Nlp-Based Extended Lexicon Model For Sarcasm Detection With Tweets And Emojis

INNOVATIVE PRODUCT DEVELOPMENT REPORT

Submitted by

S. Sindhuja	**Nusrath Parveen**	**N. Srija**
22RH1A66G6	**22RH1A66D4**	**22RH1A66D1**

Sravani

22RH1A66H1

Under the Esteemed Guidance of

Mrs. Hema Malini

Assistant Professor

in partial fulfillment of the Academic Requirements for the Degree of

BACHELOR OF TECHNOLOGY

CSE-AI & ML

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

Autonomous Institution, UGC, Govt. of India
Accredited by NAAC with A+ Grade,UGC,Govt. of India
Permanently Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institute
Maisammaguda,Dhullapally , Secundrabad,Kompally-500100

June- 2024

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NAAC with 'A+' Grade, UGC, Govt. of India

Permanently Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institution

Maisammaguda, Dhullapally, Secunderabad, Kompally-500100

DEPARTMENT OF CSE - AI &ML

CERTIFICATE

This is to certify that the Innovative Product Development work entitled **Nlp-Based Extended Lexicon Model For Sarcasm Detection With Tweets And Emojis** is carried out by **S. Sindhuja (22RH1A66G6), Nusrath Parveen (22RH1A66D4), N. Srija(22RH1A66D1), Sravani (22RH1A66H1)** in partial fulfillment for the award of degree of **BACHELOR OF TECHNOLOGY** in CSE - AI & ML, Jawaharlal Nehru Technological University, Hyderabad during the academic year 2022-2023.

Supervisor's Signature **Head of the Department**

Mrs. Hema Malini
Asst. Professor

Dr. G. Kalpana
Professor

External Examiner

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN
Autonomous Institution, UGC, Govt. of India
Accredited by NAAC with A+ Grade,UGC,Govt. of India
Permanently Affiliated to JNTUH, Approved by AICTE, ISO 9001:2015 Certified Institute
Maisammaguda, Dhullapally, Secundrabad,Kompally,500100

Department of CSE –AI & ML

DECLARATION

We hereby declare that the Innovative Product Development entitled **Nlp-Based Extended Lexicon Model For Sarcasm Detection With Tweets And Emojis** submitted to Malla Reddy Engineering College For Women affiliated to Jawaharlal Nehru Technological University, Hyderabad (JNTUH) for the award of the Degree of Bachelor of Technology in CSE-AI & ML is a result of original research work done by us. It is further declared that the Innovative Product Development report or any part thereof has not been previously submitted to any University or Institute for the award of Degree.

S. Sindhuja(22RH1A66G6),

Nusrath Praveen(22RH1A66D4),

N.Srija(22RH1A66D1),

Sravani(22RH1A66H1)

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college **Malla Reddy Engineering College for Women** and **Department of CSE-AI&ML** which gave us the opportunity to have expertise in engineering and profound technical knowledge.

We would like to deeply thank our Honorable MLA of Telangana State **Sri**.**Ch. Malla Reddy Garu,** founder chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project success.

We wish to convey gratitude to our **Principal Dr. Y. Madhavee Latha,** for providing us with the environment and mean to enrich our skills and motivating us in our endeavor and helping us to realize our full potential.

We would like to thank **Prof. A. Radha Rani**, Director of Computer Science and Engineering & Information Technology for encouraging us to take up a project on this subject and motivating us towards the Project Work.

We express our sincere gratitude to **Dr.G.Kalpana , Head of the Department** of CSE–AI&ML for inspiring us to take up a project on this subject and successfully guiding us towards its completion.

We would like to thank our guide **Mrs. P. Gayatri, Asst. Professor of CSE-AI&ML** Dept and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

With Regards and Gratitude

S. Sindhuja(22RH1A66G6),

Nusrath Parveen(22RH1A66D4),

N. Srija (22RH1A66D1),

Sravani(22RH1A66H1)

ABSTRACT

Lexicon algorithm is used to determine the sentiment expressed by a textual content. This sentiment might be negative, neutral, or positive. It is possible to be sarcastic using only positive or neutral sentiment textual contents. Hence, lexicon algorithm can be useful but insufficient for sarcasm detection. It is necessary to extend the lexicon algorithm to come up with systems that would be proven efficient for sarcasm detection on neutral and positive sentiment textual contents. In this paper, two sarcasm analysis systems both obtained from the extension of the lexicon algorithm have been proposed for that sake. The first system consists of the combination of a lexicon algorithm and a pure sarcasm analysis algorithm. The second system consists of the combination of a lexicon algorithm and a sentiment prediction algorithm. Finally, naive bayes are used to predict sarcasm detection using pretrained features.

INDEX

TITLE	i
CERTIFICATE	ii
DECLARATION	iii
ACKNOWLEDGMENT ABSTRACT	iv
1. INTRODUCTION	1
1.1 Project Definition	1
1.2 Project Overview	1
1.3 Software Requirements	1
1.4 Hardware Requirements	1-2
2. LITERATURE SURVEY	3
2.1 Existing System	3
2.2 Disadvantages	3
2.3 Proposed System	3
2.4 Advantages	4
3. METHODOLOGY	3
4 SYSTEM ARCHITECTURE	9
5. IMPLEMENTATION	13
6. RESULTS	27
7. CONCLUSION & FUTURE SCOPE	31
8. REFERENCES	32

1.INTRODCUTION

1.1 PROJECT DEFINITION

The NLP-based external lexicon model for sarcasm detection with tweets and emojis aims to develop an automated system capable of identifying sarcasm in tweets by analyzing both the textual content and accompanying emojis. The project begins with collecting a substantial dataset of labeled tweets, distinguishing between sarcastic and non-sarcastic tweets and ensuring a diverse representation of emojis. The preprocessing stage involves cleaning the text, tokenizing, lemmatizing, and processing emojis to extract meaningful representations. Feature engineering focuses on leveraging external lexicons for sentiment analysis, extracting emoji features, and considering contextual features such as word context and syntactic patterns related to sarcasm. The modeling phase encompasses traditional classifiers like SVM and Logistic Regression, alongside advanced models like LSTM and Transformer, with ensemble methods for enhanced performance. Evaluation metrics include accuracy, precision, recall, and F1-score, with ethical considerations addressing bias, privacy, and transparency. Deployment involves API development for real-time sarcasm detection and integration into applications, culminating in comprehensive documentation, reporting, and avenues for future enhancements such as continual learning and multimodal analysis.

PROJECT OVERVIEW

The project focuses on developing a sarcasm detection system for tweets using natural language processing (NLP) techniques and external lexicons, with a specific emphasis on interpreting emojis as contextual cues. By collecting and preprocessing a diverse dataset of labeled tweets, leveraging sentiment analysis from external lexicons, and incorporating emoji features, the project aims to build accurate models for identifying sarcastic tweets. Evaluation metrics and ethical considerations are integral parts of the project, ensuring robustness, fairness, and transparency in the final system. Deployment involves creating an API for real-time sarcasm detection, offering practical applications in social media monitoring and sentiment analysis.

1.3 SOFTWARE REQUIREMENTS:

Operating System	-	Windows 10 or above
Programming Language	-	python

1.4 HARDWARE REQUIREMENTS

Processor	-	Intel i3 or i4
Speed	-	1.1 GHz
RAM	-	4 GB (min)

Hard Disk	-	500 GB (min)
Key Board	-	Standard Windows Keyboard
Mouse	-	Two or Three Button Mouse
Monitor	-	SVGA

2. LITERATURE SURVEY

2.1 Existing System:

The existing system for sarcasm detection in tweets typically relies on machine learning algorithms and natural language processing (NLP) techniques. These systems often use features like word embeddings, sentiment analysis, syntactic patterns, and context analysis to identify sarcastic language. However, many current systems struggle with accurately interpreting sarcasm, especially in contexts involving emojis, cultural nuances, and evolving language trends. Emojis, for instance, can significantly alter the meaning of a tweet and pose challenges for traditional models. Additionally, ethical considerations such as bias mitigation, privacy protection, and model transparency are increasingly becoming essential aspects of sarcasm detection systems. Hence, while existing systems provide a foundation, there is ongoing research and development to enhance accuracy, address complexities like emoji interpretation, and ensure ethical use of these systems in real-world applications. 2.2 Proposed System:

2.3 Advantages:

- Improves communication understanding.
- Provides valuable insights for analysis and research.
- Enhances user experience and engagement.
- Increases automation and efficiency.

2.4 Disadvantages:

- Complexity in interpretation due to nuances.
- Challenges with emojis, slang, and cultural differences.
- Risks bias if not trained on diverse datasets.
- Raises privacy concerns regarding data usage.

3.METHODOLOGY

Lexicon algorithm is used to determine the sentiment expressed by a textual content. This sentiment might be negative, neutral, or positive. It is possible to be sarcastic using only positive or neutral sentiment textual contents. Hence, lexicon algorithm can be useful but yet insufficient for sarcasm detection. It is necessary to extend the lexicon algorithm to come out with systems that would be proven efficient for sarcasm detection on neutral and positive sentiment textual contents. In this paper, two sarcasm analysis systems both obtained from the extension of the lexicon algorithm have been proposed for that sake. The first system consists of the combination of a lexicon algorithm and a pure sarcasm analysis algorithm. The second system consists of the combination of a lexicon algorithm and a sentiment prediction algorithm. In this work, the lexicon algorithm has been extended in two ways to generate two systems that could be more efficient for sarcasm analysis, especially on neutral and positive sentiment textual contents.

Advantages

- Improves communication understanding.
- Provides valuable insights for analysis and research.
- Enhances user experience and engagement.
- Increases automation and efficiency.

3.3. PROCESS MODEL USED WITH JUSTIFICATION

First system The first system (Fig. 1) is the combination of a lexicon algorithm and a pure sarcasm analysis algorithm. This system takes textual contents as input. These contents could be from various social media platforms like Twitter or Facebook. The textual contents are parsed into the lexicon algorithm for polarity computation. Then the positive sentiment contents are parsed into the pure sarcasm analysis algorithm for sarcasm detection. The final output of this system is a list of sarcastic and nonsarcastic lingual contents

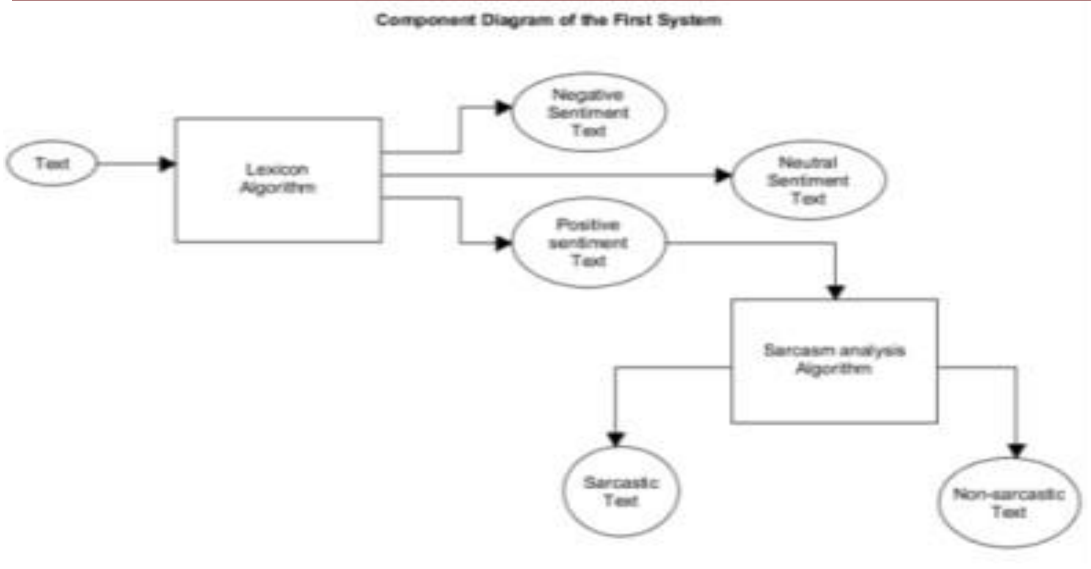


Fig. 1: Component diagram of the first system. An overview of the arrangement of components of the first system.

3.4 Second system

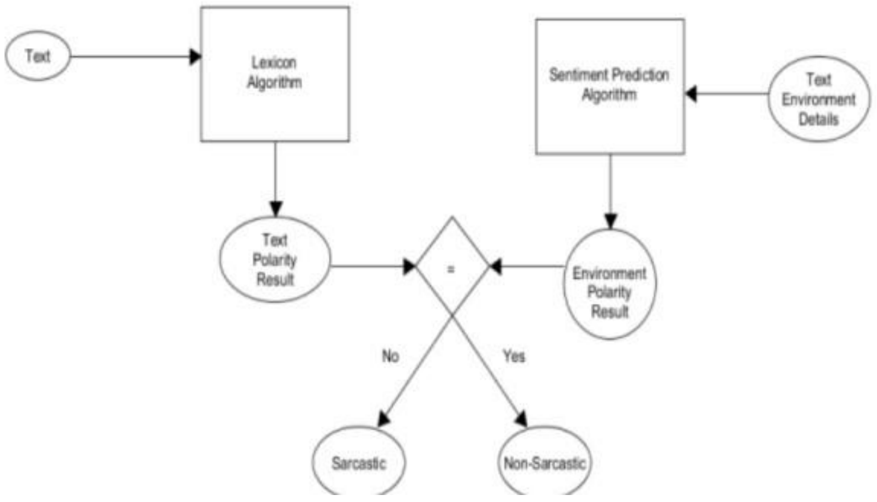
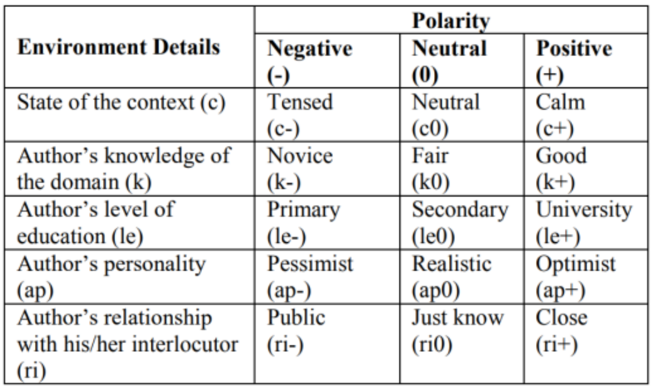


Fig. 2: Component diagram of the second system. An overview of the arrangement of the components of the second system.

The second system is the combination of a lexicon algorithm and a sentiment prediction algorithm. The lexicon algorithm is used here the same way as in the first system. The sentiment prediction algorithm consists of a mechanism that can predict the sentiment of a textual content that would be made under a specific environment. The sentiment prediction algorithm takes as input the details of the environment under which a lingual content would be made notably the state of the context, the author’s knowledge of the domain he/she would talk about, the author’s level of education, the author’s personality, the author’s relationship with his/her interlocutor. The sentiment prediction algorithm processes these details and predicts the sentiment of the textual content that would be formed under that environment. The results from both the algorithms are compared. In Case the results are different for a textual content, this later is classified as sarcastic else it is classified as nonsarcastic (Fig. 2). These environment details are processed based on a training data set that was formed as folloyw:

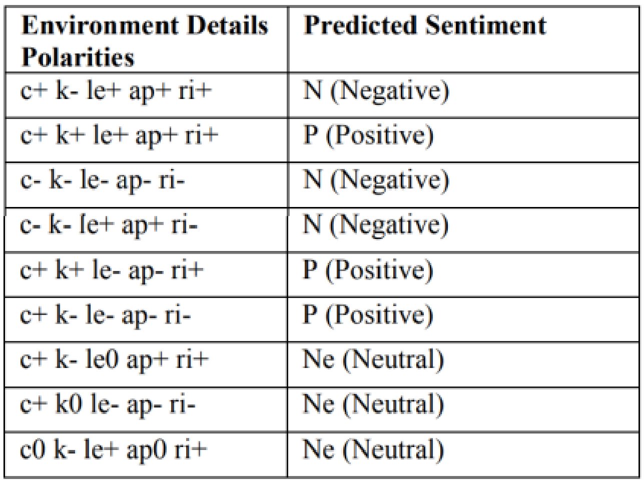
Each environment detail had a polarity. The polarity could be negative, neutral or positive (Table 1).

- An AND operation was performed in between the detail's polarities of each textual content environment to predict the sentiment of the textual content that would be made under each environment. The training data set of the sentiment prediction algorithm was formed of the environment details polarities and their predicted sentiment (Table 2) All the algorithms used in this paper have been derived from the classic Naïve Bayes algorithm. This algorithm makes use of conditional probability to predict the likeness of future occurrence of events based on their historical information. Naïve Bayes is mainly used for classification purposes. It is an algorithm that discriminates different objects based on certain features. This algorithm is built after the Bayes theorem which assumes that all features within a class are independent from one another and that is why it is known as 'naive'. Table. 1: Environment details and their polarity values



Environment Details	Polarity		
	Negative (-)	Neutral (0)	Positive (+)
State of the context (c)	Tensed (c-)	Neutral (c0)	Calm (c+)
Author's knowledge of the domain (k)	Novice (k-)	Fair (k0)	Good (k+)
Author's level of education (le)	Primary (le-)	Secondary (le0)	University (le+)
Author's personality (ap)	Pessimist (ap-)	Realistic (ap0)	Optimist (ap+)
Author's relationship with his/her interlocutor (ri)	Public (ri-)	Just know (ri0)	Close (ri+)

Naïve Bayes is mainly used for classification purposes. It is an algorithm that discriminates different objects based on certain features [11]. This algorithm is built after the Bayes theorem which assumes that all features within a class are independent from one another and that is why it is known as 'naive' [12]. There are several types of Naïve Bayes models [12][13]: - Gaussian Naïve Bayes: where the predictors take up continuous value and are not discrete. - Bernouilli Naïve Bayes: where the parameters of the predictors are Boolean values; 'yes', 'no', '1' or '0'. - Multinomial Naïve Bayes:
it is the generalization of Bernouilli where the features used by the classifier are the frequency of objects being processed. Multinomial Naïve Bayes is the model used in this paper. The steps of the Naïve Bayes algorithm can be resumed to the following [12]: Convert the data set into a frequency table, Create a likelihood table, and Use Naive Bayesian equation to calculate the posterior probability for each class.



Environment Details Polarities	Predicted Sentiment
c+ k- le+ ap+ ri+	N (Negative)
c+ k+ le+ ap+ ri+	P (Positive)
c- k- le- ap- ri-	N (Negative)
c- k- le+ ap+ ri-	N (Negative)
c+ k+ le- ap- ri+	P (Positive)
c+ k- le- ap- ri-	P (Positive)
c+ k- le0 ap+ ri+	Ne (Neutral)
c+ k0 le- ap- ri-	Ne (Neutral)
c0 k- le+ ap0 ri+	Ne (Neutral)

3.3 Naïve Bayes

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features. For some types of probability models, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without accepting Bayesian probability or using any Bayesian methods. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers.[6] Still, a comprehensive comparison with other classification algorithms in 2006 showed that Bayes classification is outperformed by other approaches, such as boosted trees or random forests.[7] An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification. This article discusses the theory behind the Naive Bayes classifiers and their implementation. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. To start with, let us consider a dataset. Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for plaing golf. Here is a tabular representation of our dataset.

4. SYSTEM ARCHITECTURE

UML Diagram:

Class Diagram:

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts

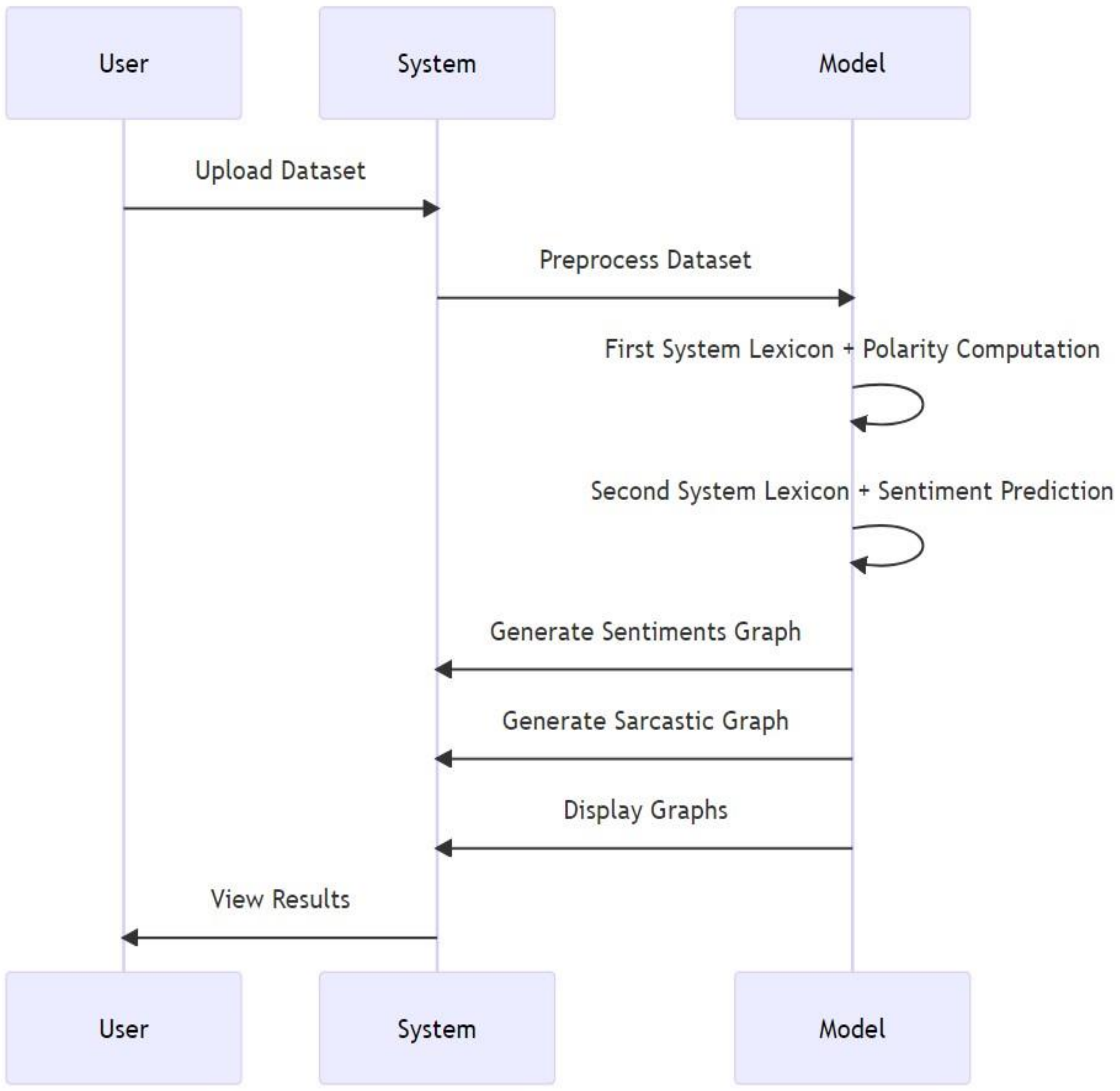
- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

Class Diagram:

```
---------------------------------------
|              Tweet                  |
--------------------------------------
|    - text: string                   |
|    - is_sarcastic: bool             |
|    - emojis: List[string]           |
 -------------------------------------
|     + get_text(): string            |
|     + get_emojis(): List[string     |
|      + is_sarcastic(): bool         |
---------------------------------------
|          SarcasmDetector            |
----------------------------------------
|       - model: Model                |
---------------------------------------
|       + train_model(data): void     |
| + detect_sarcasm(tweet): bool       |
---------------------------------------
```

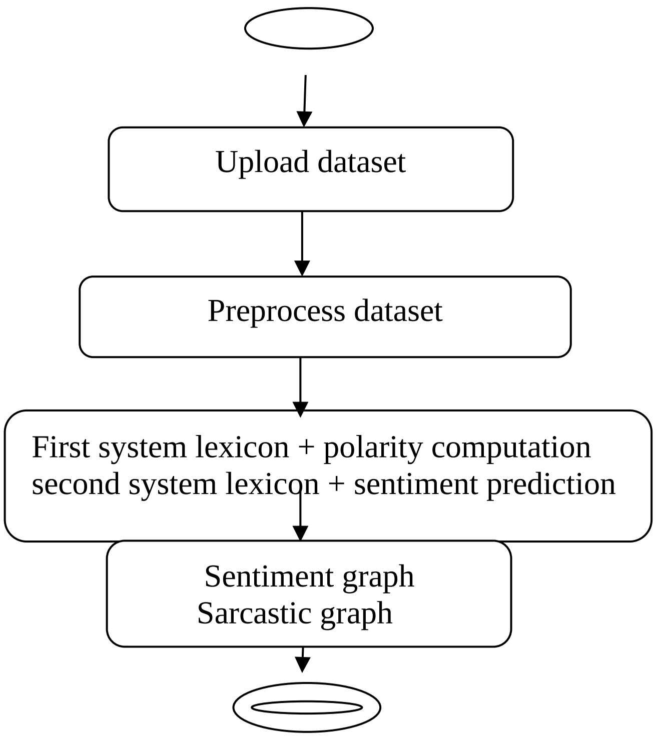
Sequence diagram:

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event senarios, and timing diagrams.


User
System
Model
Upload Dataset
Preprocess Dataset
First System Lexicon + Polarity Computation
Second System Lexicon + Sentiment Prediction
Generate Sentiments Graph
Generate Sarcastic Graph
Display Graphs
View Results
User
System
Model

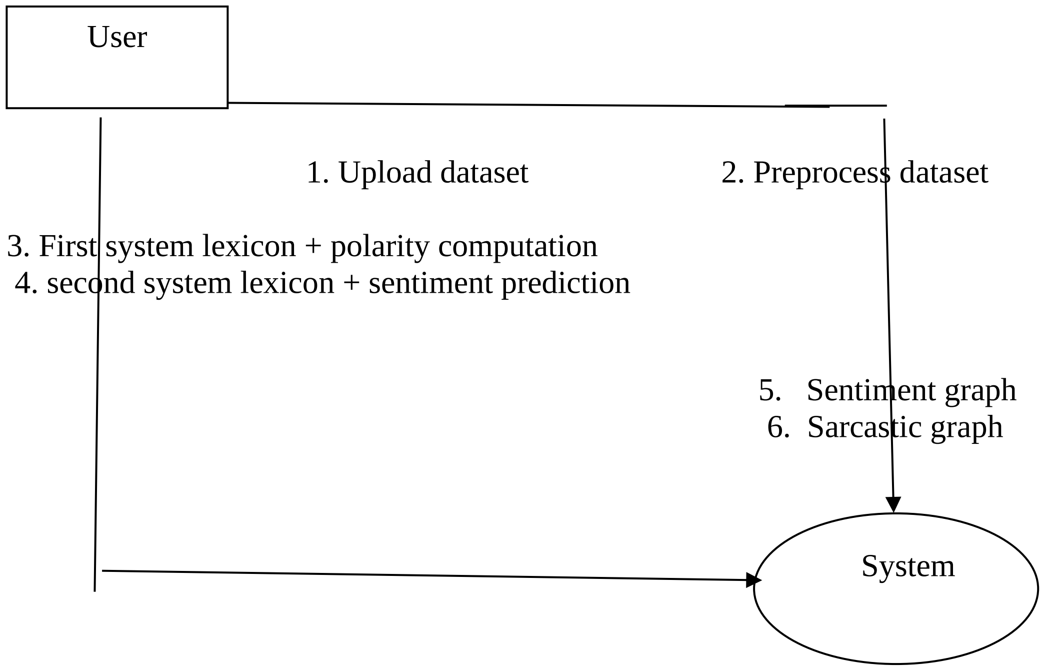
Activity Diagram:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent


Upload dataset
Preprocess dataset
First system lexicon + polarity computation
second system lexicon + sentiment prediction
Sentiment graph
Sarcastic graph

Data Flow Diagram:

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.


User
1. Upload dataset
2. Preprocess dataset
3. First system lexicon + polarity computation
4. second system lexicon + sentiment prediction
5. Sentiment graph
6. Sarcastic graph
System

5.IMPLEMENTATION

5.1 Python

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

History of Python:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

Why Python was created?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

Why the name Python?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Features of Python:

A simple language which is easier to learn

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

Free and open-source

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

Portability

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

Extensible and Embeddable

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

A high-level, interpreted language

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.

Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

Large standard libraries to solve common tasks

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb .

Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

Applications of Python:

1. Simple Elegant Syntax

Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

```
a = 2

b = 3

sum = a + b

print(sum)
```

2. Not overly strict

You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.

Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

3. Expressiveness of the language

Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

4. Great Community and Support

Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck.

SOURCE CODE

5.2 Sample Code;

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
import matplotlib.pyplot as plt
import numpy as np
from tkinter import filedialog
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from string import punctuation
from nltk.corpus import stopwords
import pandas as pd
from emoji import UNICODE_EMOJI
main = tkinter.Tk()
main.title("NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis")
#designing main screen
main.geometry("1300x1200")
sid = SentimentIntensityAnalyzer()
global filename
global dataset
global process
global sarcastic
global sentiment
def checkSarcasm(sentence):
    pos = []
    neg = []
    neu = []
    arr = sentence.split(' ')
    for i in range(len(arr)):
        word = arr[i].strip()
        if word == 'smilingfacewithhearteyes':
            word = 'excellent'
        if word == 'loudlycryingface':
            word = 'bad'
        if word == 'winkingfacewithtongue':
            word = 'happy'
        if (sid.polarity_scores(word)['compound']) >= 0.1:
            pos.append(word)
        elif (sid.polarity_scores(word)['compound']) <= -0.1:
            neg.append(word)
        else:
            neu.append(word)
    return pos,neg,neu
def clean_doc(doc):
    tokens = doc.split()
    table = str.maketrans('', '', punctuation)
    tokens = [w.translate(table) for w in tokens]
    tokens = [word for word in tokens if word.isalpha()]
    stop_words = set(stopwords.words('english'))
    tokens = [w for w in tokens if not w in stop_words]
    tokens = [word for word in tokens if len(word) > 1]
    tokens = ' '.join(tokens) #here upto for word based
    return tokens
def upload():
    global filename
    global dataset
    dataset = []
    filename = filedialog.askopenfilename(initialdir="dataset")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");
    train = pd.read_csv(filename,encoding='utf8',sep='\t')
    count = 0
    for i in range(len(train)):
        tweet = train.get_value(i,0,takeable = True)
        print(tweet)
        if str(tweet) != 'nan':
            tweet = tweet.lower()
        icon = train.get_value(i,1,takeable = True)
        if str(icon) != 'nan':
            icon = UNICODE_EMOJI[icon.strip()]
            icon = ''.join(re.sub('[^A-Za-z\s]+', '', icon))
            icon = icon.lower()
        else:
            icon = ''
        msg = ''
        if str(tweet) != 'nan':
            arr = tweet.split(" ")
            for k in range(len(arr)):
                word = arr[k].strip()
                if len(word) > 2:
                    msg+=word+" "
        textdata = msg.strip()+" "+icon
        #print(textdata)
        dataset.append(textdata)

    text.insert(END,'Total tweets found in dataset is : '+str(len(dataset)))

def Preprocessing():
    text.delete('1.0', END)
    global process
    process = []
    text.insert(END,'Messages after preprocessing and removing stopwords\n')

text.insert(END,'=============================================================
=========================\n')

    for i in range(len(dataset)):
        sentence = dataset[i]
        sentence = sentence.lower()
        sentence = clean_doc(sentence)
        text.insert(END,sentence+'\n')
        process.append(sentence)
```

```
def firstAlgorithm():
    text.delete('1.0', END)
    global sarcastic
    sarcastic = []
    for i in range(len(process)):
        sentence = process[i]
        if sentence == 'smilingfacewithhearteyes':
            sentence = 'excellent'
        if sentence == 'loudlycryingface':
            sentence = 'bad'
        if sentence == 'winkingfacewithtongue':
            sentence = 'happy'
        sentiment_dict = sid.polarity_scores(sentence)
        negative_polarity = sentiment_dict['neg']
        positive_polarity = sentiment_dict['pos']
        neutral_polarity = sentiment_dict['neu']
        compound = sentiment_dict['compound']
        result = ''
        if compound >= 0.1 :
            result = 'Positive'
        elif compound <= -0.1:
            result = 'Negative'
        else :
            result = 'Neutral'
        if result =='Positive' or result == 'Neutral':
            pos,neg,neu = checkSarcasm(sentence)
            if len(neg) > 0:
                sarcastic.append("Sarcastic")
            else:
                sarcastic.append("Non Sarcastic")
        else:
            sarcastic.append("Non Sarcastic")
        text.insert(END,'Tweets : '+dataset[i]+"\n")
        text.insert(END,'Positive Polarity : '+str(positive_polarity)+"\n")
        text.insert(END,'Negative Polarity : '+str(negative_polarity)+"\n")
        text.insert(END,'Neutral Polarity  : '+str(neutral_polarity)+"\n")
        text.insert(END,'Result : '+sarcastic[i]+"\n")

text.insert(END,'=============================================================
=========================\n')

def secondAlgorithm():
    global sentiment
    sentiment = []
    text.delete('1.0', END)
    for i in range(len(process)):
        sentence = process[i]
        if sentence == 'smilingfacewithhearteyes':
            sentence = 'excellent'
        if sentence == 'loudlycryingface':
            sentence = 'bad'
        if sentence == 'winkingfacewithtongue':
            sentence = 'happy'
        sentiment_dict = sid.polarity_scores(sentence)
        negative_polarity = sentiment_dict['neg']
        positive_polarity = sentiment_dict['pos']
        neutral_polarity = sentiment_dict['neu']
        compound = sentiment_dict['compound']
        result = ''
        if compound >= 0.1 :
            result = 'Positive'
            sentiment.append(result)
        elif compound <= -0.1:
            result = 'Negative'
            sentiment.append(result)
        else :
            result = 'Neutral'
            sentiment.append(result)
        sar = ''
        if result =='Positive' or result == 'Neutral':
            pos,neg,neu = checkSarcasm(sentence)
            if len(neg) > 0:
                sar = "Sarcastic"
            else:
                sar = "Non Sarcastic"
        else:
            sar = "Non Sarcastic"

        text.insert(END,'Tweets : '+dataset[i]+"\n")
        text.insert(END,'Positive Polarity : '+str(positive_polarity)+"\n")
        text.insert(END,'Negative Polarity : '+str(negative_polarity)+"\n")
        text.insert(END,'Neutral Polarity  : '+str(neutral_polarity)+"\n")
        text.insert(END,'Result : '+sar+"\n")
        text.insert(END,'Sentiment Prediction : '+result+'\n')

text.insert(END,'===========================================================
==========================\n')

def sarcasticGraph():
    sar = 0
    non_sar = 0
    for i in range(len(sarcastic)):
        if sarcastic[i] == "Sarcastic":
            sar = sar + 1
        if sarcastic[i] == "Non Sarcastic":
            non_sar = non_sar + 1
    height = [sar,non_sar]
    bars = ('Sarcastic','Non Sarcastic')
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()

def sentimentGraph():
    label_X = []
    category_X = []
    pos = 0
    neg = 0
    neu = 0
    for i in range(len(sentiment)):
        if sentiment[i] == 'Positive':
            pos = pos + 1
        if sentiment[i] == 'Negative':
            neg = neg + 1
        if sentiment[i] == 'Neutral':
            neu = neu + 1
    label_X.append('Positive')
    label_X.append('Negative')
    label_X.append('Neutral')
    category_X.append(pos)
    category_X.append(neg)
    category_X.append(neu)

    plt.pie(category_X,labels=label_X,autopct='%1.1f%%')
    plt.title('Sentiment Graph')
    plt.axis('equal')
    plt.show()

font = ('times', 16, 'bold')
title = Label(main, text='NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets 
and Emojis')
title.config(bg='LightGoldenrod1', fg='medium orchid')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=100)
scroll=Scrollbar(text)
```

```
text.configure(yscrollcommand=scroll.set)
text.place(x=400,y=100)
text.config(font=font1)



font1 = ('times', 12, 'bold')
uploadButton = Button(main, text="Upload Dataset", command=upload)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

preButton = Button(main, text="Preprocess Dataset", command=Preprocessing)
preButton.place(x=50,y=150)
preButton.config(font=font1)

firstButton = Button(main, text="First System Lexicon + Polarity Computation",
command=firstAlgorithm)
firstButton.place(x=50,y=200)
firstButton.config(font=font1)

secondButton = Button(main, text="Second System Lexicon + Sentiment Prediction",
command=secondAlgorithm)
secondButton.place(x=50,y=250)
secondButton.config(font=font1)

graphButton = Button(main, text="Sentiments Graph", command=sentimentGraph)
graphButton.place(x=50,y=300)
graphButton.config(font=font1)

gButton = Button(main, text="Sarcastic Graph", command=sarcasticGraph)
gButton.place(x=50,y=350)
gButton.config(font=font1)
main.config(bg='OliveDrab2')
main.mainloop()
```

6.Implementation and Testing:

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit.Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence.

System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical.Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. For example ,the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually.

Test Case Id	Test Case Name	Test Case Desc.	Test Steps			Test Case Status	Test Priority
			Step	Expected	Actual		
01	Upload data set	Test whether the First Document uploaded or not	If the first document may not uploaded	We cannot do further process	The first document uploaded successfully	High	High
02	Preprocess dataset	Test whether the data preprocessed or not	preprocessing	We cannot do further operations	The Preprocessed successfully	High	High
03	. First system lexicon + polarity computation	Verify the polarity Computation of system	First system lexicon + polarity computation	We cannot do further process	Done sucessfully	High	High
03	Second system lexicon + sentiment prediction	Verify either sentiment prediction of second system	Second system lexicon + sentiment prediction	We cannot do further operations	We can do further operations	High	High
04	Sentiment graph Sarcastic graph	Verifing the sentiment and sarcastic graph	Sentiment graph and Sarcastic graph	We cannot do further operations	We can do further operations	High	High

OUTPUT

SCREENSHOTS

To run project double click on ‘run.bat’ file to get below screen

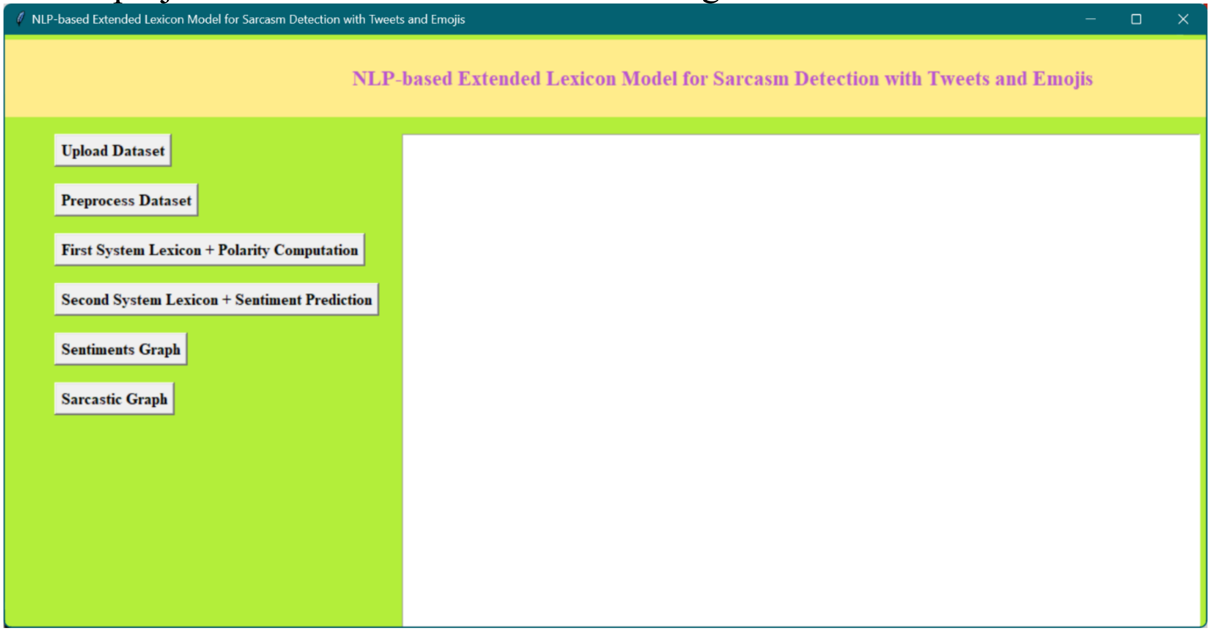
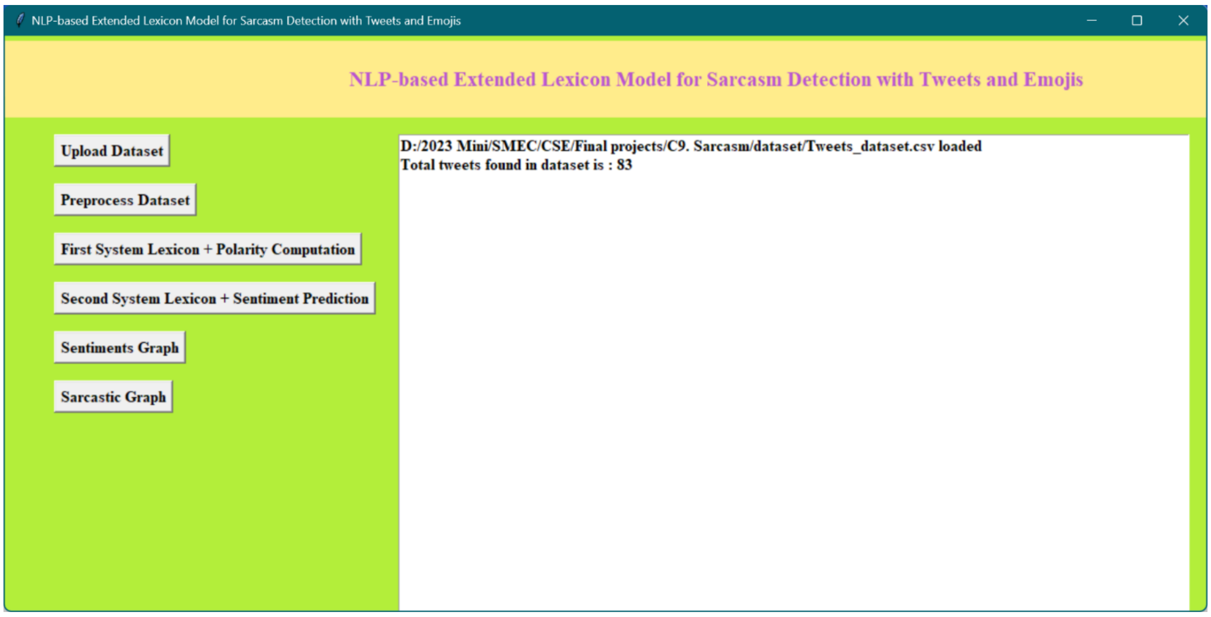

NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
Upload Dataset
Preprocess Dataset
First System Lexicon + Polarity Computation
Second System Lexicon + Sentiment Prediction
Sentiments Graph
Sarcastic Graph

Figure 1.1

Figure 1: User interface application of proposed NLP-based sarcasm detection model.

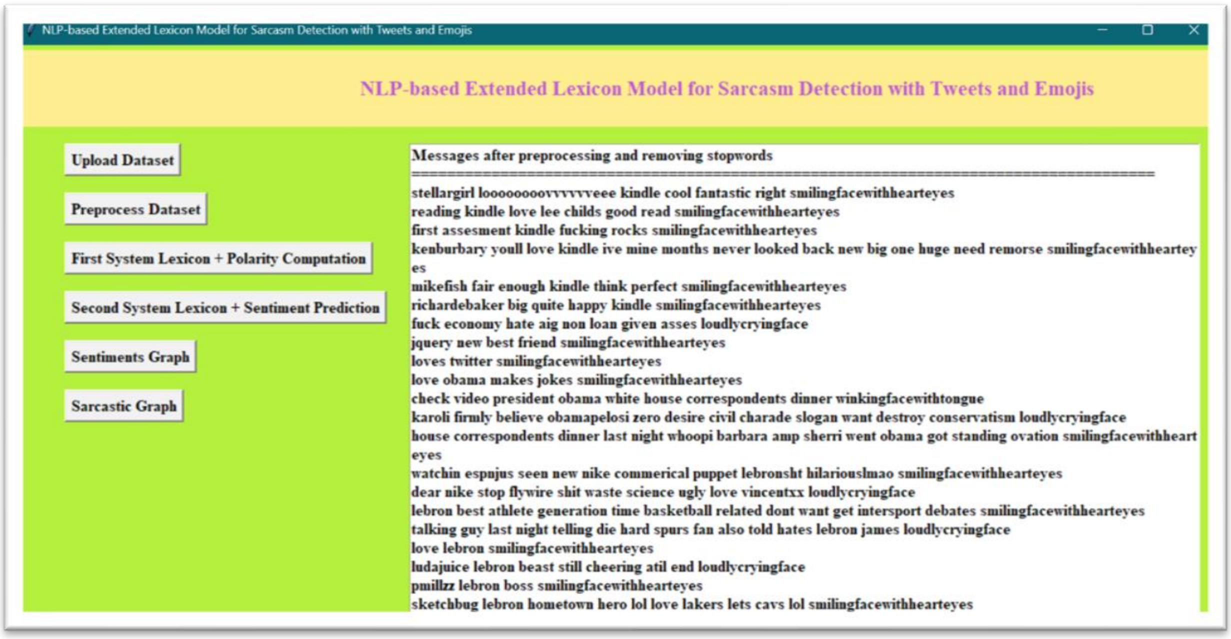

NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
Upload Dataset
Preprocess Dataset
First System Lexicon + Polarity Computation
Second System Lexicon + Sentiment Prediction
Sentiments Graph
Sarcastic Graph

```
D:/2023 Mini/SMEC/CSE/Final projects/C9. Sarcasm/dataset/Tweets_dataset.csv loaded
Total tweets found in dataset is : 83
```

Figure 1.2

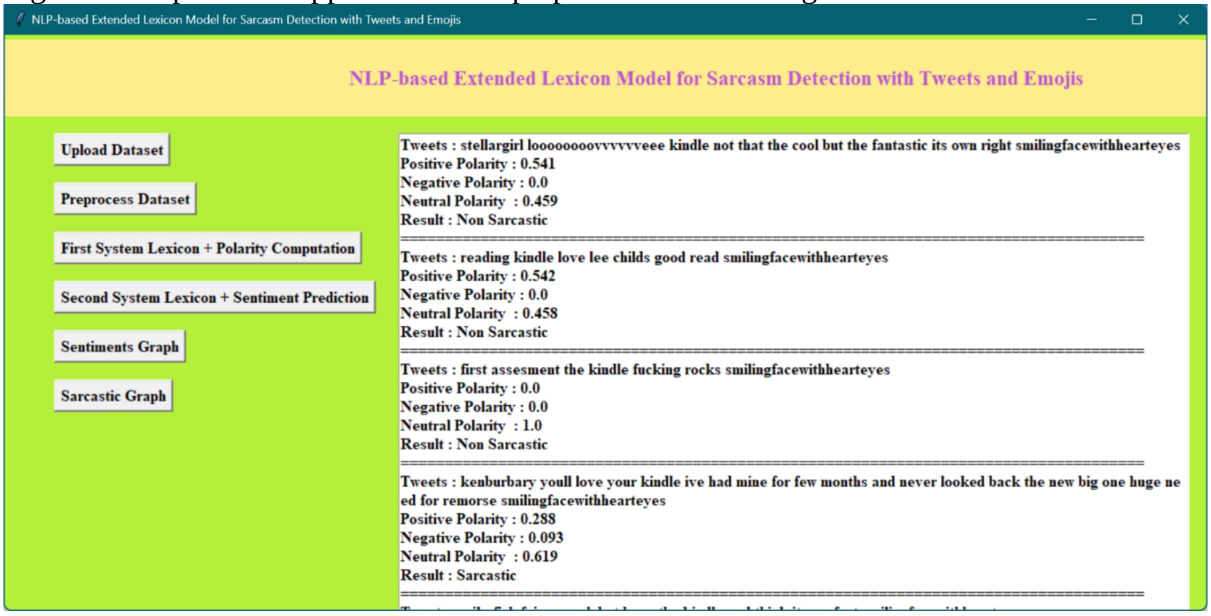
Figure 2: Illustration of user interface application after uploading the dataset.

Figure 1.3


NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
Upload Dataset
Preprocess Dataset
First System Lexicon + Polarity Computation
Second System Lexicon + Sentiment Prediction
Sentiments Graph
Sarcastic Graph

```
Messages after preprocessing and removing stopwords
==========================================================================================
stellargirl loooooooovvvvvveee kindle cool fantastic right smilingfacewithhearteyes
reading kindle love lee childs good read smilingfacewithhearteyes
first assesment kindle fucking rocks smilingfacewithhearteyes
kenburbary youll love kindle ive mine months never looked back new big one huge need remorse smilingfacewithheartey
es
mikefish fair enough kindle think perfect smilingfacewithhearteyes
richardebaker big quite happy kindle smilingfacewithhearteyes
fuck economy hate aig non loan given asses loudlycryingface
jquery new best friend smilingfacewithhearteyes
loves twitter smilingfacewithhearteyes
love obama makes jokes smilingfacewithhearteyes
check video president obama white house correspondents dinner winkingfacewithtongue
karoli firmly believe obamapelosi zero desire civil charade slogan want destroy conservatism loudlycryingface
house correspondents dinner last night whoopi barbara amp sherri went obama got standing ovation smilingfacewithheart
eyes
watchin espnjus seen new nike commerical puppet lebronsht hilariouslmao smilingfacewithhearteyes
dear nike stop flywire shit waste science ugly love vincentxx loudlycryingface
lebron best athlete generation time basketball related dont want get intersport debates smilingfacewithhearteyes
talking guy last night telling die hard spurs fan also told hates lebron james loudlycryingface
love lebron smilingfacewithhearteyes
ludajuice lebron beast still cheering atil end loudlycryingface
pmillzz lebron boss smilingfacewithhearteyes
sketchbug lebron hometown hero lol love lakers lets cavs lol smilingfacewithhearteyes
```

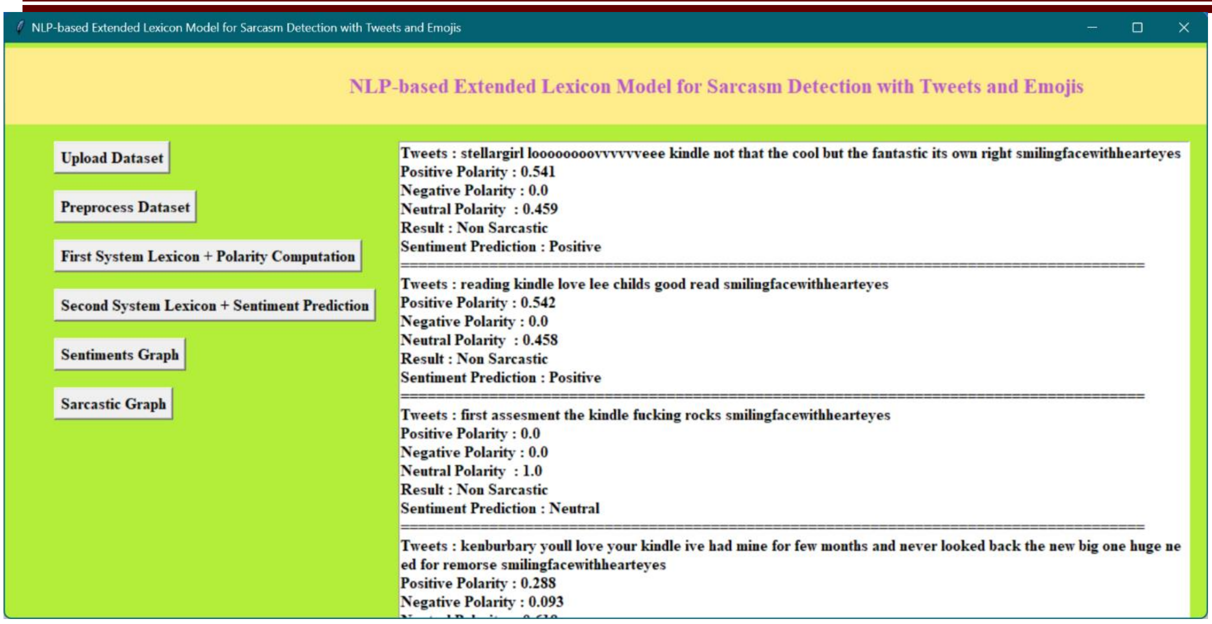
Figure 3: Proposed UI application with preprocessed data using NLTK


NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
Upload Dataset
Preprocess Dataset
First System Lexicon + Polarity Computation
Second System Lexicon + Sentiment Prediction
Sentiments Graph
Sarcastic Graph

```
Tweets : stellargirl loooooooovvvvvveee kindle not that the cool but the fantastic its own right smilingfacewithhearteyes
Positive Polarity : 0.541
Negative Polarity : 0.0
Neutral Polarity  : 0.459
Result : Non Sarcastic
==========================================================================================
Tweets : reading kindle love lee childs good read smilingfacewithhearteyes
Positive Polarity : 0.542
Negative Polarity : 0.0
Neutral Polarity  : 0.458
Result : Non Sarcastic
==========================================================================================
Tweets : first assesment the kindle fucking rocks smilingfacewithhearteyes
Positive Polarity : 0.0
Negative Polarity : 0.0
Neutral Polarity  : 1.0
Result : Non Sarcastic
==========================================================================================
Tweets : kenburbary youll love your kindle ive had mine for few months and never looked back the new big one huge ne
ed for remorse smilingfacewithhearteyes
Positive Polarity : 0.288
Negative Polarity : 0.093
Neutral Polarity  : 0.619
Result : Sarcastic
==========================================================================================
```

Figure 1.4

Figure 4: Obtained results of proposed UI application with lexicon + polarity computation.


NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
NLP-based Extended Lexicon Model for Sarcasm Detection with Tweets and Emojis
Upload Dataset
Preprocess Dataset
First System Lexicon + Polarity Computation
Second System Lexicon + Sentiment Prediction
Sentiments Graph
Sarcastic Graph

```
Tweets : stellargirl loooooooovvvvvveee kindle not that the cool but the fantastic its own right smilingfacewithhearteyes
Positive Polarity : 0.541
Negative Polarity : 0.0
Neutral Polarity  : 0.459
Result : Non Sarcastic
Sentiment Prediction : Positive
================================================================================
Tweets : reading kindle love lee childs good read smilingfacewithhearteyes
Positive Polarity : 0.542
Negative Polarity : 0.0
Neutral Polarity  : 0.458
Result : Non Sarcastic
Sentiment Prediction : Positive
================================================================================
Tweets : first assesment the kindle fucking rocks smilingfacewithhearteyes
Positive Polarity : 0.0
Negative Polarity : 0.0
Neutral Polarity  : 1.0
Result : Non Sarcastic
Sentiment Prediction : Neutral
================================================================================
Tweets : kenburbary youll love your kindle ive had mine for few months and never looked back the new big one huge ne
ed for remorse smilingfacewithhearteyes
Positive Polarity : 0.288
Negative Polarity : 0.093
```

Figure 1.5

Figure 5: Results of lexicon + sentiment prediction for proposed sarcasm detection model.

In Figure 5, the user interface presents the results of sentiment prediction using lexicon-based sentiment analysis. Alongside polarity scores, the UI also indicates whether a tweet is predicted as sarcastic or not based on sentiment and certain words. Figure 6 displays a pie chart that visualizes the distribution of sentiments in the dataset. The chart is divided into segments representing positive, negative, and neutral sentiments. The size of each segment indicates the proportion of tweets falling into each sentiment category. Figure 7 shows a graph that visualizes the sarcasm predictions made by the proposed NLP-based extended lexicon model. The graph likely has two bars: one representing the number of sarcastic tweets and the other representing the number of non-sarcastic tweets. This visualization provides insights into the model's ability to detect sarcasm.

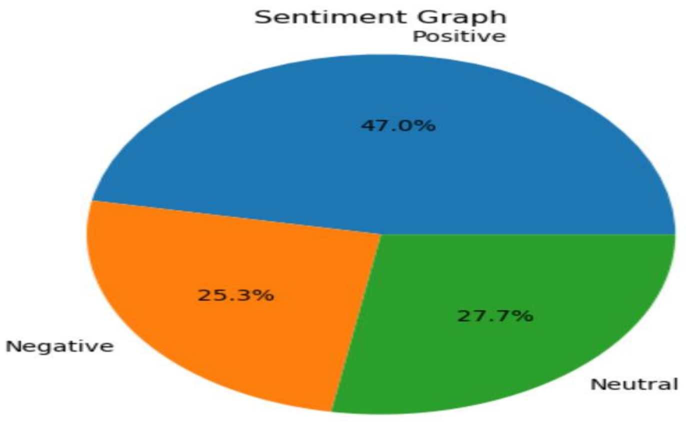


Figure 1.6

Figure 6: Pie chart of sentiment prediction on tweets dataset with different classes (positive, negative, and neutral).

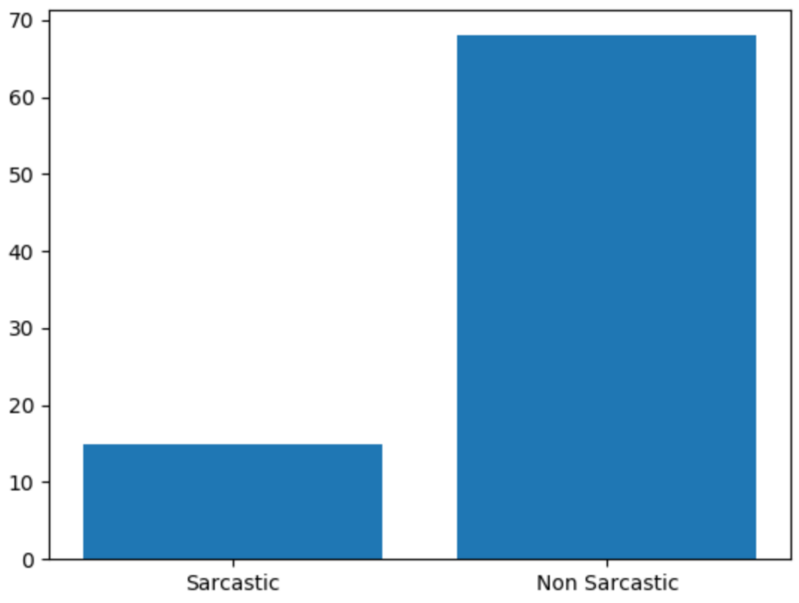


Fig 1.7

Figure 7: Sarcasm prediction graph using proposed NLP-based extended lexicon model.

7.CONCLUSION

The aim of this study was to propose ways to extend the lexicon algorithm to build systems that would be more efficient for sarcasm detection. This aim has been successfully met as two systems JOURNAL OF CRITICAL REVIEWS ISSN- 2394-5125 VOL 10, ISSUE 02, 2023 75 have been developed to address this situation. However, in the first system, it had been noticed that the training set of the sarcasm analysis algorithm must be relevant to the actual data that need to be analyzed to obtain meaningful results and to improve the accuracy of the system. The second system constitutes a vast area of study. Some work needs to be done to develop a system that would allow the collection of environmental details under which the textual contents would be made on social media platforms. A consolidated way of computing the sentiment polarity of the environments based on their details should also be developed.

FUTURE SCOPE

The future scope of this project could involve expanding the NLP-based extended lexicon model to detect sarcasm in other forms of text, such as comments or reviews. It could also explore incorporating other indicators of sarcasm, like tone of voice or context. Plus, it could be applied to different languages.

let's dive deeper into the future scope of the NLP-based extended lexicon model for sarcasm detection with tweets and emojis. Here are a few potential directions:

1**. Enhanced Model:** The project can focus on improving the accuracy and performance of the existing model by incorporating more advanced NLP techniques, such as deep learning algorithms or transformer models like BERT or GPT.

2. **Multilingual Support**: Expanding the model's capabilities to detect sarcasm in multiple languages would be a valuable addition. This would involve training the model on diverse language datasets and adapting it to different cultural contexts.

3. **Social Media Platforms:** The project could explore integrating the sarcasm detection model into various social media platforms beyond Twitter, such as Facebook, Instagram, or Reddit. This would help users better understand the intended meaning behind sarcastic posts across different platforms.

4. **Real-Time Detection**: Developing a real-time sarcasm detection system that can analyze and classify sarcastic tweets or comments as they are posted would be a valuable application. This could be useful for social media monitoring, sentiment analysis, or even content moderation.

8.REFERENCES

[1] Cambridge University Press, 2018. sarcasm. [Online] Available at: https://dictionary.cambridge.org/dictionary/english/sarc asm [Accessed 20 January 2018].

[2] Palanisamy, P., Yadav, V., & Elchuri, H. (2013). Serendio: Simple and Practical lexiconbased approach to Sentiment. 543-548.

[3] Jurek, A., Mulvenna, M. D., & Bi, Y. (2015). Improved lexicon-based sentiment analysis for social media analytics. SpringerOpen, 4-9.

[4] Kiilu, K. K., Okeyo, G., Rimiru, R., & Ogada, K. (2018). Using Naïve Bayes Algorithm in detection of Hate Tweets. International Journal of Scientific and Research Publications, 99- 107.

[5] Rathan, K., & Suchithra, R. (2017). Sarcasm detection using combinational. Imperial Journal of Interdisciplinary Research, 546-551.

[6] Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised. International Journal of Advanced Research in Artificial Intelligence, 34-38.

[7] Dataquest, 2018. Top 10 Machine Learning Algorithms for Beginners. [Online] Available at: https://www.dataquest.io/blog/top-10- machine-learning-algorithms-for-beginners/ [Accessed 15 September 2018].

[8] Haripriya, V., & Patil, D. P. (2017). A Survey of Sarcasm Detection in Social Media. International Journal for Research in Applied Science & Engineering Technology, 1748- 1753.

[9] Musto, C., Semeraro, G., & Polignano, M. (n.d.). A comparison of Lexicon-based approaches for Sentiment Analysis of microblog posts.

[10] Saxena, R., 2017. How the naive bayes classifier works in machine learning. [Online] Available at: http://dataaspirant.com/2017/02/06/naivebayes-classifier-machine-learning/ [Accessed 10 February 2019]. **[11]**Gandhi, R., 2018. Naive Bayes Classifier. [Online] Available at: https://towardsdatascience.com/naivebayes-classifier-81d512f50a7c [Accessed 10 February 2019].

[12] RAY, S., 2017. 6 Easy Steps to Learn Naive Bayes Algorithm (with codes in Python and R). [Online] Available at: https://www.analyticsvidhya.com/blog/2017/09/naivebayesexplained/ [Accessed 10 February 2019].

[13] Aggarwal, S., & Kaur, D. (2013). Naïve Bayes Classifier with Various Smoothing. International Journal of Computer Trends and Technology, 873-876.