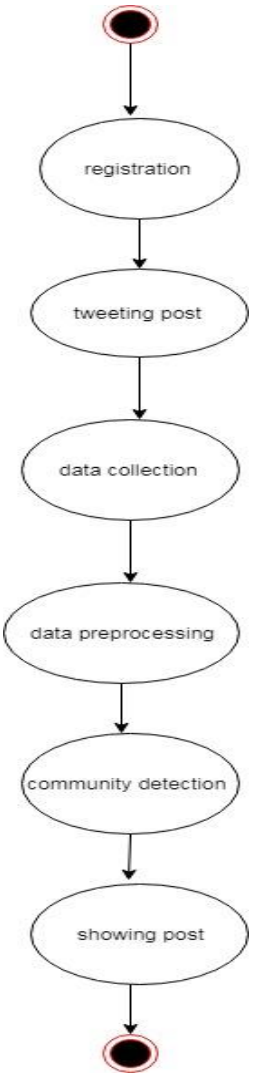


Fig. 5.3 Activity Diagram

5.4 DATABASE DESIGN

Database design is the process of producing a detailed data model of a database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object

classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

- Determine the data to be stored in the database.
- Determine the relationships between the different data elements.
- Superimpose a logical structure upon the data on the basis of these relationships.

Within the relational model the final step above can generally be broken down into two further steps, that of determining the grouping of information within the system, generally determining what are the basic objects about which information is being stored, and then determining the relationships between these groups of information, or objects. This step is not necessary with an Object database.

Determining data to be stored

In a majority of cases, a person who is doing the design of a database is a person with expertise in the area of database design, rather than expertise in the domain from which the data to be stored is drawn e.g. financial information, biological information etc. Therefore, the data to be stored in the database must be determined in cooperation with a person who does have expertise in that domain, and who is aware of what data must be stored within the system. This process is one which is generally considered part of requirements analysis, and requires skill on the part of the database designer to elicit the needed information from those with the domain knowledge. This is because those with the necessary domain knowledge frequently cannot express clearly what their system requirements for the database are as they are unaccustomed to thinking in terms of the discrete data elements which must be stored. Data to be stored can be determined by Requirement Specification.

Identifying Entities

The types of information that are saved in the database are called 'entities'. These entities exist in four kinds: people, things, events, and locations. Everything you could want to put in a database

fits into one of these categories. If the information you want to include doesn't fit into these categories, than it is probably not an entity but a property of an entity, an attribute.

To clarify the information given in this article we'll use an example. Imagine that you are creating a website for a shop, what kind of information do you have to deal with? In a shop you sell your products to customers. The "Shop" is a location; "Sale" is an event; "Products" are things; and "Customers" are people. These are all entities that need to be included in your database. But what other things are happening when selling a product? A customer comes into the shop, approaches the vendor, asks a question and gets an answer. "Vendors" also participate, and because vendors are people, we need a vendors entity.

Identifying Relationships

The next step is to determine the relationships between the entities and to determine the cardinality of each relationship. The relationship is the connection between the entities, just like in the real world: what does one entity do with the other, how do they relate to each other? For example, customers buy products, products are sold to customers, a sale comprises products, a sale happens in a shop.

5.5 DATA DICTIONARY

A data dictionary is an integral part of database. It holds the information about the database and the data that it stores , i.e. the metadata. Data dictionary contains the actual database descriptions used by the adbms. In most DBMSs ,the data dictionary is active and integrated. it means that the a DBMS checks the data dictionary every time the database is accessed. Since database is meant to be built and used by multiple users , making sure that everyone is aware of what types of data each field will accept becomes a challenge. so a data dictionary is effective add on to ensure data consistency. There exists no standard format for creating a data dictionary.meta data differs from table to table. The only prerequisite for a data dictionary is that it should be easily searchable.

CHAPTER 6

SYSTEM IMPLEMENTATION

Implementation is the process that actually yields the lowest-level system elements in the system hierarchy (system breakdown structure). The system elements are made, bought, or reused. Production involves the hardware fabrication processes of forming, removing, joining, and finishing; or the software realization processes of coding and testing; or the operational procedures development processes for operators' roles. If implementation involves a production process, a manufacturing system which uses the established technical and management processes may be required.

The purpose of the implementation process is to design and create (or fabricate) a system element conforming to that element’s design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices. This process bridges the system definition processes and the integration process.

System Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the user that it will work efficiently and effectively. The existing system was long time process.

The proposed system was developed using Python. The existing system caused long time transmission process but the system developed now has a very good user-friendly tool, which has a menu-based interface, graphical interface for the end user. After coding and testing, the project is to be installed on the necessary system. The executable file is to be created and loaded in the system. Again the code is tested in the installed system. Installing the developed code in system in the form of executable file is implementation.

6.1 MODULE DESCRIPTION

A. Twitter API

User needs to register first by giving his/her own information. While registering user should give their exact current location. If he/she is giving wrong location means, he is not supposed to register and login. And that user will be considered as a blocked user. So user needs to give only

the current location. After registration user will login with username and password. Then he/she can see their profile and can view all user tweets. Admin needs to login with username and password. If both match, he/she will be considered as a valid person. After login, admin can view all blocked user who gave wrong location while registration. Admin can able to see all users profile and tweets.

B. Post contents

In this module registered user can post tweets in instagram. If the user tries to post any content which contains bad words means, it will not get posted in the instagram account. So the algorithm will restrict the user not to post bad words. The general tweets can be posted in application and as well as in twitter.

C. Search Query

Here in this module, user can search for any query in the application. The query has been processed and extracted live tweets from the real time twitter. The Keywords related 100 tweets are extracted from the live twitter.

D. Preprocessing

In this step all the tweets are extracted from twitter are processed and the noise data are removed.

1) Stop words Removal: A dictionary based approach is been utilized to remove stop words from tweets. A generic stop word list containing 75 stop words created using hybrid approach is used. The algorithm is implemented as below given steps. The target text is tokenized and individual words are stored in array. A single stop word is read from stop word list. The stop word is compared to target text in form of array using sequential search technique. If it matches, the word in array is removed, and the comparison is continued till length of array. After removal of stop word completely, another stop word is read from stop word list and again algorithm runs continuously until all the stop words are compared. Resultant text devoid of stop words is displayed, also required statistics like stop word removed, no. of stop words removed from target text, total count of words in target text, count of words in resultant text, individual stop word count found in target text is displayed.

2) Stemming Technique: After removing the unwanted words from the tweet, stemming technique is processed. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.

E. Classification

After stemming process, all the tweet terms containing the keyword are classified into positive, negative and neutral tweets. Tree CNN is used for classification. Here we are having good words and bad words datasets. By comparing with this, we can classify the tweets into positive, negative and neutral tweets.

CHAPTER 7

SYSTEM STUDY

7.1 FEASIBILTY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The analyst conducting the study gathers information using a variety of methods, the most popular of which are:

- Developing and administering questionnaires to interested stakeholders, such as potential users of the information system.
- Observing or monitoring users of the current system to determine their needs as well as their satisfaction and dissatisfaction with the current system.
- Collecting, examining, and analyzing documents, reports, layouts, procedures, manuals, and any other documentation relating to the operations of the current system.
- Modeling, observing, and simulating the work activities of the current system.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components. These components are:

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY
- OPERATIONAL FEASIBILITY

7.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

7.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

7.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

7.5 OPERATIONAL FEASIBILITY

The ability, desire, and willingness of the stakeholders to use, support, and operate the proposed computer information system. The stakeholders include management, employees, customers, and suppliers. The stakeholders are interested in systems that are easy to operate, make few, if any, errors, produce the desired information, and fall within the objectives of the organization.

CHAPTER 8

SYSTEM TESTING

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.

TYPES OF TESTS

8.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

8.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

8.7 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8.8 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.9 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

8.10 Sample Test Case

Test case No.	Input	Expected Result	Obtained Result
01	Happy to see you	Not Sarcastic	Not Sarcastic
02	Good you broke my Window	Sarcasm	Sarcasm

Tab 8.10 Test Case Result

CHAPTER 9

SOFTWARE DESCRIPTION

A. Python

Python is a remarkably powerful dynamic, object-oriented programming language that is used in a wide variety of application domains. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries. To be precise, the following are some distinguishing features of Python:

- Very clear, readable syntax.
- Strong introspection capabilities.
- Full modularity.
- Exception-based error handling.
- High level dynamic data types.
- Supports object oriented, imperative and functional programming styles.
- Embeddable.
- Scalable
- Mature

With so much of freedom, Python helps the user to think problem centric rather than language centric as in other cases. These features makes Python a best option for scientific computing.

B. Open CV

Open CV is a library of programming functions for real time computer vision originally developed by Intel and now supported by Willogarage. It is free for use under the open source BSD license. The library has more than five hundred optimized algorithms. It is used around the world, with forty thousand people in the user group. Uses range from interactive art, to mine

inspection, and advanced robotics. The library is mainly written in C, which makes it portable to some specific platforms such as Digital Signal Processor. Wrappers for languages such as C, Python, Ruby and Java (using Java CV) have been developed to encourage adoption by a wider audience. The recent releases have interfaces for C++. It focuses mainly on real-time image processing. Open CV is a cross-platform library, which can run on Linux, Mac OS and Windows. To date, Open CV is the best open source computer vision library that developers and researchers can think of.

C. Tesseract

Tesseract is a free software OCR engine that was developed at HP between 1984 and 1994. HP released it to the community in 2005. Tesseract was introduced at the 1995 UNLV Annual Test OCR Accuracy [2] and is currently developed by Google released under the Apache License. It can now recognize 6 languages, and is fully UTF8 capable. Developers can train Tesseract with their own fonts and character mapping to obtain perfect efficiency.

9.1 ABOUT THE SOFTWARE "PYTHON"

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.

- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python - Environment Setup

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS

- Nokia mobile phones
- Windows CE
- Acorn/RISC OS
- BeOS
- Amiga
- VMS/OpenVMS
- QNX
- VxWorks
- Psion
- Python has also been ported to the Java and .NET virtual Deeps

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

You can download Python documentation https://www.python.org/doc/The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux Deep.

- Open a Web browser and go to https://www.python.org/downloads/.

- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- run ./configure script
- make
- make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

Windows Installation

Here are the steps to install Python on Windows Deep.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local Deep and then run it to find out if your Deep supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Macintosh Installation

Recent Macs come with Python installed, but it may be several years out of date.
See http://www.python.org/download/mac/ for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.

Jack Jansen maintains it and you can have full access to the entire documentation at his website − http://www.cwi.nl/~jack/macpython.html. You can find complete installation details for Mac OS installation.

Setting up PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix −

- **In the csh shell** − type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
- **In the bash shell (Linux)** − type export PATH="$PATH:/usr/local/bin/python" and press Enter.
- **In the sh or ksh shell** − type PATH="$PATH:/usr/local/bin/python" and press Enter.
- **Note** − /usr/local/bin/python is the path of the Python directory

Setting path at Windows

To add the Python directory to the path for a particular session in Windows −

At the command prompt − type path %path%;C:\Python and press Enter.

Note − C:\Python is the path of the Python directory

Python Environment Variables

Here are important environment variables, which can be recognized by Python −

Sr.No.	Variable & Description
1	**PYTHONPATH** It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by the Python installer.
2	**PYTHONSTARTUP** It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH.
3	**PYTHONCASEOK** It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.
4	**PYTHONHOME** It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.

Running Python

There are three different ways to start Python −

Interactive Interpreter

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter **python** the command line.

Start coding right away in the interactive interpreter.

```
$python # Unix/Linux
or
python% # Unix/Linux
or
C:> python # Windows/DOS
```

Here is the list of all the available command line options −

Sr.No.	Option & Description
1	**-d** It provides debug output.
2	**-O** It generates optimized bytecode (resulting in .pyo files).
3	**-S** Do not run import site to look for Python paths on startup.
4	**-v** verbose output (detailed trace on import statements).
5	**-X** disable class-based built-in exceptions (just use strings); obsolete starting with version 1.6.
6	**-c cmd**

	run Python script sent in as cmd string
7	**File** run Python script from given file

Script from the Command-line

A Python script can be executed at command line by invoking the interpreter on your application, as in the following −

```
$python script.py # Unix/Linux

or

python% script.py # Unix/Linux

or

C: >python script.py # Windows/DOS
```

Note − Be sure the file permission mode allows execution.

Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

- **Unix** − IDLE is the very first Unix IDE for Python.
- **Windows** − PythonWin is the first Windows interface for Python and is an IDE with a GUI.
- **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

Note − All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavor of Linux.

We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Audience

This tutorial has been prepared for those who seek to learn the basics and various functions of Pandas. It will be specifically useful for people working with data cleansing and analysis. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus. Pandas library uses most of the functionalities of NumPy. It is suggested that you go through our tutorial on NumPy before proceeding with this tutorial. You can access it from Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Audience

This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.umPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for

MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.tandard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, **pip**.

```
pip install numpy
```

The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python).

9.2 Sample Code

9.2.1 Model Code

```
import pandas as pd

import warnings

warnings.filterwarnings('ignore')

from lib.utils import *

train = pd.read_csv('train.csv')

print("Training Set:"% train.columns, train.shape)

print('Train_Set -----')

print(train.isnull().sum())

train.head()

import re

from sklearn.utils import resample
```

```
def  clean_text(df, text_field):
    df[text_field] = df[text_field].str.lower()
    df[text_field] = df[text_field].apply(lambda elem: re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^rt|http.+?", "", elem))
    return df
train_clean=clean_text(train,'tweet')
train_majority = train_clean[train_clean.label==0]
train_minority = train_clean[train_clean.label==1]
train_minority_upsampled = resample(train_minority,
                                    replace=True,
                                    n_samples=len(train_majority),
                                    random_state=123)
train_upsampled = pd.concat([train_minority_upsampled, train_majority])
train_upsampled['label'].value_counts()
from wordcloud import WordCloud
import matplotlib.pyplot as plt
fig, axs = plt.subplots(1,2 , figsize=(16,8))
text_pos = " ".join(train_clean['tweet'][train.label == 0])
text_neg = " ".join(train_clean['tweet'][train.label == 1])
train_cloud_pos = WordCloud(collocations = False, background_color = 'white').generate(text_pos)
```

```
train_cloud_neg = WordCloud(collocations = False, background_color =
'black').generate(text_neg)

axs[0].imshow(train_cloud_pos, interpolation='bilinear')

axs[0].axis('off')

axs[0].set_title('Non-Hate Comments')

axs[1].imshow(train_cloud_neg, interpolation='bilinear')

axs[1].axis('off')

axs[1].set_title('Hate Comments')

plt.show()

import xgboost as xgb

import numpy as np

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.pipeline import Pipeline

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.feature_extraction.text import TfidfTransformer

from sklearn.model_selection import train_test_split, RepeatedStratifiedKFold, cross_val_score

pipeline_xgb = Pipeline([

    ('vect', CountVectorizer()),

    ('tfidf',  TfidfTransformer()),

    ('nb', xgb.XGBClassifier(use_label =False)),])

x_train, x_test, y_train, y_test = train_test_split(train_upsampled['tweet'],
```

```
                                        train_upsampled['label'],random_state = 0)
X,y=scaler_transform(train_upsampled['tweet'])
from sklearn.model_selection import train_test_split
ac=[]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
x = layers.Input(shape=(12, 7))
model=tf.keras.layers.LSTM(
    units=8,
    activation="tanh",
    recurrent_activation="sigmoid",
    use_bias=True,
    kernel_initializer="glorot_uniform",
    recurrent_initializer="orthogonal",
    bias_initializer="zeros",
    unit_forget_bias=True,
    kernel_regularizer=None,
    recurrent_regularizer=None,
    bias_regularizer=None,
```

```
    activity_regularizer=None,
    kernel_constraint=None,
    recurrent_constraint=None,
    bias_constraint=None,
    dropout=0.0,
    recurrent_dropout=0.0,
    return_sequences=False,
    return_state=False,
    go_backwards=False,
    stateful=False,
    time_major=False,
    unroll=False,
)
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(8,activation='relu',input_dim=4))
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer='adam')
model.fit(X_train, y_train,epochs=5)
```

```
ac.append(accuracy_score(model,y_test,sample_weight=1)*100)

from keras.models import Sequential

from keras.layers import Dense

class NodeList (object):

    def __init__(self, label, values, nodes, class_position):

        self.label = label

        self.values = values

        self.nodes = nodes

        self.class_position = class_position

    def __str__(self):

        return {'label': self.label, 'values': self.values, 'nodes':self.nodes, 'class_position':
self.class_position}

    def __unicode__(self):

        return {'label': self.label, 'values': self.values, 'nodes':self.nodes, 'class_position':
self.class_position}

    def __repr__(self):

        return 'label: ' + str(self.label) + ' values: ' + str(self.values) + ' nodes: ' + str(self.nodes)+'
class_pos: '+ str(self.class_position)

class CnnNode (object):

    def __init__(self, num_classes, labels = [], input_shape=(28,28,1), max_leafes=10):

        self.net = CNN(num_classes, input_shape)

        self.num_classes = num_classes
```

```
        self.childrens = [label for label in labels]
        self.childrens_leaf = [True for _ in range(num_classes)]
        self.labels = labels
        self.max_leafes = max_leafes
        self.labels_transform = {}
        for nc in range(num_classes):
            self.labels_transform[labels[nc]] = []
            self.labels_transform[labels[nc]].append(labels[nc])
    def get_num_leafnodes(self):
        count = 0
        for is_leaf in self.childrens_leaf:
            if is_leaf:
                count = count + 1
        return count
model = Sequential()
model.add(Dense(11,activation='relu',input_dim=4))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam')
model.fit(X_train,y_train,epochs=3)
ac.append(accuracy_score(model,y_test,sample_weight=0.2)*100)
import numpy as np
```

```
import seaborn as sns

import matplotlib as plt

plt.style.use('ggplot')

x=['CNN+LSTM',' Tree CNN']

ac=ac

ax=sns.barplot(x=x,y=ac)

ax.set_title('Accuracy comparison')

ax.set_ylabel('Accuracy')

#ax.yaxis.set_major_locator(ticker.LinearLocator())

print("the accuracy of {} is {} and {} is {}".format(x[0],ac[0],x[1],ac[1]))

ax.set_ylim(50,100)

import pandas as pd

data={'Agorithms':x,

    "accuracy":ac}

df=pd.DataFrame(data)

df.head()

print(ac)
```

9.2.2 app.py

```
import os

from flask import Flask
```

```
from flask import flash, jsonify, make_response, render_template, request, redirect, session, url_for
import detection
import pandas as pd
from gensim.models.fasttext import FastText
from werkzeug.utils import secure_filename
app = Flask(__name__, template_folder='views')
app.config["DEBUG"] = True
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')
@app.route('/detect', methods=['GET', 'POST'])
def detect_text():
    if request.method == "POST":
        if 'text' in request.form and request.form['text'] != "":
            text = []
            text.insert(0, request.form['text'])
            predict_results = detection.sarcasm_detection(text)
            result = dict()
            result['text'] = text[0]
            result['predict_result'] = predict_results[0]
```

```
            return render_template('index.html', result=result, category="text")
        else:
            flash('Input the text first to do cyberbullying detection', 'error_text')
            return redirect(url_for('index'))
    else:
        flash('Only POST method allowed', 'error_text')
        return redirect(url_for('index'))
@app.route('/detect_file', methods=['GET', 'POST'])
def detect_file():
    if request.method == "POST":
        file = request.files['table']
        if 'table' in request.files and file and file.filename != "":
            table = pd.read_csv(file, header=None)
            predict_results = detection.sarcasm_detection(table[0])
            results = dict()
            ctr_result = 0
            for predict_result in predict_results:
                if len(predict_results[predict_result]) != 0:
                    results[ctr_result] = dict()
                    results[ctr_result]['text'] = table[0][ctr_result]
                    results[ctr_result]['predict_result'] = predict_results[predict_result]
```

```
                ctr_result += 1
            return render_template('index.html', results=results, category="table")
        else:
            flash('Upload the file first to sarcasm detection', 'error_file')
            return redirect(url_for('index'))
    else:
        flash('Only POST method allowed', 'error_file')
        return redirect(url_for('index'))
app.run()
```

9.2.3 Detection.py

```
import pandas as pd
import re
import keras
import numpy as np
import time
import nltk
import tensorflow as tf
from keras.wrappers.scikit_learn import KerasClassifier
from keras_preprocessing import text
from keras.preprocessing.text import text_to_word_sequence
```

```
from keras.models import Model, Sequential, load_model
from keras.layers import LSTM
from keras.layers import Embedding
from keras.layers import Dense
from gensim.models.fasttext import FastText
def tokenization(dataTeks):
    tokenizer = keras.preprocessing.text.Tokenizer(num_words=1439, split=" ")
    tokenizer.fit_on_texts(dataTeks.values)
    word_index = tokenizer.word_index
    X = tokenizer.texts_to_sequences(dataTeks.values)
    X = tf.keras.preprocessing.sequence.pad_sequences(X, maxlen=25)
    return X, word_index
def textPreprocessing(dataTeks):
    dataTeks = dataTeks.apply(lambda x: x.lower())
    dataTeks = dataTeks.apply(lambda x: re.sub('\\\\"', ' ', x))
    dataTeks = dataTeks.apply(lambda x: re.sub('[^a-zA-Z0-9\s]', ' ', x))
    dataTeks = dataTeks.apply(lambda x: re.sub('\s+', ' ', x))
    dataTeks = dataTeks.apply(lambda x: x.strip())

    data, word_index = tokenization(dataTeks)
    return data, word_index
```

```
def prediction(pred):
    threshold = 0.5
    hasil_predict = [""] * (len(pred) + 1)
    for i in range(0, len(pred)):
        if pred[i] >= threshold:
            hasil_predict[i] = "Sarcasm"
        else:
            hasil_predict[i] = "Not Sarrcastic"
    return hasil_predict
def fasttext(word_index):
    # get the vectors
    loaded_ft = FastText.load("ft_model_100_andriansyah_defaultconfig.bin")
    embedding_matrix = np.zeros((len(word_index)+1, 100))
    word_not_found = []
    for word, i in word_index.items():
        if word in loaded_ft.wv:
            embedding_vector = loaded_ft[word]
            # words that cannot be found will be set to 0
            if embedding_vector is not None:
                embedding_matrix[i] = embedding_vector
            else:
```

```
                word_not_found.append(word)
        else:
            word_not_found.append(word)
    return embedding_matrix;
#global graph
#graph = tf.get_default_graph()
epoch = 10
batch_size = 8
unit = 25
dropout = 0.05
regularization = 0.001
activation = 'sigmoid'
optimizer = 'Adadelta'
def sarcasm_detection(dataTeks):
    dataTeks = pd.DataFrame(dataTeks)
    data, word_index = textPreprocessing(dataTeks[0])
    embedding_matrix = fasttext(word_index)
    if epoch==10:
        model = Sequential()
        model.add(Embedding(len(word_index)+1, 100, input_length=data.shape[1],
                            weights=[embedding_matrix], trainable=False))
```

```
        model.add(LSTM(unit, return_sequences=True, dropout=dropout,
recurrent_dropout=dropout,
                       kernel_regularizer=keras.regularizers.l2(regularization), name='lstm_3'))
        model.add(LSTM(unit, dropout=dropout, recurrent_dropout=dropout,
                       kernel_regularizer=keras.regularizers.l2(regularization), name='lstm_4'))
        model.add(Dense(1, activation=activation, name='dense_2'))
        model.load_weights('model1.h5', by_name='lstm_3')
        model.load_weights('model1.h5', by_name='lstm_4')
        model.load_weights('model1.h5', by_name='dense_2')
        model.compile(loss='binary_crossentropy', optimizer=optimizer)
        pred = model.predict(data)
    hasil_predict = prediction(pred)
    result = dict()
    ctr = 0
    for hasil in hasil_predict:
        result[ctr] = hasil
        ctr += 1
    return result
```

9.3 Sample Output

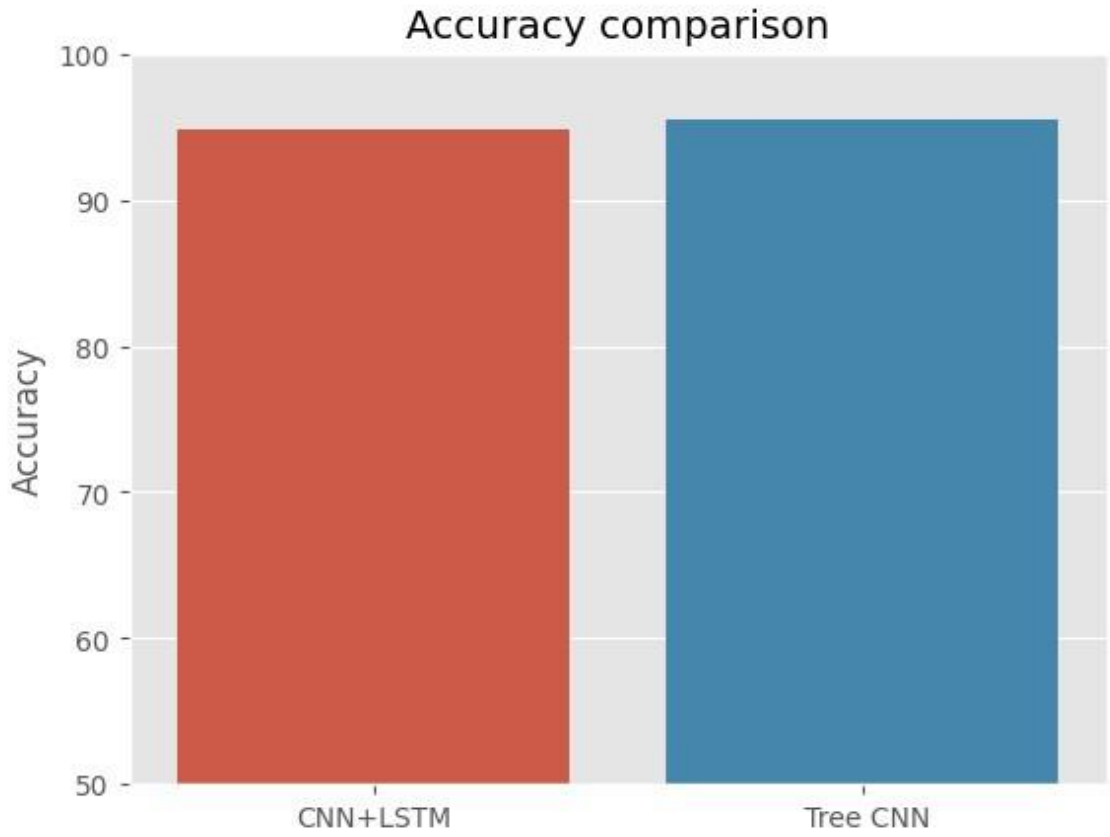


Fig 9.3.1 Accuracy Analysis

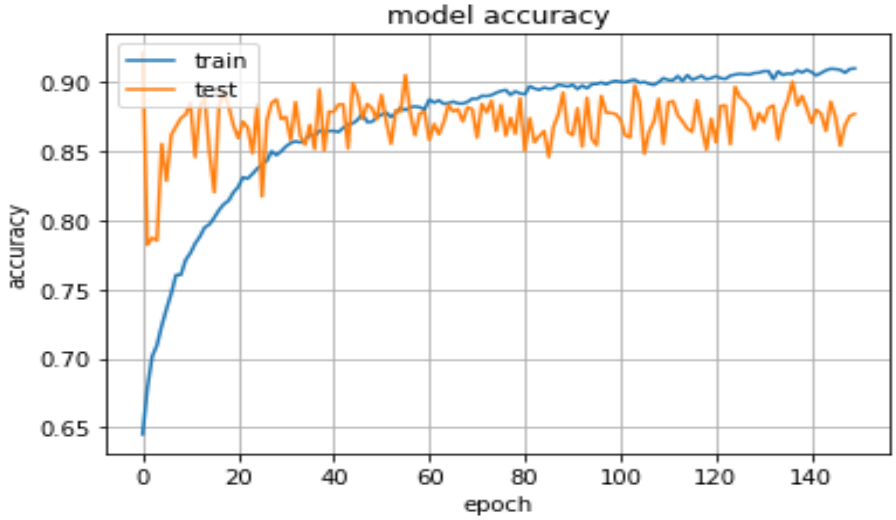


Fig 9.3.2 Model Accuracy on Test and Train Data

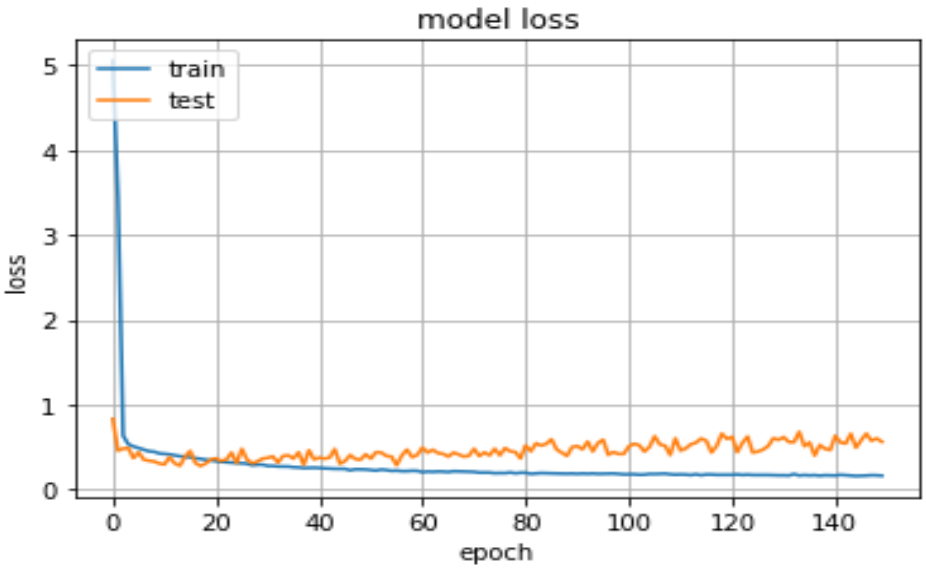


Fig 9.3.3 Model Loss on Test and Train Data

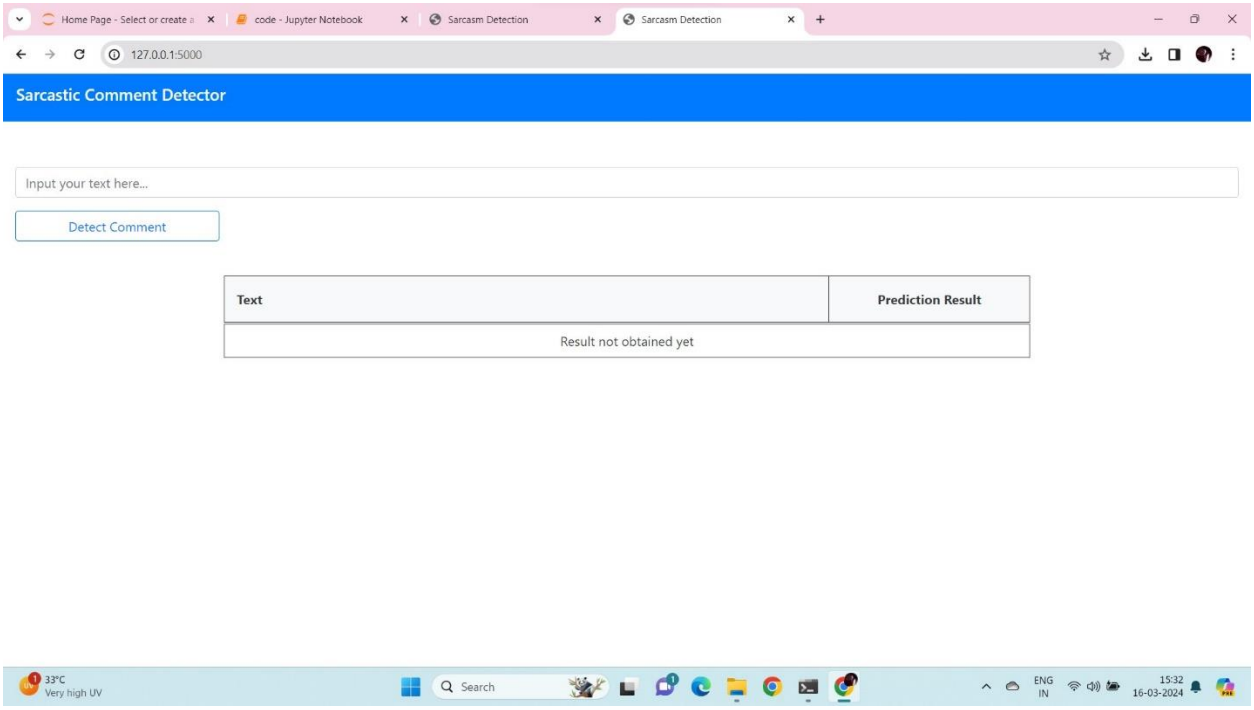


Fig 9.3.4 Home Page

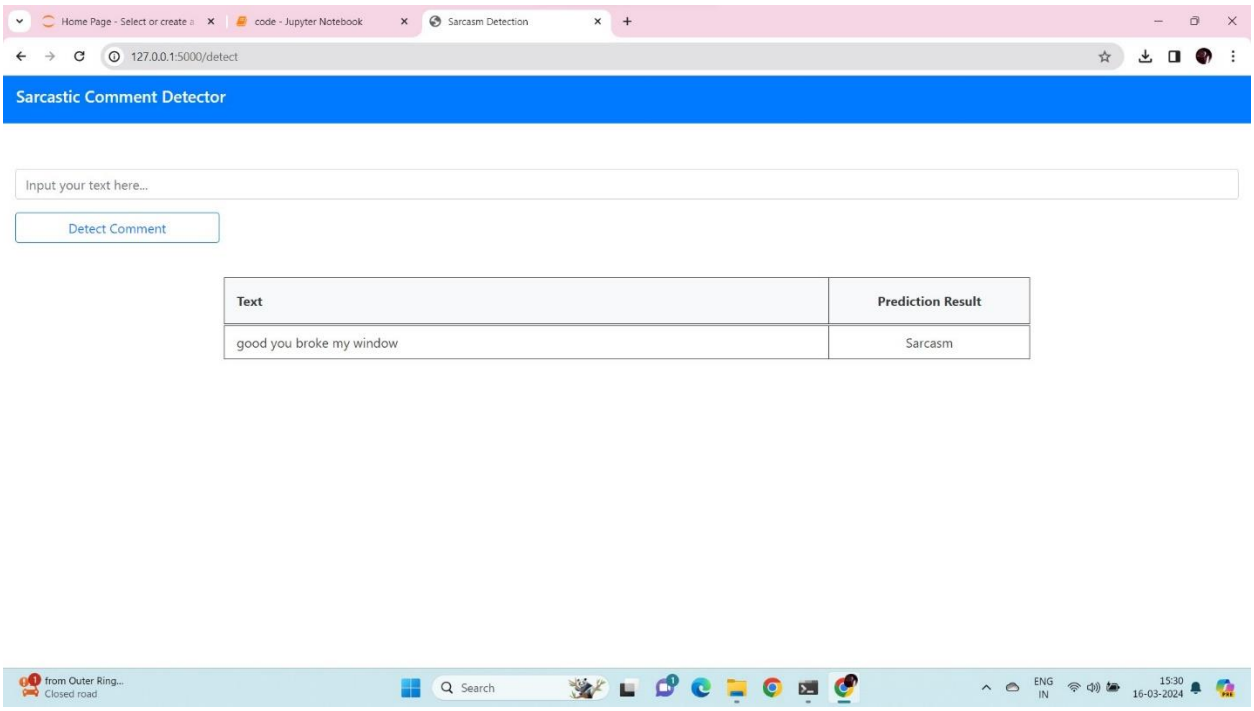


Fig 9.3.5 Sample Output 1

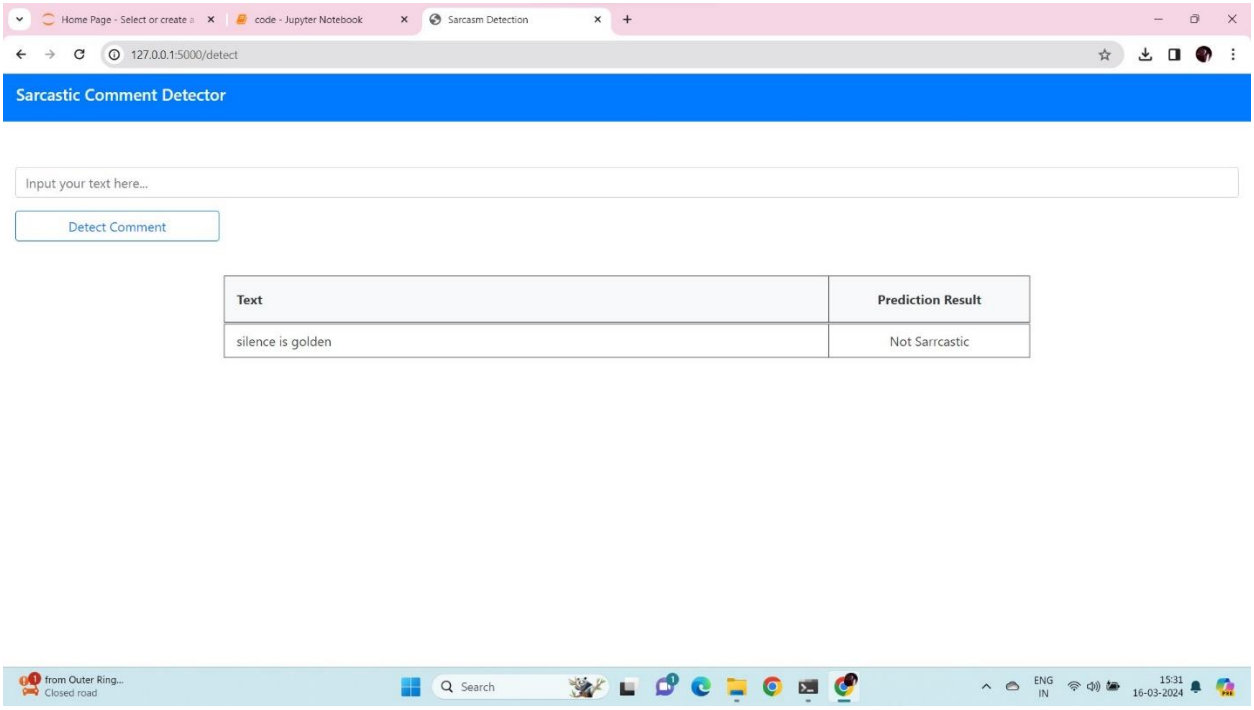


Fig 9.3.6 Sample Output 2

CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 Conclusion

The experiments conducted on the Instagram and Twitter dataset demonstrate the effectiveness of the two proposed models. The prediction ability of the proposed model is further verified on the opinion word prediction task. Based on the learned influence, we explore the expression styles of users with different influence powers, which provide the valuable information for companies to manage their accounts and design marketing plans.

10.2 Future Scope

Proposed approach can be used to extract both facts and opinions from social media content. It uses two different cross domain datasets to analyze the sentiment of another domain. Most of the time the rating of specific reviews plays an important role for sentiment analysis. Because in tweets there is a chance to include the duplicate review. Because of that duplicate review there is a chance to incorrect analysis for a specific product. To avoid this along with the content of a specific review the additional attributes like Rating and helpful to be used for finding the term frequency. From that the duplicate reviews can be easily filtered out from the analysis. Hence the accuracy of the sentiment polarity will be very high.

The proposed approach offers a versatile solution for extracting both factual information and subjective opinions from the vast landscape of social media content. By harnessing the power of two distinct cross-domain datasets, this methodology enables a comprehensive analysis of sentiment across different domains. One of the key challenges in sentiment analysis, particularly when dealing with platforms like Twitter, lies in the potential presence of duplicate reviews. These duplicates can significantly skew the analysis and lead to inaccurate conclusions regarding a particular product or topic.

In mitigating this challenge, the approach takes into account not only the textual content of reviews but also additional attributes such as rating and helpfulness. These supplementary attributes provide valuable context and can help discern genuine opinions from duplicate or spam-like content. By incorporating these factors into the analysis, the approach enhances its ability to accurately identify and filter out duplicate reviews, thereby improving the overall precision of sentiment polarity determination.

The inclusion of rating and helpfulness metrics serves as a crucial refinement step in the sentiment analysis process. Ratings offer a quantifiable measure of user satisfaction or dissatisfaction with a product or service, providing valuable insights into the sentiment expressed within a review. Meanwhile, the helpfulness metric provides an indication of the perceived value of a review within the community, further aiding in the identification of relevant and meaningful content.

By leveraging these additional attributes alongside the textual content of reviews, the approach adopts a more holistic approach to sentiment analysis. Rather than relying solely on the text itself, which may be prone to duplication or manipulation, it incorporates multiple dimensions of information to more accurately gauge sentiment.

Furthermore, the integration of term frequency analysis based on these attributes facilitates the identification and exclusion of duplicate reviews. By examining the frequency of terms within reviews while considering associated ratings and helpfulness scores, the approach can effectively distinguish between genuine reviews and duplicates. This nuanced understanding of review content allows for a more precise assessment of sentiment polarity, thereby enhancing the overall accuracy of the analysis.

As a result of these enhancements, the proposed approach demonstrates a notable improvement in sentiment analysis accuracy. By systematically filtering out duplicate reviews and leveraging additional attributes to refine the analysis, it minimizes the risk of erroneous conclusions and ensures that sentiment polarity assessments are based on reliable and representative data. This heightened accuracy not only benefits businesses and organizations seeking to understand consumer sentiment but also contributes to a more meaningful interpretation of social media discourse as a whole.

In essence, the proposed approach represents a sophisticated and effective strategy for extracting valuable insights from social media content. By incorporating a multi-dimensional analysis framework that accounts for both textual content and supplementary attributes, it offers a robust solution for sentiment analysis across diverse domains, ultimately enhancing the quality and reliability of derived insights.

REFERENCES

[1] Daniele Cenni, Paolo Nesi, Gianni Pantaleo, Imad Zaza., “Twitter vigilance: A multi-user platform for cross-domain Twitter data analytics, NLP and sentiment analysis”, In Proceedings of the IEEE International Conference.

[2] Oussalah M, Bhat F, Challis K, Schnier T. A software architecture for Twitter collection, search and geolocation services, In Knowledge-Based Systems. Vol. 37, pp.105-120, 2013.

[3] Alexandre Trilla, Francesc Alias, "Sentence- Based Sentiment Analysis For Expressive Text-To-Speech, IEEE Transactions on Audio, Speech, and Language Processing (Volume: 21 , Issue: 2 , Feb. 2013).

[4] Shulong Tan, Yang Li, Huan Sun, Ziyu Guan, Xifeng Yan, Jiajun Bu, Chun Chen Xiaofei He, "Interpreting The Public Sentiment Variations On Twitter", IEEE Transactions on Knowledge and Data Engineering (Volume: 26 , Issue: 5 , May 2014).

[5] Desheng Dash Wu, Lijuan Zheng, David L. Olson, “A Decision Support Approach For Online Stock Forum Sentiment Analysis”, IEEE Transactions on Systems, Man, and Cybernetics: Systems (Volume: 44, Issue: 8 , Aug. 2014).

[6] Ruhi U., Social Media Analytics as a Business Intelligence Practice: Current Landscape & Future Prospects, In Journal of Internet Social Networking & Virtual Communities, 2014.

[7] Shokoufeh Salem Minab, Mehrdad Jalali, Mohammad Hossein Moattar, “Online Analysis Of Sentiment On Twitter”, 2015 International Congress on Technology, Communication and Knowledge (ICTCK).

[8] Lavika Goel, Anurag Prakash, “Sentiment Analysis of Online Communities Using Swarm Intelligence Algorithms”, 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN).

[9] Lu Ma, Dan Zhang, Jian-wu Yang, Xiong Luo., “Sentiment Orientation Analysis Of Short Text Based On Background And Domain Sentiment Lexicon Expansion”, 2016 5th International Conference on Computer Science and Network Technology (ICCSNT).

[10] H. Sankar, V. Subramaniyaswamy, "Investigating Sentiment Analysis Using Deep Learning Approach", International Conference on Intelligent Sustainable Systems (ICISS) (2017).

[11] K. Young, M. Pistner, J. O'Mara, and J. Buchanan. Cyber-disorders: The mental health concern for the new millennium. Cyberpsychol. Behav., 2019..

[12] J. Block. Issues of DSM-V: internet addiction. American Journal of Psychiatry, 2019.

[13] K. Young. Internet addiction: the emergence of a new clinical disorder, Cyberpsychol. Behav., 2019.

[14] I.-H. Lin, C.-H. Ko, Y.-P. Chang, T.-L. Liu, P.-W. Wang, H.-C. Lin, M.-F. Huang, Y.-C. Yeh, W.-J. Chou, and C.-F. Yen. The associa- tion between suicidality and Internet addiction and activities in Taiwanese adolescents. Compr. Psychiat., 2019.

[15] Y. Baek, Y. Bae, and H. Jang. Social and parasocial relationships on social network sites and their differential relationships with users' psychological well-being. Cyberpsychol. Behav. Soc. Netw., 2019.

[16] D. La Barbera, F. La Paglia, and R. Valsavoia. Social network and addiction. Cyberpsychol. Behav., 2019.

[17] K. Chak and L. Leung. Shyness and locus of control as predictors of internet addiction and internet use. Cyberpsychol. Behav., 2019.

[18] K. Caballero and R. Akella. Dynamically modeling patients health state from electronic medical records: a time series approach. KDD, 2019.

[19] L. Zhao and J. Ye and F. Chen and C.-T. Lu and N. Ramakr- ishnan. Hierarchical Incomplete multi-source feature learning for Spatiotemporal Event Forecasting. KDD, 2019.

[20] E. Baumer, P. Adams, V. Khovanskaya, T. Liao, M. Smith, V. Sosik, and K. Williams. Limiting, leaving, and (re)lapsing: an exploration of Facebook non-use practices and experiences. CHI, 2019.