

1. INTRODUCTION

Communication is the process of exchanging information. As time goes by, many ways and platforms of communication are being developed. Since the industrial revolution, the original way of communicating; face-to-face communication has been used as a model to develop the various ways of communicating known to date. Transposing the principles and codes of the natural face-to-face communication to today's online communication is a major challenge for developers. Sarcasm is the communication practice that consists of meaning the opposite of what is said in order to mock or insult someone. Sarcasm makes use of positive lingual contents in order to convey a negative message. Different types of approaches have been developed in order to implement sarcasm detection on online communication platforms. However the levels of efficiency of these approaches have been the principal worries of developers. In this paper, propositions are made on how the lexicon algorithm can be extended in order to come out with systems that would be proven more efficient for sarcasm detection on textual contents.

2. LITERATURE SURVEY

2.1 Serendio: Simple and Practical lexicon based approach to Sentiment.

AUTHORS: Prabu palanisamy, Vineet Yadav and Harsha Elchuri

In this paper we presented our system that we used for the SemEval-2013 Task 2 for doing Sentiment Analysis for Twitter data. We got an F-score of 0.8004 on the test data set. We presented a lexicon based method for Sentiment Analysis with Twitter data. We provided practical approaches to identifying and extracting sentiments from emoticons and hashtags. We also provided a method to convert non-grammatical words to grammatical words and normalize non-root to root words to extract sentiments. A lexicon based approach is a simple, viable and practical approach to Sentiment Analysis of Twitter data without a need for training. A Lexicon based approach is as good as the lexicon it uses. To achieve better results, word sense disambiguation should be combined with the existing lexicon approach.

2.2. Improved lexicon-based sentiment analysis for social media analytics.

AUTHORS: Anna Jurak*, Maurice D. Mulvenna and Yaxin Bi

In this work we presented a new approach to lexicon based sentiment analysis of Twitter messages. In the new approach, the sentiment is normalized, which allows us to obtain the intensity of sentiment rather than positive/negative decision. A new evidence-based combining function was developed in an effort to improve performance of the algorithm in the cases where a mixed sentiment occurs in a message. The evaluation was performed with the Stanford Twitter test set and IMDB data set. It was found from the results that the two new functions improve performance of the standard lexicon-based sentiment analysis algorithm. It could be noticed that the method is more appropriate for short messages such as tweets. When applied with long documents the method performed significantly better on the sentence than on the document level. Following this, our intention was to investigate the relationship between the amount and the level of negative sentiment related to a public demonstration and the level of violence and disorder

during the event. In other words, we aimed to ascertain if sentiment analysis could be applied as a supportive tool while predicting a level of disruption prior to public events. As a first step in this study we decided to examine Twitter as a source of data. Four different demonstrations were selected and the negative sentiment related to these events was analysed over 6 days prior to each event. Following the case study and a number of analyses we were able to reveal that there was a relationship to some extent between the negative sentiment and the level of disorder during the EDL events. Further research is however required in this area in an effort to provide more accurate findings and conclusions. At the current stage we can, however, conjecture that sentiment analysis of social media content can provide valuable, security-related information regarding some upcoming public events. In the next step we wish to collect more data related to public events and further investigate the relationship between negative sentiment and the level of violence and disorder during events. Following this, we aim to develop a predictive model that can be used by police services as a single tool to help indicate violence propensity. In future work we wish to focus more on multilingual sentiment analysis. Given that data pulled from social media are created by users from all over the globe, there is a consequent demand to perform sentiment analysis in more than just one language. The most challenging problem while trying to translate sentiment lexicon in a different language is inflection and conjugation of words applied in some of the languages. Unlike in English, some languages make use of grammatical gender and plural. Following this, verbs, nouns and adjectives are inflected for person or number and verbs are marked for tense. For example, while in English the verb “love” can be used in 4 different forms (love, loved, loving, loves), in Polish language there are 20 different forms depending of tense and person. Besides this, the adjective “nice”, for example, in Polish language can be used in 5 different forms. Consequently, it would be very inefficient to include all the different forms of words in the lexicon, especially when talking about real time analysis. In some preliminary work we were able to demonstrate that by application of an appropriate string similarity function it is possible to perform sentiment analysis with the lexicon containing only regular form of words. Another important issue while translating a lexicon into another language is disambiguation. It is important to ensure that for ambiguous words, the appropriate meanings have been translated and included into new lexicon. Consequently, an automatic translation may not provide the desired results. In our work, semi-automatic translation has been applied where all ambiguous words were translated manually. We were able

to show that by translating words from the English lexicon into regular Polish and Portuguese words and by application of a string similarity function, the sentiment analysis of Polish/Portuguese tweets can be performed on a similar level of accuracy as for the English language. At the same time, some preliminary experiments demonstrated that the proposed method could be easily adapted to languages such as Malay, where no inflection or conjugation is being applied to the words.

2.3. Using Naïve Bayes Algorithm in detection of Hate Tweets:

AUTHORS: Kiilu, K. K., Okeyo, G., Rimiru, R., & Ogada, K

The aim of the study was to evaluate the performance for sentiment classification in terms of accuracy, precision and recall. In this paper, we compared various supervised machine learning algorithms of Naïve Bayes' for sentiment analysis and detection of the hate tweets in twitter. Apart from the system's ability to predict for a given tweet whether it is hateful or not, the system also generates a list of users who frequently post such content. This provides us with an interesting insight into the usage pattern of hate-mongers in terms of how they express bigotry, racism and propaganda. The experimental results show that the classifiers yielded better results for the hate tweets review with the Naïve Bayes' approach giving above 80% accuracies and outperforming other algorithms. This research had a number of key weaknesses that can be addressed. One major consideration would be to include emotions and video images in detecting hate tweets among various users in twitter targeting various groups or individuals. Another problem that could be addressed is the limitation of twitter API for commercial research where authorization is limited. Currently Twitter allows users to collect approximately 1600 tweets per day and will only provide data that has been uploaded in the last six days. To gain real value from a sentiment analysis it would be required to have massive amounts of data on the product or service which is currently not available without premium accounts or using third parties. Given the legal and moral implications of hate speech it is important that we are able to accurately distinguish between the two. Thus we can say Naïve Bayes' classifier can be used successfully to analyze movie reviews.

2.4 Sarcasm detection using combinational Logic and Naïve Bayes Algorithm.

AUTHORS: K. Rathan, R. Suchithra.

Sarcasm detection on twitter tweets is more complicated as it provides very less detailed results, and developing a dictionary for these kind of text documents takes more time and resources. Social media posts are hard to analyze on the phrase or sentence level because of their unique structure and grammar. Since twitter allows user to enter 140 characters processing time also increases. The sarcasm detection was ignored for different languages (except English), repeated tweets and empty or a single letter/word tweets. Finally by using different types of features and their combinational logic we were able to detect sarcasm in twitter training data set. Preprocessing being the very important part of our project it was successfully completed. And the results were clean preprocessed and tagged tweets. In this phase we were able to remove anomalies or the noises such as hyperlinks, emails, links and mentions. We got 100% accuracy in detection and deletion of these noises. In POS tagging most of the important words were successfully detected whereas the undetected ones were due to misspelled words or the words which may be missing from dictionary or it may have been prepositions (in, of, as, a, the) which we are not considering overall we got 75% and over detection which gave us a better POS tagging dataset for feature extraction. The algorithm also detected emoticons and renamed them. Finally the result data set is moved to post processing. Out of 300 tweets we considered 100 #sarcastic tweets, 100 non-sarcastic tweets and 100 sarcastic tweets with no # tags. The results were found to be extraordinary. The algorithm was able to classify accurately over this combined dataset. We found 68% sarcastic in one dataset and 71.35% in another dataset. The future work will be focused on backtracking of tweets (analyzed based on user's past replies and comments) and multilingual language support.

3. SYSTEM ANALYSIS

3.1. EXISTING SYSTEMS:

Lexicon algorithm makes use of a sentiment lexicon. A sentiment lexicon is a collection of known and precompiled sentiment terms. Lexicon algorithm computes the polarity of each term in a textual content in order to deduce the sentiment expressed through that textual content. This sentiment can either be negative, neutral or positive. However, lexicon algorithm is insufficient for sarcasm detection as it limits itself to give the polarity of a textual content without being able to specify whether that textual content is a sarcastic one or not.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

- ❖ It is necessary to extend the lexicon algorithm in order to come out with systems that would be proven efficient for sarcasm detection on neutral and positive sentiment textual contents.

3.2. PROPOSED SYSTEM:

In this paper, the lexicon algorithm has been extended in two ways so as to generate two systems that could be more efficient for sarcasm analysis, especially on neutral and positive sentiment textual contents. A. First system The first system 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 non-sarcastic lingual contents. B. 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.

3.2.1. ADVANTAGES OF PROPOSED SYSTEM:

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 non-sarcastic.

- **Naive Bayes Algorithm:**

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. 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. 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 playing golf. Here is a tabular representation of our dataset.

Table 1:

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+)

In TABLE I, every environment detail has a polarity value.

Table 2:

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)

Table 3:

Textual contents	Results of Sentiment prediction	Final Result
name name it's okay g it's his job . (+)	Positivity	Non-sarcastic
if u dont g the truth dont c new batman/superman movie visit gatorworld or wear very gs diabetes crohns . (+)	Positivity	Non-sarcastic
name so they finally figured that out we have some really the rnc . (+)	Positivity	Non-sarcastic
de g off just before the g the warm up goal for the gs just g better . (+)	Positivity	Non-sarcastic
name i have been followed by name finally my e true pumpifyfacts spice . (+)	Positivity	Non-sarcastic
obama honors uva g professor with early career award - workforce mfg . (+)	Negativity	Sarcastic
name the perfect office - kids edition pkeducate kids parents . (+)	Positivity	Non-sarcastic
prehensive sexuality is education & amp; g adolescents their right to it g them their hopes & amp; aspirati " . (+)	Positivity	Non-sarcastic
stemcelltalksto is initiative that we're happy to suppo stem highschoolstudents . (+)	Negativity	Sarcastic
every time k g k if you' g to tweet call that fucker out oh wait is this still a subtweet . (-)	Negativity	Non-sarcastic
this team looks just like it did before we got mack g . (-)	Positivity	Sarcastic
name ribeiro another of my ""favorites"" . (-)	Negativity	Non-sarcastic
name name name name lololol cause obvs i am a necessity . (-)	Positivity	Sarcastic
name why won't african-americans accept carson as their reformer i don't get it why won't they . (-)	Negativity	Non-sarcastic
name blocked . (-)	Negativity	Non-sarcastic
health: keys g hiv among adolescent girls . (-)	Negativity	Non-sarcastic

There are several types of Naïve Bayes models:

- 1. Gaussian Naïve Bayes:** where the predictors take up continuous value and are not discrete.
- 2. Bernoulli Naïve Bayes:** where the parameters of the predictors are Boolean values; 'yes', 'no', '1' or '0'.

3. Multinomial Naïve Bayes: It is the generalization of Bernoulli 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:

1. Convert the data set into a frequency table.
2. Create a likelihood table.
3. Use Naive Bayesian equation to calculate the posterior probability for each class.

3.3. SYSTEM REQUIREMENTS:

➤ HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

➤ SOFTWARE REQUIREMENTS:

- **Operating System:** Windows
- **Coding Language:** Python 3.7

4. SYSTEM STUDY:

➤ FEASIBILITY 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.

Three key considerations involved in the feasibility analysis are:

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

• 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.

• 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.

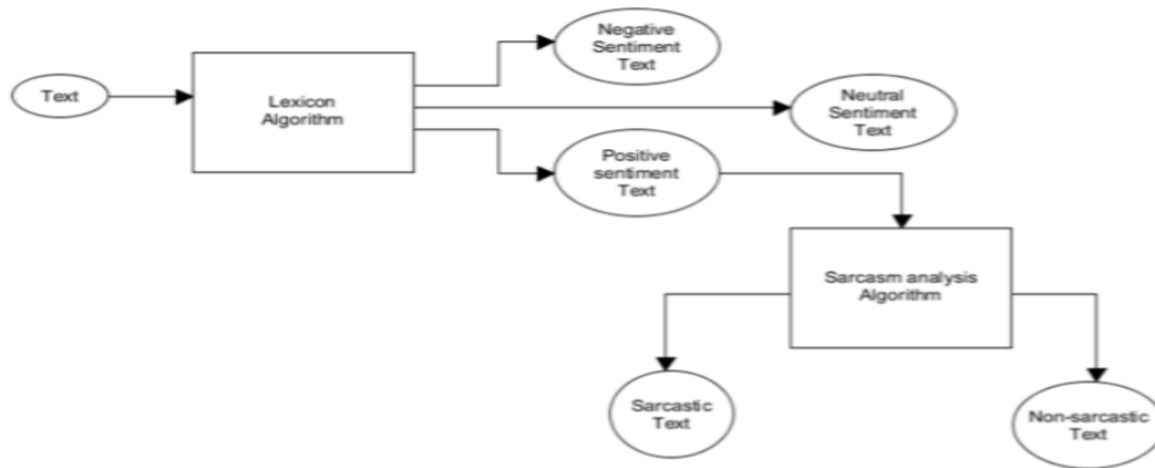
- **SOCIAL FEASIBILITY:**

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.

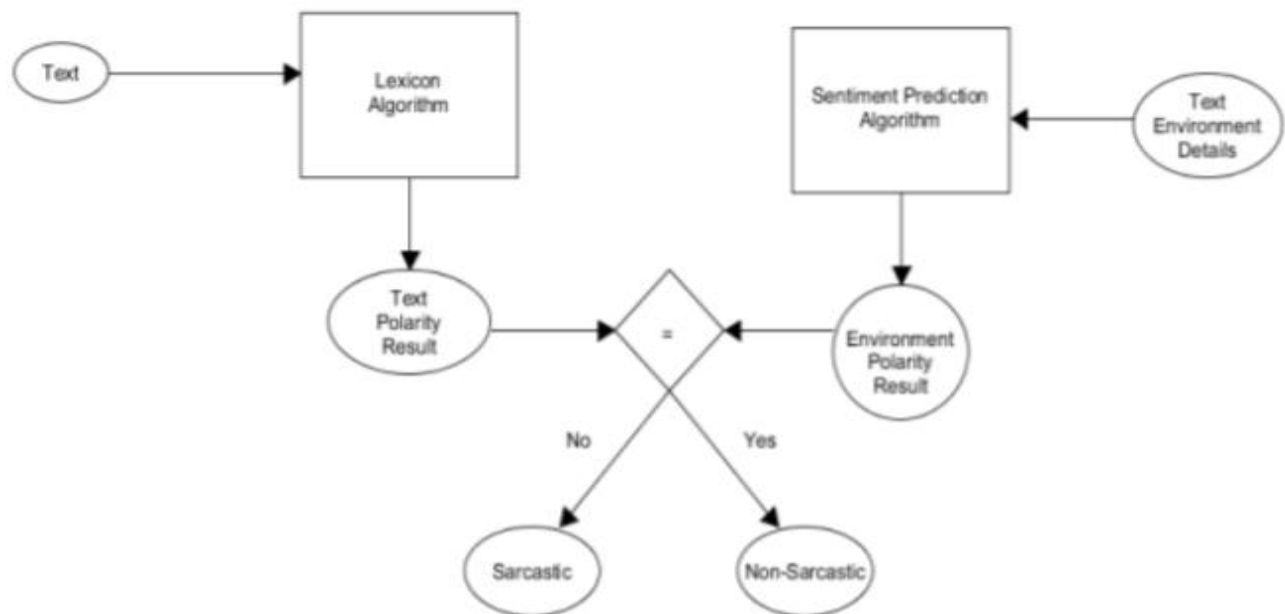
5. SYSTEM DESIGN

5.1. SYSTEM ARCHITECTURE:

First Block:



Second Block:



These environment details are processed based on a training data set that was formed as follow:

- Each environment detail had a polarity. The polarity could be negative, neutral or positive. An AND operation was performed in between the details 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

STRUCTURE OF THE ALGORITHMS

All the algorithms used in this paper have been derived from the classic Naïve Bayes algorithm. Naïve Bayes algorithm was named after Rev. Thomas Bayes. This algorithm makes use of conditional probability to predict the likeness of future occurrence of events based on their historical information.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (1)$$

Where A and B are events and $P(B) \neq 0$ [11].

(P (A|B) is a conditional probability: the likelihood of event A occurring given that B is true. P(B|A) is also a conditional probability: the likelihood of event B occurring given that A is true . P(A) and P(B) are the probabilities of observing A and B independently for each other; this is known as the marginal probability. 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’.

5.2. UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

- The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.
- The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.
- The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.
- The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

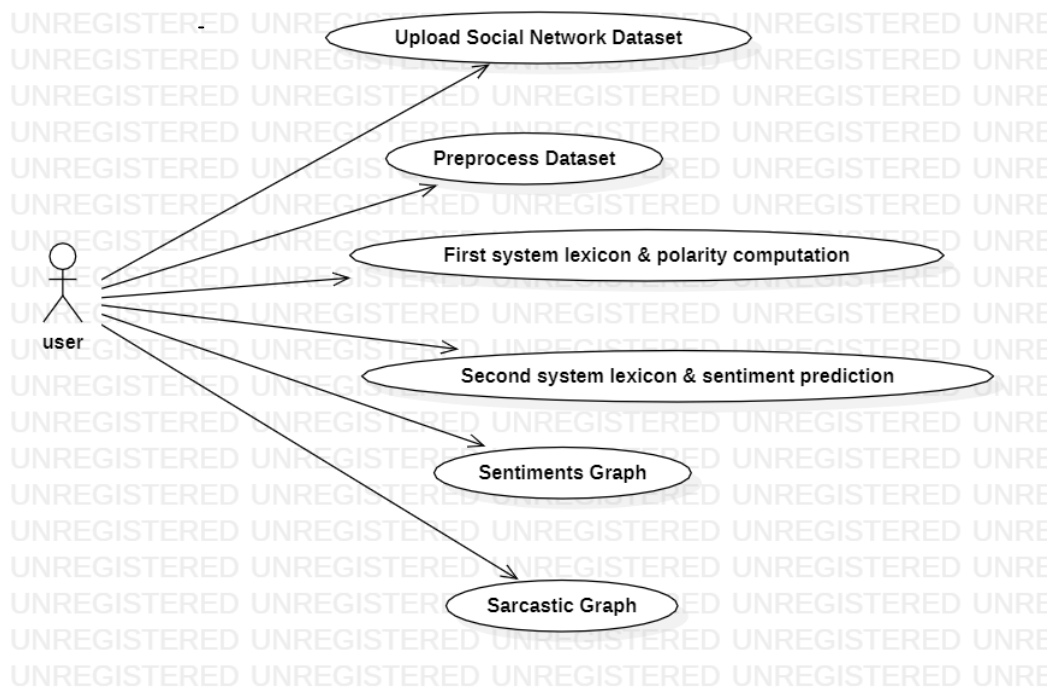
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

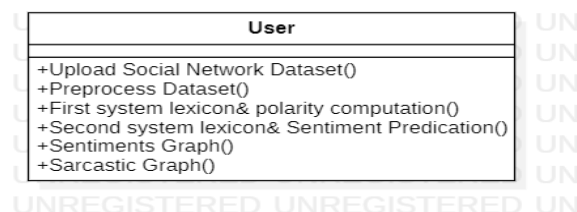
5.2.1. USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



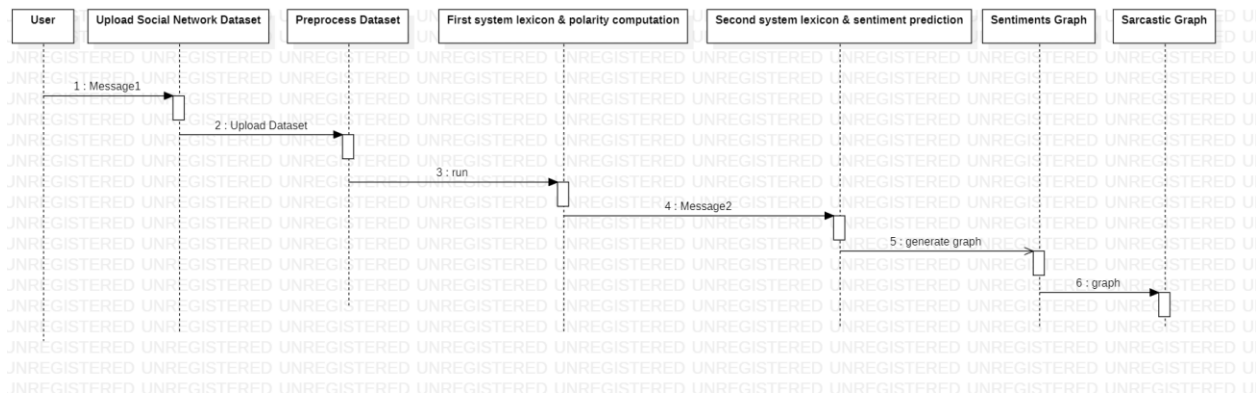
5.2.2. CLASS DIAGRAM:

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



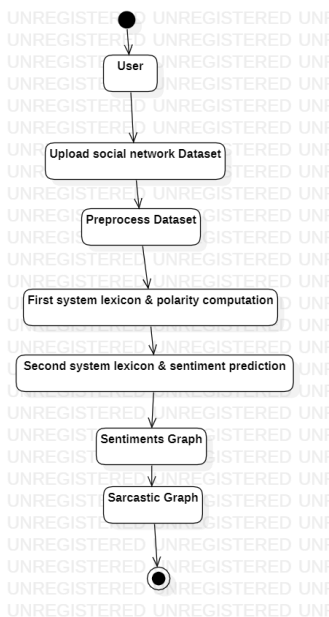
5.2.3. SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



5.2.4. ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



5.3. IMPLEMENTATION:

To implement this project we are using VADER sentiment API from python which built on Naïve Bayes algorithm and using this API we can calculate polarity from sentences. All tweets used in this project for testing saved inside dataset folder.

MODULES DESCRIPTION:

First System: in this module author calculate polarity of sentences such as positive polarity, negative polarity and neutral polarity and then calculate sarcastic by checking positive polarity. If positive and neutral polarity is high and contains some negative words in messages then it will consider as sarcastic otherwise non-sarcastic. Take below example sentence/tweet **‘Mark Zuckerberg used to be a hero of the digital age, but now he has lived long enough to see himself become the villain’**

In above sentence person saying hero to Mark Zuckerberg in one sentence and in other sentence proving him villain. So we can see in above positive sentence user is giving some negativity or using insulting words and such tweets/messages consider as sarcastic.

Second System: In second module we will calculate sentiments from sentences and if sentence is positive or neutral and if positive sentence contains negative words then display/predict sentence as positive with sarcasm else positive without sarcasm.

6. SOFTWARE ENVIRONMENT

What is Python:-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python:-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible:

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable:

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity:

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities:

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy:

When working with Java, you may have to create a class to print '**Hello World**'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable:

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. Object-Oriented:

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source:

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable:

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted:

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

- **Advantages of Python over Other Languages:**

1. Less Coding:

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable:

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for everyone:

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

- **Disadvantages of Python**

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations:

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers:

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions:

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well,

it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers:

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple:

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

- **History of Python: -**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it." Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created

a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

What is Machine Learning: -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories of Machine Learning:-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete

categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machine Learning:-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality – Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment – Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning:-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML –

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis

- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping.

➤ **How to Start Learning Machine Learning?**

Arthur Samuel coined the term “**Machine Learning**” in 1959 and defined it as a “**Field of study that gives computers the capability to learn without being explicitly programmed**”. And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer is the Best Job of 2019 with a 344% growth and an average base salary of **\$146,085** per year. But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!! .This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites:

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus:

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily

focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics:

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python:

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts:

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature** – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** – The idea is to give a set of inputs(features) and its expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning** – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning** – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.

- **Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

➤ **Advantages of Machine learning:-**

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

➤ Disadvantages of Machine Learning:-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

➤ **Python Development Steps: -**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose:-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python:

Python is an interpreted high-level programming language for general-purpose programming.

Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Modules Used in Project:-

1. Tensorflow:

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

2. Numpy:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

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

3. Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. 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.

4. Matplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

5. Scikit – learn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

6. Python:

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. 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 also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

➤ Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace. The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

➤ How to Install Python on Windows and Mac :

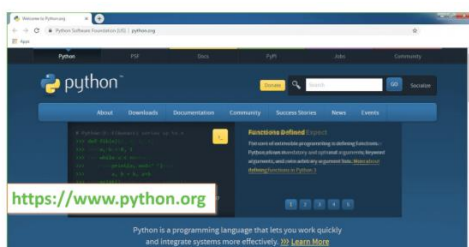
There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

➤ Download the Correct version into the system:

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

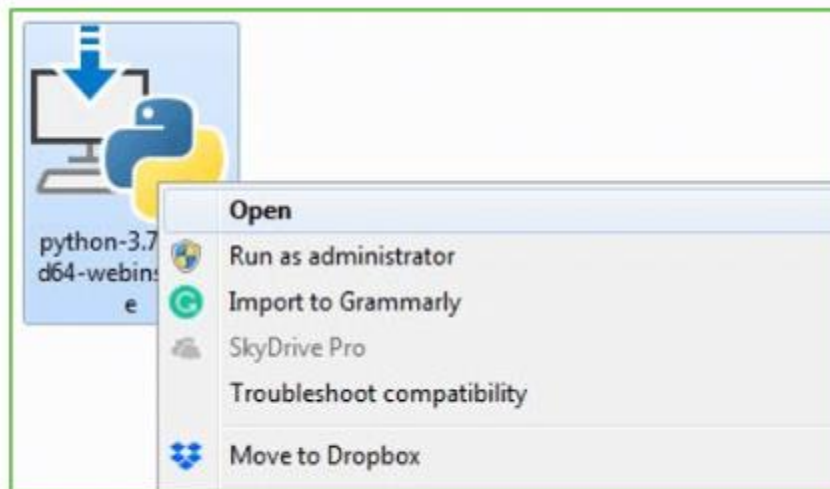
Step 5: Here you see a different version of python along with the operating system.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Source release	Source release		6811673e5b2b4ae77b5b6d1b0f9be	23817963	SG
32 compressed source tarball	Source release		d53e4aa66097051c2eca45ee3604803	17133432	SG
macOS 64-bit installer	Mac OS X	for Mac OS X 10.5 and later	6428b4f7583dbf1a4c2bafce08e6	34898436	SG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5db605c38217a45778b9e4e938d43f	28082845	SG
Windows help file	Windows		483999573a2c58b2ac58cad6b4f7d2	8131761	SG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9800c3c7b8ec080abe8218aa472ba2	7564781	SG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b4a7f6de9db3043a383e363400	26680368	SG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28c31c0088aef72aef653a3b37051b4bd2	1362904	SG
Windows x86 embeddable zip file	Windows		9fa33b61288428795da94123574139d9	6743628	SG
Windows x86 executable installer	Windows		33c802942d5446ac3b8451476394789	25663848	SG
Windows x86 web-based installer	Windows		29670c5a5d1178f82c30983ea371d87c	1324608	SG

To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer. To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer. Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation **Note:** To know the changes or updates that are made in the version you can click on the Release Note Option.

➤ Installation of Python:

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.



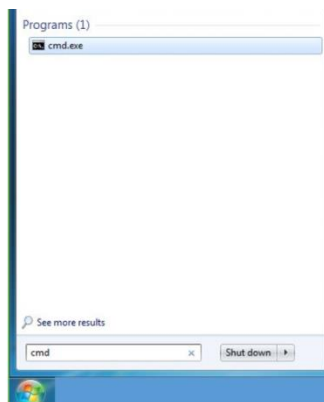
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

➤ **Verify the Python Installation:**

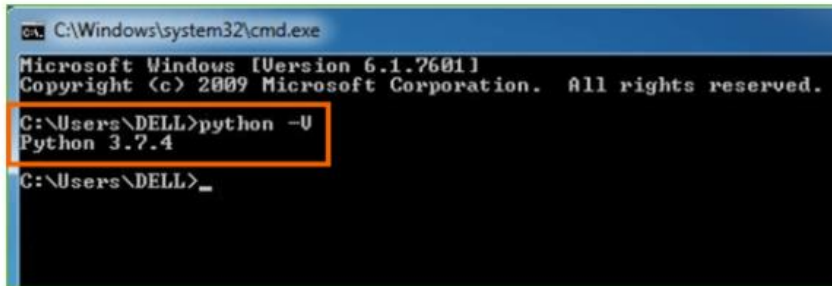
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python -V** and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python -U
Python 3.7.4

C:\Users\DELL>_
```

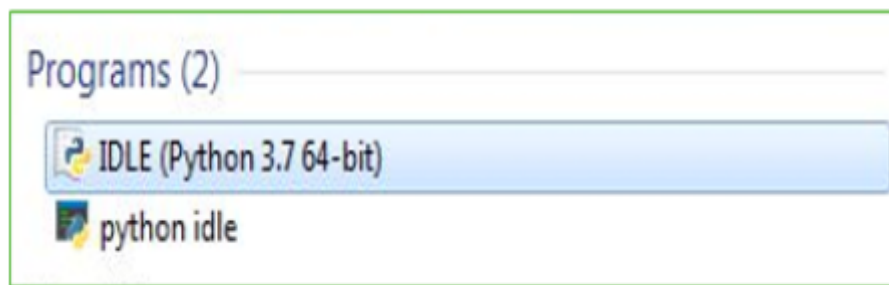
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

➤ Check how the Python IDLE works

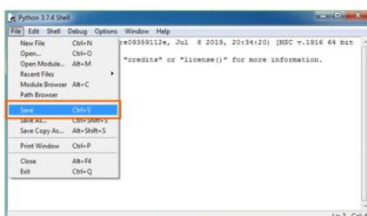
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



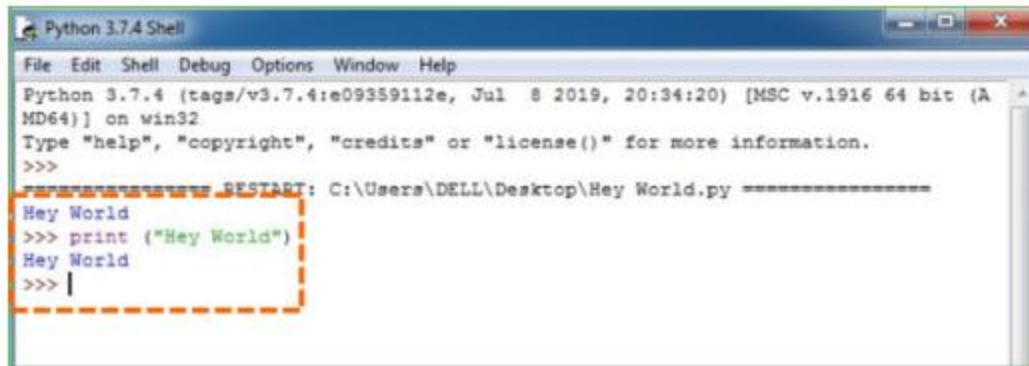
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter **print (“Hey World”)** and Press Enter.



The screenshot shows a 'Python 3.7.4 Shell' window. The title bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays the following content: 'Python 3.7.4 (tags/v3.7.4:09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', and a prompt '>>>'. Below the prompt, the text '***** RESTART: C:\Users\DELL\Desktop\Hey World.py *****' is shown. A dashed orange box highlights the following code and its output: the prompt '>>>', the command 'print ("Hey World")', the prompt '>>>', and the output 'Hey World'. The cursor is positioned at the end of the last prompt '>>>'.

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python doesn't need semicolons at the end of the statements otherwise it won't work.

6.1. CODING:

```
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("EXTENSION OF THE LEXICON
ALGORITHM FOR SARCASM DETECTION")
#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, '=====')
    text.insert(END, '=====')

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

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='EXTENSION OF THE LEXICON ALGORITHM FOR SARCASM DETECTION')
title.config(bg='LightGoldenrod1', fg='mediumorchid')
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 Social
Network 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="Run 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()
```

7. 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

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.

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 satisfactory, as shown by successful 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.

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.

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.

5. White Box Testing:

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

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. 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.

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. 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.

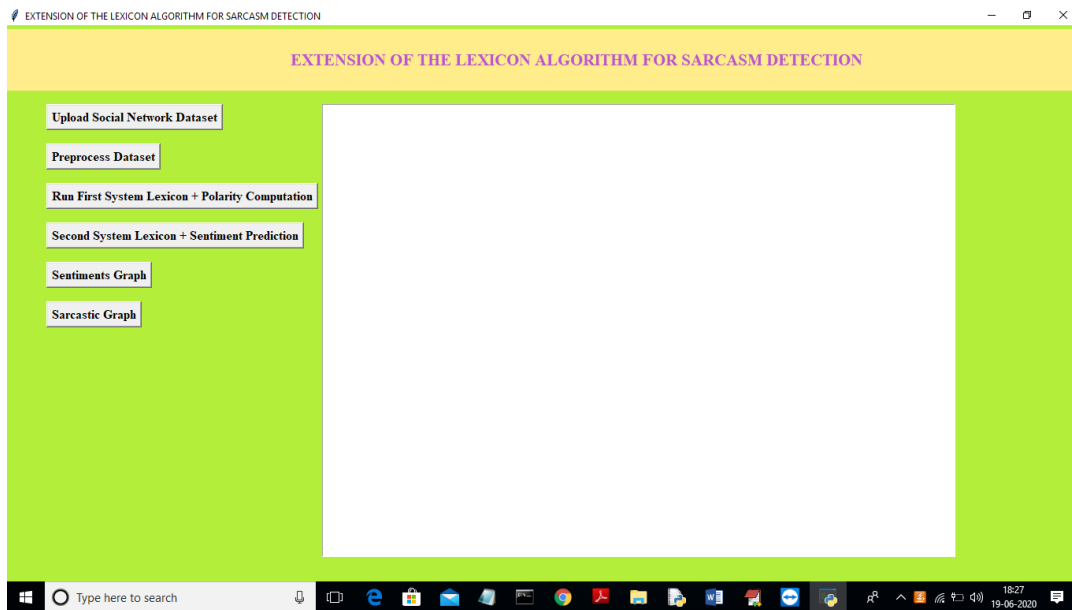
9. Acceptance Testing

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.

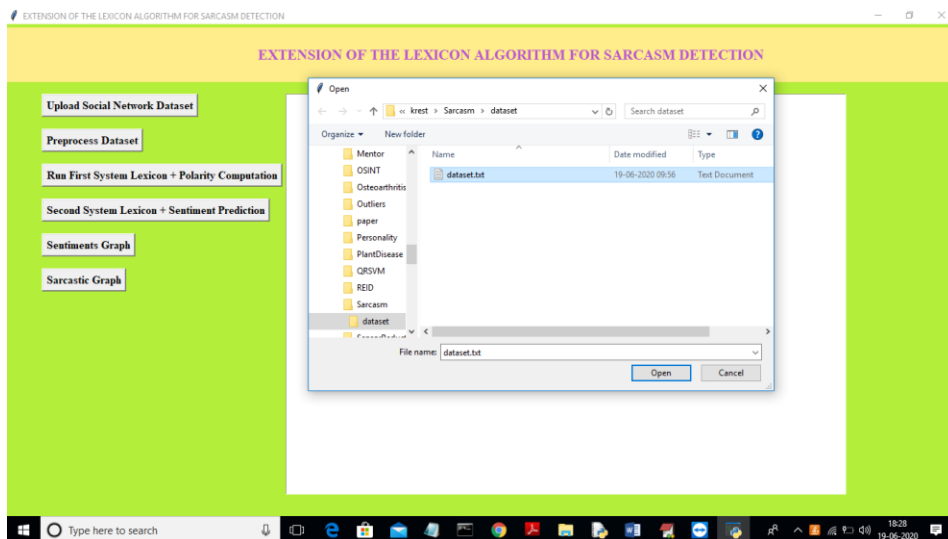
8. RESULT

8.1. OPEN THE FILE:

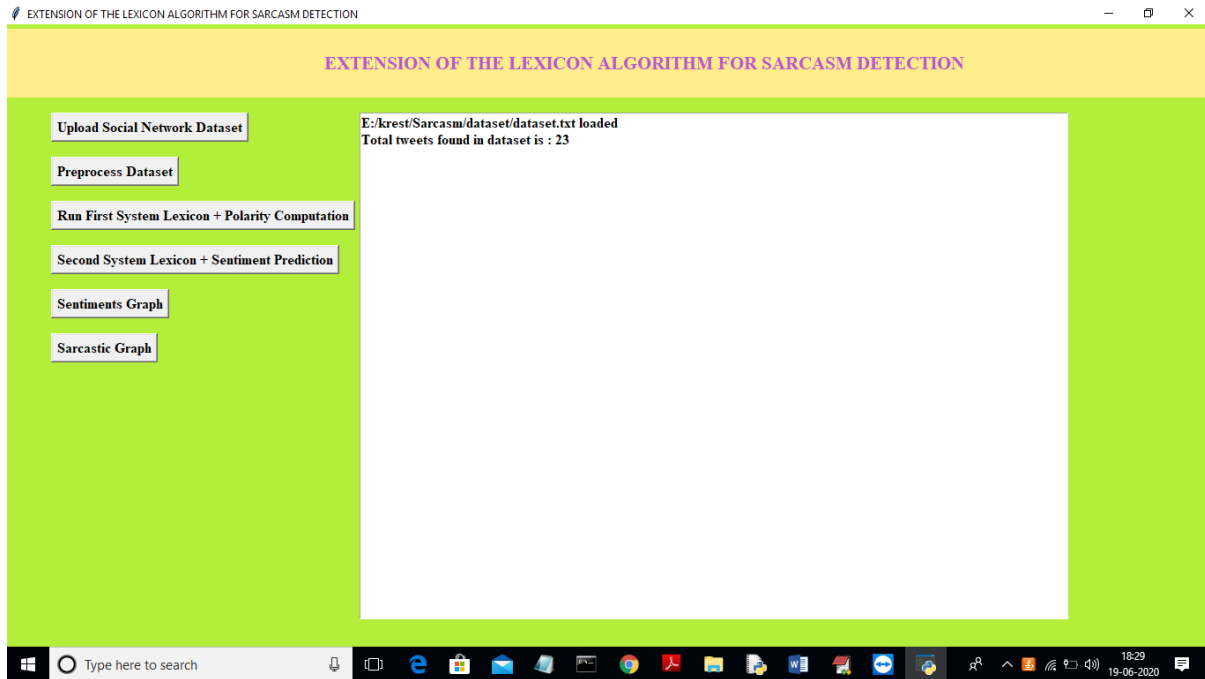


8.2. UPLOAD DATASET:

In above screen click on ‘Upload Social Network Dataset’ button and upload tweets messages.



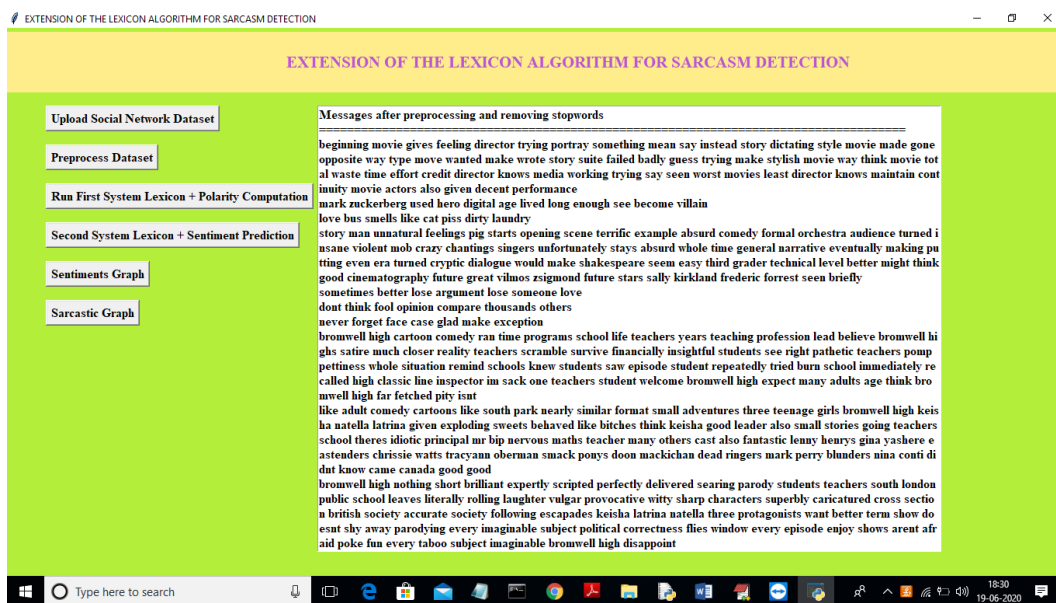
In above screen uploading ‘dataset.txt’ file and now click on ‘Open’ button to load dataset and to get below screen.



In above screen in dataset 23 tweets messages found.

8.3. PREPROCESS DATASET:

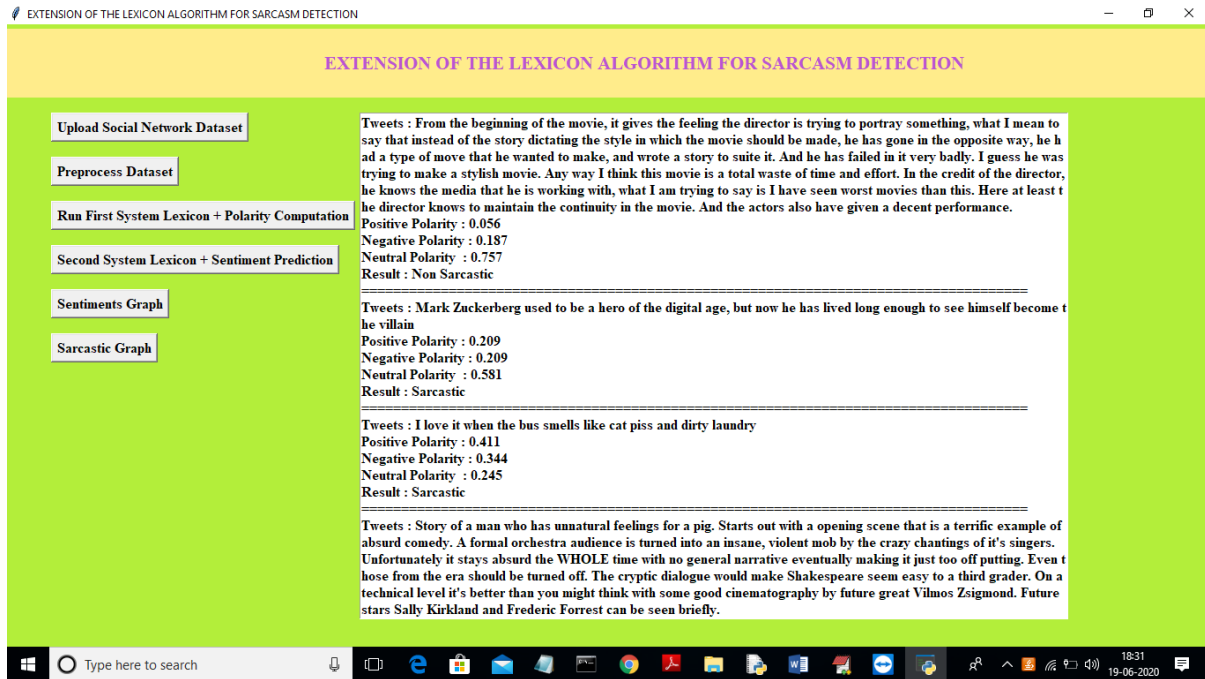
Now click on ‘Preprocess Dataset’ button to remove special symbols and stop words from messages.



In above screen we can see all messages after removing special symbols and stop words.

8.4. RUN FIRST SYSTEM LEXICON + POLARITY COMPUTATION:

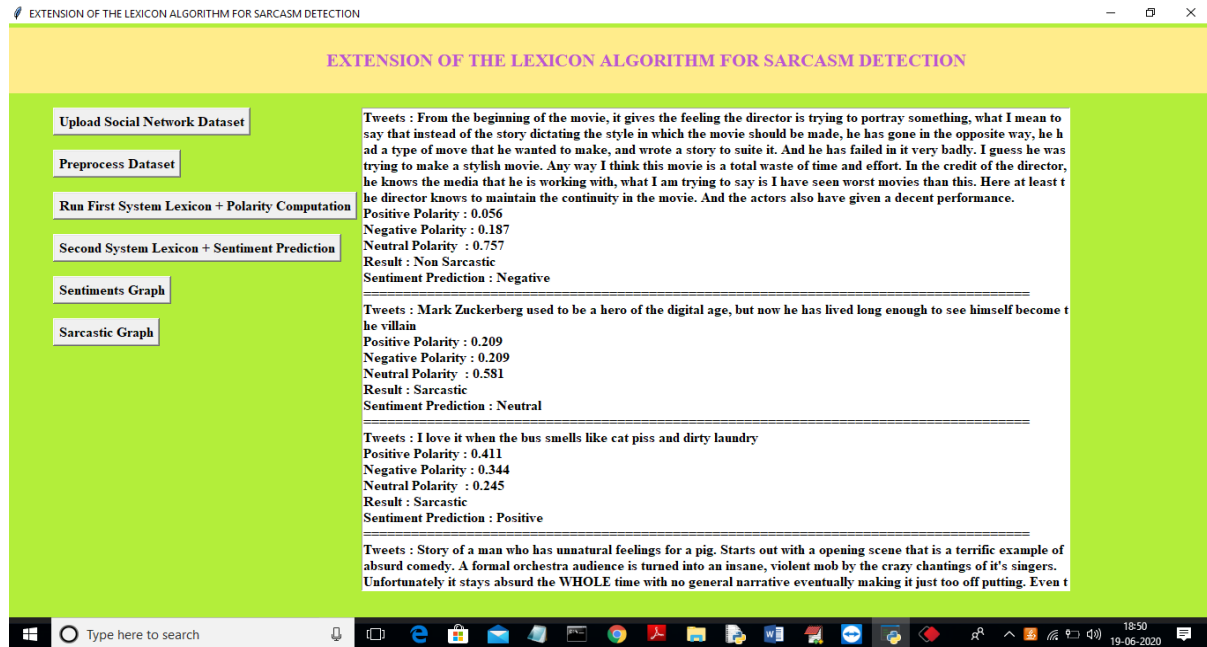
Now click on 'Run First System Lexicon + Polarity Computation' button to calculate polarity of message.



In above screen for each message we can see tweet data and positive, negative and neutral polarity score and the message in tweets is sarcastic or non-sarcastic. The tweets will be classified to positive, negative or neutral based on its high score for example in first tweet neutral got high score as 0.757 so tweet will consider as neutral. If that neutral tweets contains some negative words then consider as sarcastic. You can scroll down above screen text area to see all messages details.

8.5. SECOND SYSTEM LEXICON + SENTIMENT PREDICTION:

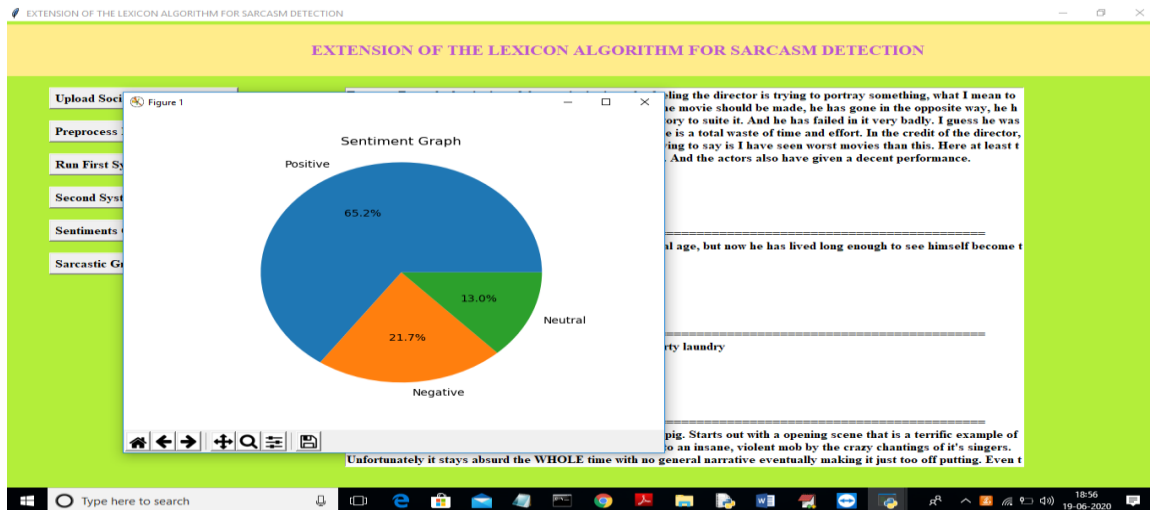
Now click on ‘Second System Lexicon + Sentiment Prediction’ button to predict sentiments of sentences/tweets.



In above screen we can see same results with extra details such as whether tweet/message is positive or negative or neutral. You can scroll down above text area to see all messages.

8.6. SENTIMENT GRAPH:

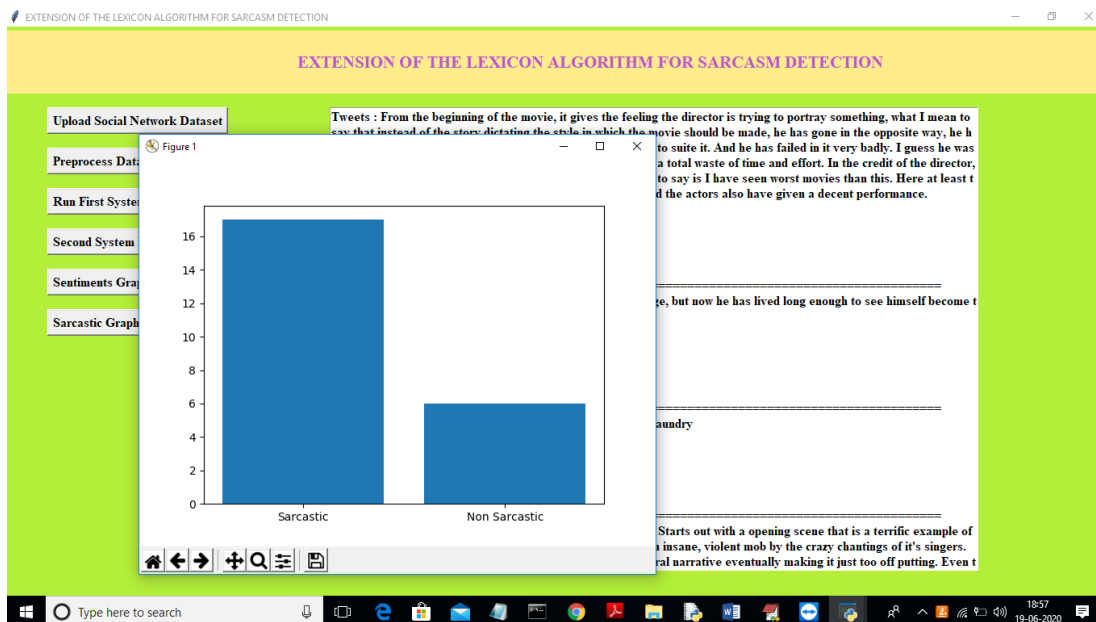
Now click on 'Sentiments Graph' button to get below graph.



In above screen using pie chart we can see percentage of positive, negative or neutral tweets.

8.7. SARCASTIC GRAPH:

Now click on 'Sarcastic Graph' button to get below graph.



In above graph x-axis represents type of tweets and y-axis represents count of sarcastic or non-sarcastic tweets.

9. CONCLUSION

The aim of this study was to propose ways to extend the lexicon algorithm in order to build systems that would be more efficient for sarcasm detection. This aim had been successfully met as two systems 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 in order to obtain meaningful results and to improve the accuracy of the system. The second system constitutes a vast area of study. Some work need to be done in order to develop a system that would allow the collection of environment 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.

10. REFERENCES

- [1]. Cambridge University Press, 2018. sarcasm. [Online] Available at: <https://dictionary.cambridge.org/dictionary/english/sarcasm> [Accessed 20 January 2018].
- [2]. Palanisamy, P., Yadav, V., & Elchuri, H. (2013). Serendio: Simple and Practical lexicon based 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/naivebayes-explained/> [Accessed 10 February 2019].