



TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A Final Year Project Report

On

“Sentiment Analysis and News Classification”

By

KESHAB BAHADUR SUNARI 070BCT515

KIRAN BOHAJU 070BCT516

SARAJU PALUKASI 070BCT539

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND
COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE
REQUIREMENT FOR THE BACHELOR’S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING LALITPUR,
NEPAL

August 2017

LETTER OF APPROVAL

The undersigned hereby certify that they have read and recommended to the Institute of Engineering for acceptance, this project report entitled "**Sentiment Analysis and News Classification**" submitted by **Keshab Bahadur Sunari, Kiran Bohaju, and Saraju Palukasi** in partial fulfillment of the requirement for the Bachelor's Degree in Computer Engineering.

Supervisor

Babu Ram Dawadi

Department of Electronics and Computer

Engineering

Internal Examiner

Bibha Sthapit

Department of Electronics and Computer

Engineering

External Examiner

Krishna Prasad Bhandari

Nepal Telecom Limited

Dr. Dibakar Raj Pant

Head

Department of Electronics and Computer

Engineering

Tribhuvan University, Nepal

Date of Approval:

COPYRIGHT

The authors have agreed that the Library, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purpose may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or the other use of this report for financial gain without approval of to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering and author's written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Department of Electronics and Computer Engineering,
Institute of Engineering, Pulchowk Campus,
Tribhuvan University, Nepal

ACKNOWLEDGEMENTS

We have taken efforts in this project “Sentiment Analysis And News Classification”. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them, even if we forget mentioning some.

We are highly indebted to our dear supervisor Er. Babu Ram Dawadi and the project coordinator Dr. Nanda Bikram Adhikari for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude also towards our colleagues for their aid and encouragement which helped us in completion of this project.

Online resources, many of which we may not have the sources to, have come as a help in understanding ideas and communicating within the team. Deepest of thanks to all those people and organizations who provided this indirect help in our project concept.

With Warm Regards

- Keshab Bahadur Sunari (070/BCT/515)
- Kiran Bohaju (070/BCT/516)
- Saraju Palukasi (070/BCT/539)

ABSTRACT

The advancement of technology and internet service has led to rapid growth of different opinions and reviews on the web. People are always concerned about what others think. So what others think has always been an important aspect of decision making process. Hence, it is necessary to extract opinion- bearing text from the web and analyze what the text conveys. So to achieve this, it is important to know whether the reviews on the web are positive or negative, typically called as sentiment classification.

A very preliminary idea struck our brains which is basically to do something which demonstrates this field of machine learning. We wanted to train a machine to do a particular task after learning it. Considering the time challenges and scope of our major project, we concluded that news classification on the basis of sentiments is a good time biter for starters.

Keywords: Sentiment Analysis, News Scraping, Text Processing, Tokenization, Word Stemming, POS tagging, News Classification, Naïve Bayes, TF-IDF Classifier, Visualization

Table of Contents

LETTER OF APPROVAL	i
ABSTRACT.....	iv
LIST OF FIGURES	viii
LIST OF TABLE	ix
LIST OF SYMBOLS/ABBREVIATION.....	x
1. INTRODUCTION	1
1.1. Background.....	2
1.2. Motivation.....	3
1.3. Problem Statement.....	3
1.4. Proposed Solution	4
1.5. Objective	5
1.6. Scope.....	5
2. LITERATURE REVIEW	7
3. THEORITICAL BACKGROUND	10
3.1. POS Tagging.....	10
3.2. Porter Stemming Algorithm.....	10
3.3. Naïve Bayes Classifier.....	11
3.4. SVM.....	12
3.5. K-NN Classifier	12
3.6. Term Frequency and Inverse Document Frequency	14
4. REQUIREMENT ANALYSIS.....	16
4.1. Software Model	16
4.2. Tools, Technology and Platform	17
4.2.1. Language.....	17
5. METHODOLOGY	20
5.1. Data Collection	20
5.1.1. Scraping RSS Feed.....	20
5.2. Text Processing.....	20
5.2.1. Tokenization	21
5.2.2. Word Stemming	22
5.2.3. Stop Word Cleaning.....	26

5.2.4.	Part Of Speech (POS) Tagging	26
5.3.	Supervised Learning Algorithms.....	27
5.3.1.	Naïve Bayes Algorithm	27
5.3.2.	TF-IDF Classifier	30
5.3.3.	Evaluation Classifier	33
6.	ARCHITECTURE	35
6.1.	News Article Extraction.....	37
6.2.	Data Processing	37
6.3.	Text Classification.....	37
6.4.	Result And Visualization	38
6.5.	Relevant Diagrams	38
6.5.1.	Use Case Diagram	38
6.5.2.	Data Flow Diagram	39
6.5.3.	Sequence Diagram.....	39
7.	DEVELOPMENT IMPLEMENTATION PROCESS	40
7.1.	Web Crawling.....	40
7.2.	Text Processing.....	41
7.2.1.	Tokenization	41
7.2.2.	Filtering Words	42
7.2.3.	Stemming	43
7.2.4.	POS Tagging.....	44
7.2.5.	Replacing Word Matching regular expressions	45
7.3.	Web Interface	47
7.3.1.	Home page	47
7.3.2.	Visualization	49
7.4.	Naïve Bayes.....	52
8.	RESULT	53
8.1.	Performance For Positive News.....	54
8.2.	Performance For Negative News	54
8.3.	Performance For Neutral News	55
8.4.	Overall Performance.....	55
9.	LIMITATION.....	56

10. CONCLUSION	57
11. FUTURE WORK.....	58
12. GANTT CHART.....	59
APPENDIX.....	60
REFERENCES.....	62

LIST OF FIGURES

Figure 4.1 Prototype Model.....	17
Figure 5.1 Text Processing.....	21
Figure 6.1 System Block Diagram.....	36
Figure 6.2 Use Case Diagram.....	38
Figure 6.3 Data Flow Diagram.....	39
Figure 6.4 Sequence Diagram.....	39
Figure 7.1 Scraping title, article.....	41
Figure 7.2 A sample output of Tokenization.....	42
Figure 7.3 A sample output of Filtration Process.....	43
Figure 7.4 A sample output of Word Stemming.....	44
Figure 7.5 A sample output of POS-Tagging	45
Figure 7.6 A sample output of replacing contraction.....	46
Figure 7.7 Home page.....	47
Figure 7.8 Home page with news categories.....	48
Figure 7.9 Home page showing sentiment analysis.....	48
Figure 7.10 Sample of news article display.....	49
Figure 7.11 Visualization using Bar Diagram.....	50
Figure 7.12 Visualization using Pie-Char.....	50
Figure 7.13 Result display by Naïve Bayes Classifier.....	51
Figure 7.14 Result Display by TF-IDF classifier.....	51
Figure 7.15 Sample output of Naïve bayes Classifier.....	52
Figure 12.1 Gantt Chart.....	59

LIST OF TABLE

Table 5.1 Confusion matrix.....34

Appendix 11.1 POS Tagging Labels.....60

LIST OF SYMBOLS/ABBREVIATION

NLP	Natural Language Processing
ML	Machine Learning
NB	Naïve Bayes
POS	Parts-of-Speech
TF-IDF	Term Frequency – Inverse Document Frequency
KNN	K-Nearest Neighbor
TF	Term Frequency
IDF	Inverse Document Frequency
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
AI	Artificial Intelligence
PHP	Personal Home Page
GUI	Graphical User Interface
CSS	Cascading Style Sheets
SVM	Support Vector Machines

1. INTRODUCTION

Sentiment analysis refers to the use of Natural language processing, text analysis, computational linguistics to identify and extraction of subjective information in source material. It is also known as opinion mining. The applications of sentiment analysis are to review and social media for a variety of application, ranging from marketing to customer service.

Newspaper and blogs express opinion of news entities (people, places, things) while reporting on recent events. News can be good or bad, but it is seldom neutral. Although full comprehension of natural language text remains well beyond the power of machines, the statistical analysis of relatively simple sentiment provides important entities.

With the arrival of internet, there has been a radical change in the social life, routine and decisions of common people. Today it's everyday activity and regular practice for each person to read news online and watch advertisements regarding a movie, a product or a book before actually placing money into it. As it has changed their lifestyle, it also has impact on the social life of an individual. The exposure to new online social media such as articles, blogs, message boards, news channels such as Web content is influencing their social life and the way people look at various things around them. People's perspective tends to undergo a change as per the content they read.

The social media has now occupied the major space on the Web. The new user-centric Web hosts a huge amount of data every day. Users are not only consuming the web, but they are also a part of web and co-creators of content on web. The user is now contributing to social media ranging from articles, blog posts, news, tweets, reviews, photo/video upload, etc. This is creating a large amount of the unstructured data on the Web.

The majority of the content that we read today is on the negative aspects of various things e.g. corruption, rapes, thefts etc. Reading such news is spreading negativity amongst the people. Positive news has been dominated and getting less attention. The positivity surrounding the good news has been drastically reduced by the number of bad news.

1.1. Background

The need of research and rising requisites to develop sentiment analysis as a new technology on the computer world can so easily be felt. It is so because so many general problems like telling how happy a face is, or what someone means by a sentence are still only possible for people to sort out.

Sentiment Analysis is the process of determining whether a piece of a given document carries positive, negative or neutral sentiments. It's also known as opinion mining, deriving the opinion or attitude of a speaker. A common use case for this technology is to discover how people feel about a particular topic.

This is gaining popularity mainly due to a great rise in social media, which are providing people a new platform on which to express their ideas and opinions. Moreover, it is now relatively easy for the other side to listen to their clients and customers. But, how can we make it easier? If we were to imagine a system replacing a bunch of people doing the hard work of reading each comment and post to get a statistic of their critics, they could be doing something really useful information about customer's feedback is so vital to a company. It is a basis on which a company makes crucial decisions like what to do next, what to change. Basically, what people want! With a system like this one, that information could be flying in at pace.

The implementation of Sentiment Analysis makes use of a variety of fields in Computer Science such as Natural Language Processing, Text Analysis and Computational Linguistics, working on different levels: document, sentence, or entity.

1.2. Motivation

It is fair to say that people are generally very interested in what other people think about global entities. Also, public opinion about entities help people forms their own opinions. It also helps people make their decisions. Today's world is huge and there is a large amount of data flowing in. New data is being created every day and the ways to handle them are somewhat lagging. Huge amount of data in the form of text is created everyday by people. Looking at the rate this data is growing, we thought we were growing slow. This became our motivation that led us to plan doing something elementary that supported the idea in mind. So we came up with the idea of sentiment analysis of the text, that categorizes the user opinions toward an organization as positively criticized, negatively criticized or neutral.

1.3. Problem Statement

Motivated by different real-world applications, researchers have considered a wide range of problems over a variety of different types of corpora. We now examine the key concepts involved in these problems.

Given an opinionated piece of text, wherein it is assumed that the overall opinion in it is about one single issue or item, classify the opinion as falling under one of two opposing sentiment polarities, or locate its position on the continuum between these two polarities. Much work on sentiment polarity classification has been conducted in the context of reviews (e.g., “thumbs up” or “thumbs down” for movie reviews).

The input to a sentiment classifier is not necessarily always strictly opinionated. Classifying a news article into good or bad news has been considered a sentiment classification task in the literature. But a piece of news can be good or bad news without being subjective (i.e., without being expressive of the private states of the author): for instance, “the stock price rose” is objective information that is generally considered to be good news in appropriate contexts. It is not our main intent to provide a clean-cut definition for what should be considered “sentiment polarity classification” problems, but it is perhaps useful to point out that (a) in determining the sentiment polarity of opinionated texts where the authors do explicitly express their sentiment through statements like “this laptop is great”, objective information such as “long battery life” is often used to help determine the overall sentiment; (b) the task of determining whether a piece of objective information is good or bad is still not quite the same as classifying it into one of several topic-based classes, and hence inherits the challenges involved in sentiment analysis.

1.4. Proposed Solution

Propose solutions is built in python 3 using NLTK module and take input from the user and display output. It is based on linguistic approach. It takes the string, tokenized it and matches the tokens with the datasets in database with the tags added with tokens. Finally it evaluates the score for each polar words and calculates the score for the given text.

The file contains code is divided into classes:

- Splitter class: The given input is in string format. The whole paragraph cannot be evaluated as it. First this class splits the text into tokens/words using tokenize function of NLTK module.

- Pos tagger class: When splitting is done, this class adds tags to the each tokens so that these tokens can be classified as verbs or nouns or adverbs or adjectives, etc. This class does the tagging work and returns the tagged sentences.
- Dictionary tagger class: This class uses the datasets available with the project to make a dictionary for all the tokens tokenized by splitter class and make a dictionary of tagged tokens.

1.5. Objective

The objective of this project is to provide a platform for serving good news and create a positive environment. The new challenging task here is to analyze large volume of unstructured text to be more specific news articles and devise suitable algorithms to understand the opinion of others and find positive and negative aspect of it. This would enable us to focus only on the good news which will help spread positivity around and would allow people to think positively.

- (i) To classify the news articles as positive, negative or neutral.
- (ii) To determine the trend of news happening around the world.
- (iii) To visualize the classified news i.e. positive, negative or neutral in the form of pi-charts and bar-graph.
- (iv) To create a platform to provide positive news happening around.

1.6. Scope

The most important scope of this project will be to provide a platform that filter outs negative articles and serve only good news after classifying news article using sentiment analysis.

Simply, news articles can be classified into respective positive, negative or neutral class using supervised learning classifier such as Naïve bayes, SVM, KNN or decision tree. Furthermore, only positive news can be served to news reader .This will definitely help to start a day with positive vibes.

Sentiment analysis is already evolving rapidly from a very simple (positive, negative, neutral) to more granular and deep understanding. Confidence scores, as well as emotion are important facts in social media monitoring to track customer reviews, survey responses, competitors. These news classifier evaluates the content of human expression in meaningful ways. This requires machine learning approaches that are superseding more traditional rules based approaches.

The applications for sentiment analysis are endless. Thousands of text documents can be processed for sentiment (and other features including named entities, topics, themes, etc.) in seconds, compared to the hours it would take a team of people to manually complete. There is too much potential in machine learning, overtaking some of the manual labor and time of some lexicon based tasks that are labor intensive. This way, huge amount of useful information can be generated in a little amount of time and with minimal effort.

2. LITERATURE REVIEW

Opinion mining as a task can be approached through a machine learning classification approach or by using a lexicon-based method. Dave et al. in their work reported in 2003, used scoring, smoothing, Naive Bayes (NB), Maximum Entropy (ME) and Support Vector Machine (SVM) for assigning sentiment to documents. Here, for scalar ratings, the intuitive approach was to give each word a weight equal to the average of the scores of the documents it appears in. Then the average word score was found in a test document in order to predict its classification. Also, the slope of best fit line along its distribution was used to assign each word a score which yielded the same result. Naive Bayes classification system, with separate probabilities maintained for each of the 5 scoring levels, actually worked even better.

Kim and Hovy in a work reported in 2004, used a probabilistic method for assigning sentiment to expressions. The system contains a module for determining word sentiment and another for combining sentiments within a sentence. The system's best model performed at 81% accuracy with the manually provided holder and at 67% accuracy with automatic holder detection. In their previous research, they had built a sentence subjectivity classifier. In most cases it classifies neutral and weak sentiment sentences as non-opinion bearing sentences.

Goldberg and Zhu [8] build on Pang and Lee's work by incorporating unlabeled data, using a graph-based semi-supervised learning algorithm. Their transductive learning setting showed performance improvements over the work of Pang and Lee, but only when the amount of labelled data was small. Mullen and Collier use a support vector machine to combine feature sets from several previous efforts to good success on both Pang and Lee's Epinions.com data set and a new data set of music reviews.

Bikel et al. in their work in 2007, implemented subsequence kernel based voted perceptron and compared its performance with standard SVM. The subsequence kernel provides the richness of “gappy n-gram” features that appear partially to obviate the need for knowledge rich methods for modeling sentiment in individual sentences. The voted perceptron provides a convenient locally-weighted learning model that can learn a function from word sequences to the real numbers that correlates with the metric of the input labels—even when only trained on 9 the extreme-labeled examples. Even though the model has achieved classification accuracy up to 89% on this task, they state that much investigation still remains. A baseline comparison using a conventional support vector machine classifier was performed. The classifier was unable to distinguish reviews with accuracies better than 61% under any conditions.

Durant and Smith, tried sentiment analysis of political weblogs. Since then many researchers have been trying to use some machine learning classifier for sentiment analysis of unstructured texts. All these classifiers are supervised in nature and require annotated data for training. Lexicon-based methods are unsupervised, in nature used for opinion mining and do not require any training or prior annotated data. Instead this approach relies on use of sentiment lexicons. However, initially, these methods were not very accurate primarily due to inappropriate and limited sentiment lexicons being used. This is why, Turney, used the World Wide Web itself as a corpus and implemented the unsupervised SO-PMI-IR algorithm on movie and travel reviews. The sentiment lexicon used in this approach are either created manually or automatically. The research in lexicon-based approach focused on using adjectives as pointers of the semantic orientation. Turney tested a model on 412 reviews from the Epinions.com website. Parts of-speech tagging was performed on each review, then used a point- wise mutual information/information retrieval (PMIIR) model to each phrase to determine its “semantic orientation” (positive or negative), then took the average semantic orientation of all the phrases in the review.

Taboada et al. also built a dictionary for lexicon based methods with their corresponding semantic orientation (SO). To extract SO for each individual word, a common method based on

point wise mutual information was used. Mutual information between a set of seed words and the target words was calculated using two different methods: a NEAR search on the search engine AltaVista (since discontinued); an AND search on Google. These two dictionaries were tested against a manually annotated dictionary of positive and negative words. The results show that all three methods are quite close, and none performs particularly well. The overall accuracy 10 using the larger dictionary was found to be 56.75% for calculated SO using AND on Google.

With the extension of WordNet as a sentiment lexicon, Sebastiani and Esuli worked towards gloss analysis. The method is based on the quantitative analysis of the glosses of such terms, i.e. the definitions that these terms are given in on-line dictionaries, and on the use of the resulting term representations for semi-supervised term classification. This method outperforms all known methods when tested on the recognized standard benchmarks for this task.

An Adverb-Adjective Combinations (AAC)-based sentiment analysis technique was proposed by Farah Benamara that uses a linguistic analysis of adverbs of degree. A set of general axioms (based on a classification of adverbs of degree into five categories) was defined that all adverb scoring techniques must satisfy. Instead of aggregating scores of both adverbs and adjectives using simple scoring functions, they proposed an axiomatic treatment of AACs based on the linguistic classification of adverbs. They concluded that adjectives are more important than adverbs in terms of how a human being views sentiment - this is because Adjective Priority Scoring (APS) beats Adverb First Scoring in their observations. Second, their result seem to imply that, the “weight” given to adverb scores should be about 35% of the weight given to adjective scores.

3. THEORITICAL BACKGROUND

3.1. POS Tagging

In corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context—i.e. its relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc. Once performed by hand, POS tagging is now done in the context of computational linguistics, using algorithms which associate discrete terms, as well as hidden parts of speech, in accordance with a set of descriptive tags. POS-tagging algorithms fall into two distinctive groups: rule-based and stochastic. E. Brill's tagger, one of the first and most widely used English POS-taggers, employs rule-based algorithms.

3.2. Porter Stemming Algorithm

The Porter stemming algorithm is a process for removing suffixes from words in English. Removing suffixes automatically is an operation which is especially useful in the field of information retrieval. In a typical IR environment a document is represented by a vector of words, or terms. Terms with a common stem will usually have similar meanings. For example:

CONNECT

CONNECTED

CONNECTING

CONNECTION

CONNECTIONS

Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes –ED, -ING, -ION, IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous.

Porter algorithm was made in the assumption that we don't have a stem dictionary and that the purpose of the task is to improve IR performance (not as a linguistic exercise). The program is given an explicit list of suffixes, and, with each suffix, the criterion under which it may be removed from a word to leave a valid stem.

3.3. Naïve Bayes Classifier

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. It 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.

3.4. SVM

It is a very popular classification algorithm which classifies the data into two classes. It will also be implemented in the analysis part of the proposed system. It uses supervised learning to classify the input data. As the algorithm is trained, it learns a hyperplane which can then classify the test data into either of the two classes.

3.5. K-NN Classifier

According to Leif a non-parametric method of pattern classification popularly known as K-Nearest Neighbor rule was believed to have been first introduced by Fix and Hodges in 1951, in an unpublished US Air Force School of Aviation Medicine report. The method however, did not gain popularity until the 1960's with the availability of more computing power, since then it has become widely used in pattern recognition and classification. K-Nearest Neighbor could be described as learning by analogy, it learns by comparing a specific test tuple with a set of training tuples that are similar to it. It classifies based on the class of their closest neighbors, most often, more than one neighbor is taken into consideration hence, the name K-Nearest Neighbor (K-NN), the "K" indicates the number of neighbors taken into account in determining the class. The K-Nearest-Neighbor (KNN) classification method has been trained to be used on-line and in real-time to identify clients/visitors click stream data, matching it to a particular user group and recommend a tailored browsing option that meet the need of the specific user at a particular time.

K-Nearest Neighbor classifier for pattern recognition and classification in which a specific test tuple is compared with a set of training tuples that are similar to it. The K-Nearest Neighbor (K-NN) algorithm is one of the simplest methods for solving classification problems; it often yields competitive results and has significant advantages over several other data mining methods.

- (i) Providing a faster and more accurate recommendation to the client with desirable qualities as a result of straightforward application of similarity or distance for the purpose of classification.
- (ii) Our recommendation engine collects the active users' click stream data, match it to a particular user's group in order to generate a set of recommendation to the client at a faster rate.

The K-Nearest Neighbor classifier usually applies the Euclidean distance between the training tuples and the test tuple.

$$d(x_i, x_t) = \sqrt{(x_{i1} - x_{t1})^2 + (x_{i2} - x_{t2})^2 + \dots + (x_{ip} - x_{tp})^2} \quad \text{----- equation (1)}$$

In general term, the Euclidean distance between two Tuples for instance

$X1 = (x11, x12, \dots, x1n)$ and $X2 = (x21, x22, \dots, x2n)$ will be:

$$\text{dist}(x_1, x_2) = \sqrt{\sum_{i=1}^n (x_{1i} - x_{2i})^2} \quad \text{----- equation (2)}$$

3.6. Term Frequency and Inverse Document Frequency

Tf-idf stands for term frequency-inverse document frequency and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- (i) **TF: Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

$$TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document}).$$

- (ii) **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that

certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:

$$\text{IDF}(t) = \log_e(\text{Total number of documents} / \text{Number of documents with term } t \text{ in it}).$$

4. REQUIREMENT ANALYSIS

Every project is constrained with some requirement specifications. Requirement specification is the view that provides picture of functional and nonfunctional requirement of the system. The Software Requirement Specification finds the basis to describe the aim and motives for the software development. The project is constrained by the following requirements:

4.1. Software Model

The software development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out. Choosing right model for developing of the software product or application is very important. We are supposed to do our project using prototype model as software development methodology.

The basic idea is a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. The prototype is usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

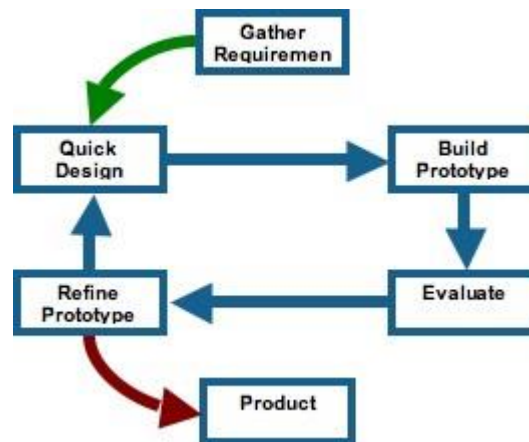


Fig. 4.1: Prototype Model

4.2. Tools, Technology and Platform

4.2.1. Language

Python was used as the server side scripting language for our project and developed an entire application for sentiment analysis. For Client side scripting, HTML, CSS, JavaScript were used.

Framework:

Python Django framework was used for coding our web application. Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so we can focus on writing our app without needing to reinvent the wheel.

Django has been used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc.).

Natural Language Toolkit (NLTK)

NLTK, a comprehensive Python library was used for natural language processing and text analytics. The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries. NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. It is often used in rapid prototyping of text processing program.

Scrapy

Scrapy is a free and open source web crawling framework, written in Python. Originally designed for web scraping, it can also be used to extract data using APIs or as a general purpose web crawler. It is currently maintained by Scraping hub Ltd., a web scraping development and Services Company.

Scrapy project architecture is built around ‘spiders’, which are self-contained crawlers which are given a set of instructions. Following the spirit of other don’t repeat yourself frameworks, such as Django, it makes it easier to build and scale large crawling projects by allowing developers to re-use their code. Scrapy also provides a web crawling shell which can be used by developers to test their assumptions on a site’s behavior.

5. METHODOLOGY

5.1. Data Collection

The data has been collected mostly from these three sources:

- (i) Manually entry by team member

5.1.1. Scraping RSS Feed

Scraping RSS feed refer to the data extraction from the RSS feed which include title, description and the link of the news article which we want to scrape. For this we have used Scrapy which is a python framework large scale web scraping. The extracted url is used to extract the details of articles. The data can be extract from the following:

- (i) **RSS feed BBC:** - Extraction of article's URL for the detailed extraction of the article.
- (ii) **BBC web article:** - Article title, description, article publication Date and the article is extracted from the URL scraped in the RSS feed.

5.2. Text Processing

Natural language processing (NLP) deals with the application of computational models to text or speech data. Application areas within NLP include automatic (machine) translation between languages; dialogue systems, which allow a human to interact with a machine using natural language; and information extraction, where the goal is to transform unstructured text into structured representations that can be searched and browsed in flexible ways. NLP technologies are having a dramatic impact on the way people interact with computers, on the way people

interact with each other through the use of language, and on the way people access the vast amount of linguistic data now in electronic form. From a scientific viewpoint, NLP involves fundamental questions of how to structure formal models (for example statistical models) of natural language phenomena, and of how to design algorithms that implement these models. In this process we will study mathematical and computational models of language, and the application of these models to key problems in natural language processing. The course has a focus on machine learning methods, which are widely used in modern NLP systems: we will cover formalisms such as hidden Markov models, probabilistic context-free grammars, log-linear models, and statistical models for machine translation.

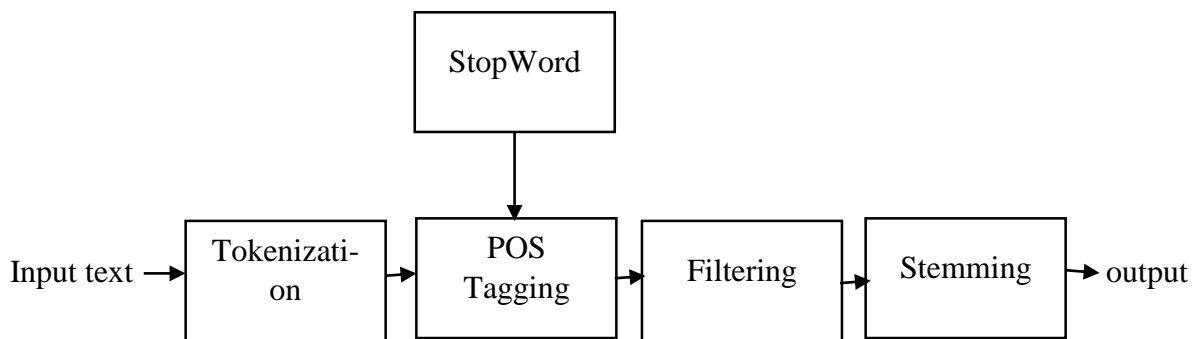


Fig 5.1 : Text Processing

5.2.1. Tokenization

Tokenization is the name given to the process of chopping up sentences into smaller pieces (words or tokens). The segmentation into tokens can be done with decision trees, which contains information to correctly solve the issues you might encounter. Some of these issues you would have to consider are:

- (i) The choice for the delimiter will for most cases be a whitespace (“We’re going to Barcelona” --- [“We’re”, “going”, “to”, “Barcelona.”])), but what should you do when

you come across words with a white space in them (“We’re going to The Hague.” --- [“We’re”, “going”, “to”, “The”, “Hague”]).

- (ii) What should you do with punctuation marks? Although many tokenizers are geared towards throwing punctuation away, for Sentiment analysis a lot of valuable information could be deduced from them. ! puts extra emphasis on the negative/positive sentiment of the sentence, while ? can mean uncertainty (no sentiment).
- (iii) “, ‘ , [], () can mean that the words belong together and should be treated as a separate sentence. Same goes for words which are bold, italic, underlined, or inside a link. If you also want to take these last elements into consideration, you should scrape the html code and not just the text.

5.2.2. Word Stemming

Stemming is the term used in linguistic morphology and information retrieval to describe the process for reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation. A stemmer for English, for example, should identify the string “cats” as based on the root “cat”, and “stems”, “stemmer”, “stemming”, “stemmed” as based on “stem”. A stemming algorithm reduces the words “fishing”, “fished”, and “fisher” to the root word, “fish”. On the other hand, “argue”, “argued”, “argues”, “arguing”, and “argus” reduce to the stem “argu” (illustrating the case where the stem is not itself a word or root) but “argument” and “arguments” reduce to the stem “argument”.

Algorithms for stemming (The Porter Stemming Algorithm).

The porter stemmer algorithm was chosen because it had a very simple core concept and even if exceptional words were found, it offered a very easy bypass around them. General assumptions for the algorithm:

- (i) Define a vowel as any of the letters: a, e, i, o, u
- (ii) Define a double as of the following: bb dd ff gg mm nn pp rr tt
- (iii) Define a valid li-ending as one of: c d e g h k m n r t
- (iv) Define a short syllable in a word as either
 - a. a vowel followed by a non-vowel other than w, x or y and preceded
 - b. a vowel at the beginning of the word followed by a non-vowel.

So rap, trap, entrap end with a short syllable, and ow, on, at are classed as short syllables. But uproot, bestow, disturb do not end with a short syllable. A word is called short if it ends in a short syllable if the word has two letters or less, leave it as it is.

Algorithm steps:

- (i) Search for the longest among the following suffixes, and perform the action indicated.
 - a. sses: replace by ss
 - b. ied ies: replace by I if preceded by more than one letter, otherwise by i.e. (soties – tie, cries – cri)
 - c. s : delete if the preceding word part contains a vowel not immediately before the s (so gas and this retain the s, gaps and kiwis lose it)

- (ii) Search for the longest among the following suffixes, and perform the action indicated.
- a. eed eedly: replace by ee ed edly ing ingly: delete if the preceding word part contains a vowel, and after the deletion: if the word ends at, bl or iz add e (so luxuriat – luxuriate), or
 - b. if the word ends with a double remove the last letter (so hopp – hop), or
 - c. if the word is short, add e (so hop – hope) replace suffix y by i if preceded by a non vowel which is not the
 - first letter of the word
 - cry – cri
 - by – by
 - say-say
- (iii) Search for the longest among the following suffixes, and perform the action indicated
- a. tional: replace by tion
 - b. enci: replace by ence
 - c. anci: replace by ance
 - d. abli: replace by able
 - e. entli: replace by ent
 - f. izer ization: replace by ize
 - g. ational ation ator: replace by ate
 - h. alism aliti alli: replace by al

- i. fullness: replace by ful
- j. ousli ousness: replace by ous
- k. iveness iviti: replace by ive
- l. biliti bli: replace by ble
- m. ogi: replace by og if preceded by l
- n. fulli: replace by ful
- o. lessli: replace by less
- p. li: delete if preceded by a valid li-ending

(iv) Search for the longest among the following suffixes, perform the action indicated

- a. tional: replace by tion
- b. ational: replace by ate
- c. alize: replace by al
- d. icate iciti ical: replace by ic
- e. ful ness: delete

(v) Search for the longest among the following suffixes, and, if found, perform the action indicated.

- a. al, ance, ence, er, ic, able, ible, ant, ement, ment, ent, ism, ate,
- b. iti, ous, ive, ize

5.2.3. Stop Word Cleaning

Stemming is the term used in linguistic morphology and information retrieval to describe the process for reducing inflected words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called conflation. A stemmer for English, for example, should identify the string “cats” as based on the root “cat”, and “stems”, “stemmer”, “stemming”, “stemmed” as based on “stem”. A stemming algorithm reduces the words “fishing”, “fished”, and “fisher” to the root word, “fish”. On the other hand, “argue”, “argued”, “argues”, “arguing”, and “argus” reduce to the stem “argu” (illustrating the case where the stem is not itself a word or root) but “argument” and “arguments” reduce to the stem “argument”.

5.2.4. Part Of Speech (POS) Tagging

In corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph.

A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc. Once performed by hand, POS tagging is now done in the context of computational linguistics, using algorithms which associate discrete terms with set of predefined tags. POS taggers use several kinds of information such as dictionaries, rules etc. to perform their operation.

A word may ideally have one tag or may have more than one tag for example, the word run may either be a noun or a verb according to its context taggers may use either use rules or probability to break this ambiguity according to the type of algorithm used (rule base vs stochastic). There are mainly two type of taggers: rule-based and stochastic. Rule-based taggers use hand-written rules to distinguish the tag ambiguity. Stochastic taggers are either HMM based, choosing the tag sequence which maximizes the product of word likelihood and tag sequence probability, or cue-based, using decision trees or maximum entropy models to combine probabilistic features.

Ideally a typical tagger should be robust, efficient, accurate, tunable and reusable. In reality taggers either definitely identify the tag for the given word or make the best guess based on the available information. As the natural language is complex it is sometimes difficult for the taggers to make accurate decisions about tags. So occasional errors in tagging is not taken as a major roadblock to research.

5.3. Supervised Learning Algorithms

5.3.1. Naïve Bayes Algorithm

Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem with strong (naive) independence assumptions between the features. The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents. 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. An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.

Abstractly, Naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $x=(x_1,\dots,x_n)$ representing some n features (independent variables), it assigns to this instance probabilities.

$$P(C_k|(x_1,\dots,x_n) \quad \text{----- equation(3)}$$

for each of K possible outcomes C_k

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$P(C_k|x) = P(C_k) P(x|C_k) / P(x) \quad \text{----- equation(4)}$$

$$P(c|x) = \frac{P(x|c).P(c)}{P(x)} \quad \text{----- equation(5)}$$

$$P(c|x) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c) \quad \text{----- equation(6)}$$

where,

$P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).

$P(c)$ is the prior probability of *class*.

$P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.

$P(x)$ is the prior probability of *predictor*.

(i) Train Multinomial Naïve Bayes(C,D)

1. $V \leftarrow \text{ExtractVocabulary}(D)$
2. $N \leftarrow \text{CountDocs}(D)$
3. For each $c \in C$
4. Do $N_c \leftarrow \text{CountDocsInClass}(D, c)$
5. $\text{prior}[c] \leftarrow N_c / N$
6. $\text{text}_c \leftarrow \text{CountenateTextOfAllDocsInClass}(D, c)$
7. for each $t \in V$
8. do $T_{ct} \leftarrow \text{CountTokensOfTerm}(\text{text}_c, t)$
9. for each $t \in V$
10. do $\text{condprob}[t][c] \leftarrow \frac{T_{ct} + 1}{\sum_{t'} T_{ct'} + 1}$
11. return $V, \text{prior}, \text{condprob}$

(ii) Apply Multinomial Naïve Bayes(C, V, prior, condprob, d)

1. $W \leftarrow \text{ExtractTokensFromDocs}(V, d)$
2. for each $c \in C$
3. do $\text{score}[c] \leftarrow \log \text{prior}[c]$
4. for each $t \in W$
5. do $\text{score}[c] += \log \text{cond prob}[t][c]$

6. $\text{return arg max}_{c \in C} \text{score}[c]$

5.3.2. TF-IDF Classifier

In information retrieval, tf-idf, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Nowadays, tf-idf is one of the most popular term-weighting schemes. For instance, 83% of text-based recommender systems in the domain of digital libraries use tf-idf.

Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification.

(i) Term Frequency:

A set of English text documents is available and which document is most relevant to the query "the brown cow" is needed to be determined. A simple way to start out is by eliminating documents that do not contain all three words "the", "brown", and "cow", but this still leaves many documents. To further distinguish them, the number of times each term occurs in each document can be counted; the number of times a term occurs in a document is called its term frequency. However, in the case where the length of documents vary greatly, adjustments are often made.

The first form of term weighting is due to Hans Peter Luhn and is based on the Luhn Assumption: the weight of a term that occurs in a document is simply proportional to the term frequency.

(ii) Inverse Document Frequency:

Because the term "the" is so common, term frequency will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "brown" and "cow". The term "the" is not a good keyword to distinguish relevant and non-relevant documents and terms, unlike the less common words "brown" and "cow". Hence an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely.

Karen Spärck Jones conceived a statistical interpretation of term specificity called Inverse Document Frequency (IDF), which became a cornerstone of term weighting: the specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.

(iii) Definition:

TF-IDF is the product of two statistics, term frequency and inverse document frequency. Various ways for determining the exact values of both statistics exist.

In the case of the term frequency $tf(t,d)$, the simplest choice is to use the raw count of a term in a document, i.e. the number of times that term t occurs in document d . If the raw count is denoted by ft,d , then the simplest tf scheme is $tf(t,d) = ft,d$. Other possibilities include:

Boolean “frequencies”: $tf(t,d) = 1$ if t occurs in d and 0 otherwise;

Term frequency adjusted for document length: $ft,d / (\text{number of words in } d)$

Logarithmically scaled frequency: $tf(t,d) = 1 + \log (ft,d)$, or zero if ft,d is zero

Augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the raw frequency of the most occurring term in the document:

$$tf(t, d) = 0.5 + 0.5 * \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}} \quad \text{----- equation(7)}$$

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$idf(t, D) = \log \left(\frac{N}{|\{d \in D : t \in d\}|} \right) \quad \text{----- equation(8)}$$

where,

N : total number of documents in the corpus $N = |D|$

$|\{d \in D : t \in d\}|$: Number of documents where the term t appears (i.e., $tf(t,d) \neq 0$).

If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D: t \in d\}|$.

Then tf-idf is calculated as

$$\text{tfidf}(t,d,D) = \text{tf}(t,d) * \text{idf}(t,D) \quad \text{----- equation(9)}$$

A high weight in tf-idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and tf-idf) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and tf-idf closer to 0.

The corresponding classifier, a TF-IDF classifier, is the function that assigns a class label $y = C_k$ for some k to a query q , as :

$$y = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} \sum_{t \in q \cap d_k} \text{tf}_{t,d_k} * \text{idf}_{t,d} \quad \text{----- equation(10)}$$

Where,

d_k is the collection of all the paragraphs of class C_k and d is the mega-document containing paragraphs of all the classes C_1, C_2, \dots, C_k .

5.3.3. Evaluation Classifier

(i) Confusing Matrix

In the field of machine learning, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each column of the matrix represents the instances in an actual class while each row represents the instances in a predicted class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another).

Table 5.1 confusing matrix

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table). A confusion matrix is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives.

- a. **False positives (FP):** Number of incorrect predictions that an instance is positive
- b. **False negatives (FN):** Number of incorrect predictions that an instance is negative

- c. **True positives (TP)**: Number of correct predictions that an instance is positive
- d. **True negatives (TN)**: Number of correct predictions that an instance is negative

(ii) **Performance Measures**

- **Accuracy**: The proportion of the total number of predictions that are correct.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad \text{----- equation(11)}$$

- **Recall**: The proportion of positive cases that are correctly identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{----- equation(12)}$$

- **Precision**: The proportion of the predicted positive cases that are correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{----- equation(13)}$$

- **Balanced F1-measure**: The weighted harmonic mean of precision and recall.

$$F = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{----- equation(14)}$$

6. ARCHITECTURE

Analyzing Sentiment of the text is itself a challenging task in Natural Language Processing. There are many approaches found for the solution for the sentiment analysis of news articles. Initially different words which carry sentiments (like positive, negative and neutral) will be collected by Harvard General Inquirer (HGI) and WordNet. Later data for analyzing the structure of the text will be needed. The RSS feeds from various source based on category will be read. After that, words in the headlines will be tagged. The verb, adverbs and adjectives from the headline can be found using POS tagger. After that the feature will be extracted from the text. It is very difficult to process text which contains millions of different unique words. Therefore, feature extraction is one of the approaches used when applying machine learning algorithms like Support Vector Machine (SVM) or naïve bayes for text categorization.

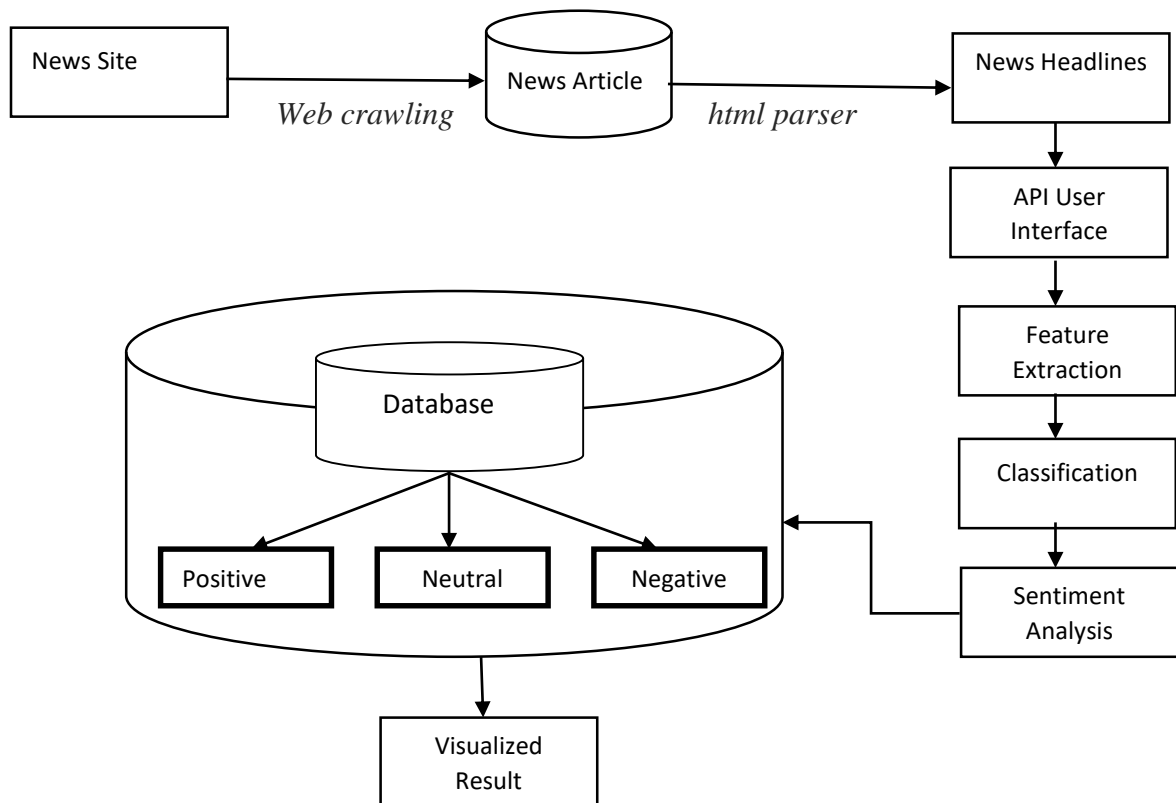


Fig 6.1 : System Block Diagram for Sentiment Analysis and News Classification

6.1. News Article Extraction

The first module comprises of two tasks. In the first task, the news articles are downloaded from a website using a web crawler. These articles are in the HTML format. In the second task, headlines is extracted from HTML article page. This task can be done using the HTML Parser. The HTML parser selects the headlines from HTML documents and creates a temporary text file.

6.2. Data Processing

In the second module data preprocessing steps are performed. The second module is based on the Natural Language Processing (NLP) operations. Once the temporary text file is created, it is subjected to the NLP operations such as Sentence identification, Tokenization, removing punctuations, Parts of speech tagging. These tasks will be done using the WEKA tool. This module gives candidate keywords and combinations of words which will be further useful for determining sentiments of the article.

6.3. Text Classification

In third module text classification task is performed. The candidates keywords generated in previous module are taken as input for this task. This candidate keyword is compared with the words in positive dictionary if match found then word is collected in positive class. If word not found in positive dictionary then it will be match with negative dictionary on success word is collected in negative class. If previous cases are not fulfill then it is declared as neutral one. The information about sentiment is conveyed by adjectives or more specifically by certain combinations of adjectives with other parts of speech. This task of sentiment analysis is performed in this module.

6.4. Result And Visualization

After that the graphical result is created using positive, negative and neutral count. The graphical result shows sentiment of the corresponding news article. From this sentiment it is determined whether the article is positive, negative or neutral.

6.5. Relevant Diagrams

6.5.1. Use Case Diagram

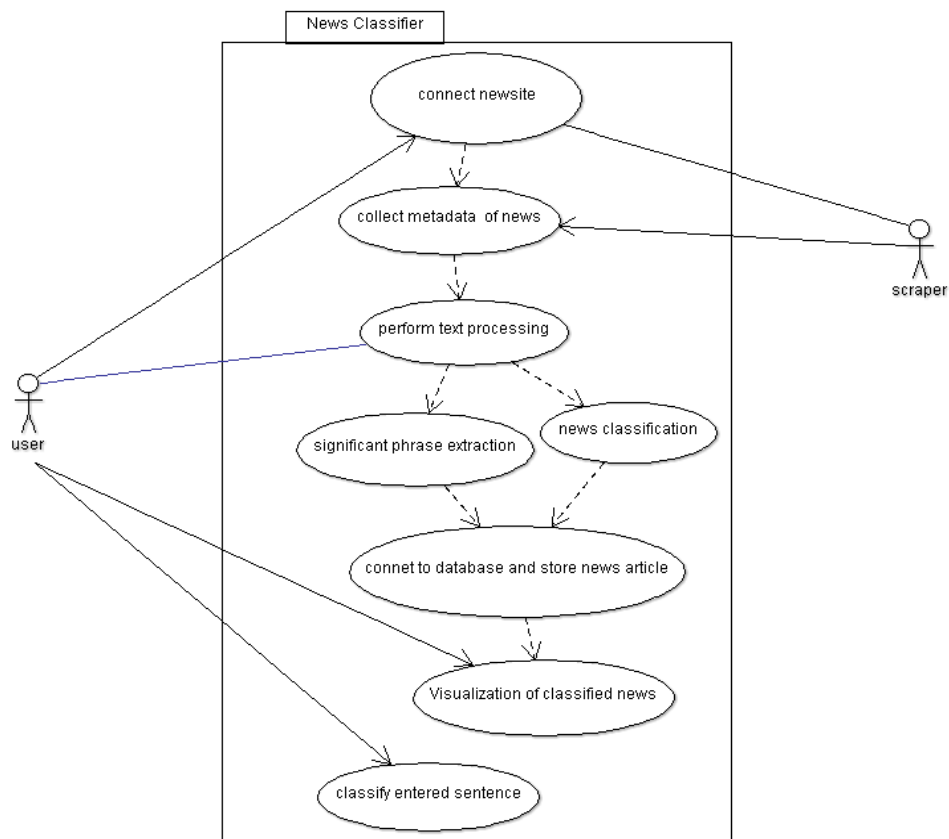


Fig. 6.2: Use Case Diagram

6.5.2. Data Flow Diagram

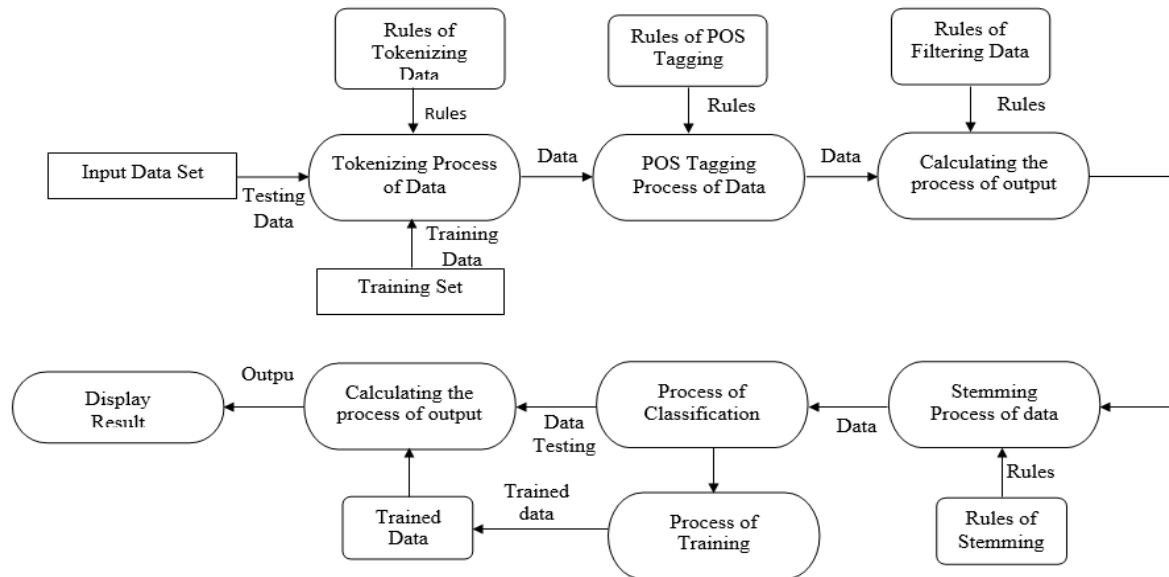


Fig. 6.3: Data Flow Diagram

6.5.3. Sequence Diagram

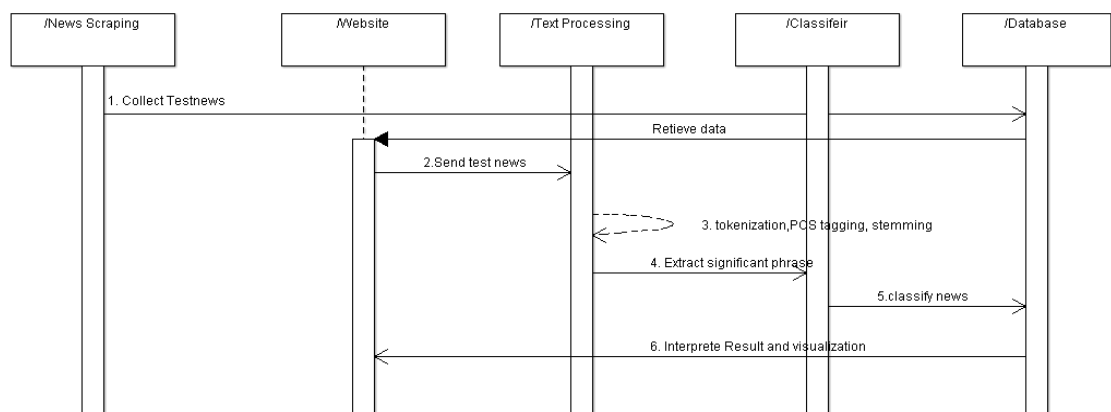


Fig. 6.4: Sequence Diagram

7. DEVELOPMENT IMPLEMENTATION PROCESS

In this section we explain sequentially the entire development process and the activities involved in it. We have to go through a series of approaches for implementation sentiment analysis. Some of the approaches are News headline extraction, text processing, training classifier model, using classifier model for test data.

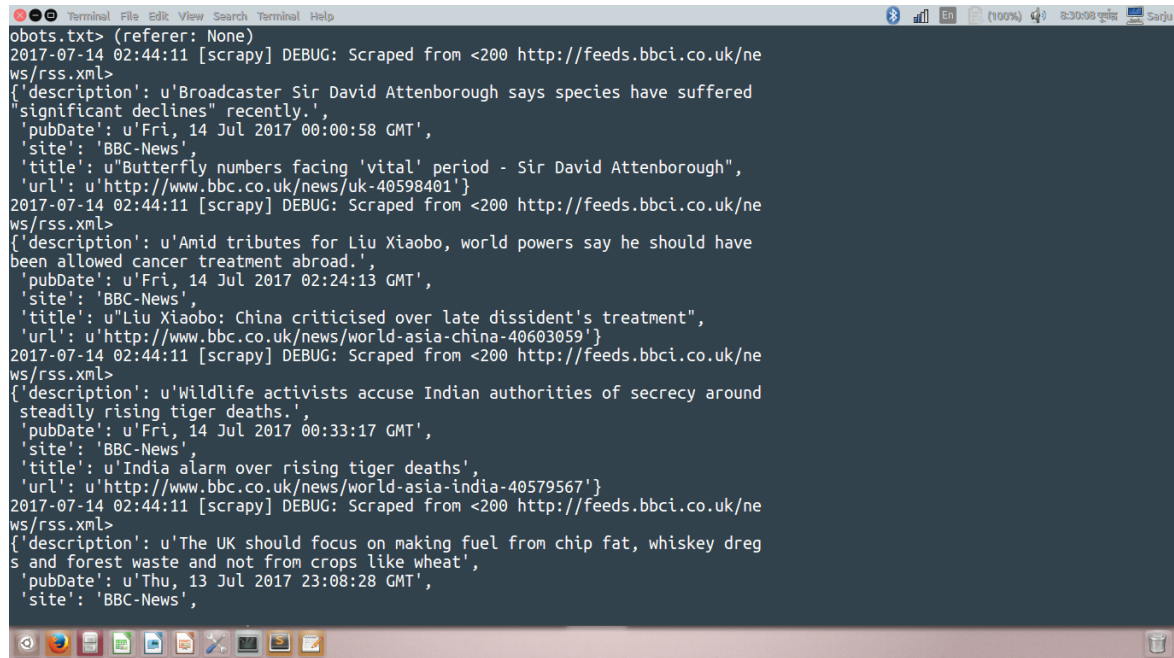
Among these approaches/problems, some of them are solved which are explained below:

7.1. Web Crawling

A web crawler (also known as a web spider or web robot) is a program or automated script which browses the World Wide Web in a methodical, automated manner. This process is called Web crawling or spidering. Many legitimate sites, in particular search engines, use spidering as a means of providing up-to-date data.

In context of our project we have need the web crawling for the scraping the news headline from the news web blogger for sentiment analysis to determine either the news is positive or negative. For this we have used Scrapy as a web crawler. Since our project is web base application so we have to implement a web development. For this we choose Django as a web development tool since it based on python programming.

Using Django and scrapy we have initially crawled the news title from wikinews.org and saved to the mysql database and the output is shown in fig: 7.1



```

obots.txt> (referer: None)
2017-07-14 02:44:11 [scrapy] DEBUG: Scraped from <200 http://feeds.bbc.co.uk/news/rss.xml>
{'description': u'Broadcaster Sir David Attenborough says species have suffered significant declines" recently.',
 'pubDate': u'Fri, 14 Jul 2017 00:00:58 GMT',
 'site': 'BBC-News',
 'title': u"Butterfly numbers facing 'vital' period - Sir David Attenborough",
 'url': u'http://www.bbc.co.uk/news/uk-40598401'}
2017-07-14 02:44:11 [scrapy] DEBUG: Scraped from <200 http://feeds.bbc.co.uk/news/rss.xml>
{'description': u'Amid tributes for Liu Xiaobo, world powers say he should have been allowed cancer treatment abroad.',
 'pubDate': u'Fri, 14 Jul 2017 02:24:13 GMT',
 'site': 'BBC-News',
 'title': u"Liu Xiaobo: China criticised over late dissident's treatment",
 'url': u'http://www.bbc.co.uk/news/world-asia-china-40603059'}
2017-07-14 02:44:11 [scrapy] DEBUG: Scraped from <200 http://feeds.bbc.co.uk/news/rss.xml>
{'description': u'Wildlife activists accuse Indian authorities of secrecy around steadily rising tiger deaths.',
 'pubDate': u'Fri, 14 Jul 2017 00:33:17 GMT',
 'site': 'BBC-News',
 'title': u'India alarm over rising tiger deaths',
 'url': u'http://www.bbc.co.uk/news/world-asia-india-40579567'}
2017-07-14 02:44:11 [scrapy] DEBUG: Scraped from <200 http://feeds.bbc.co.uk/news/rss.xml>
{'description': u'The UK should focus on making fuel from chip fat, whiskey dregs and forest waste and not from crops like wheat',
 'pubDate': u'Thu, 13 Jul 2017 23:08:28 GMT',
 'site': 'BBC-News',

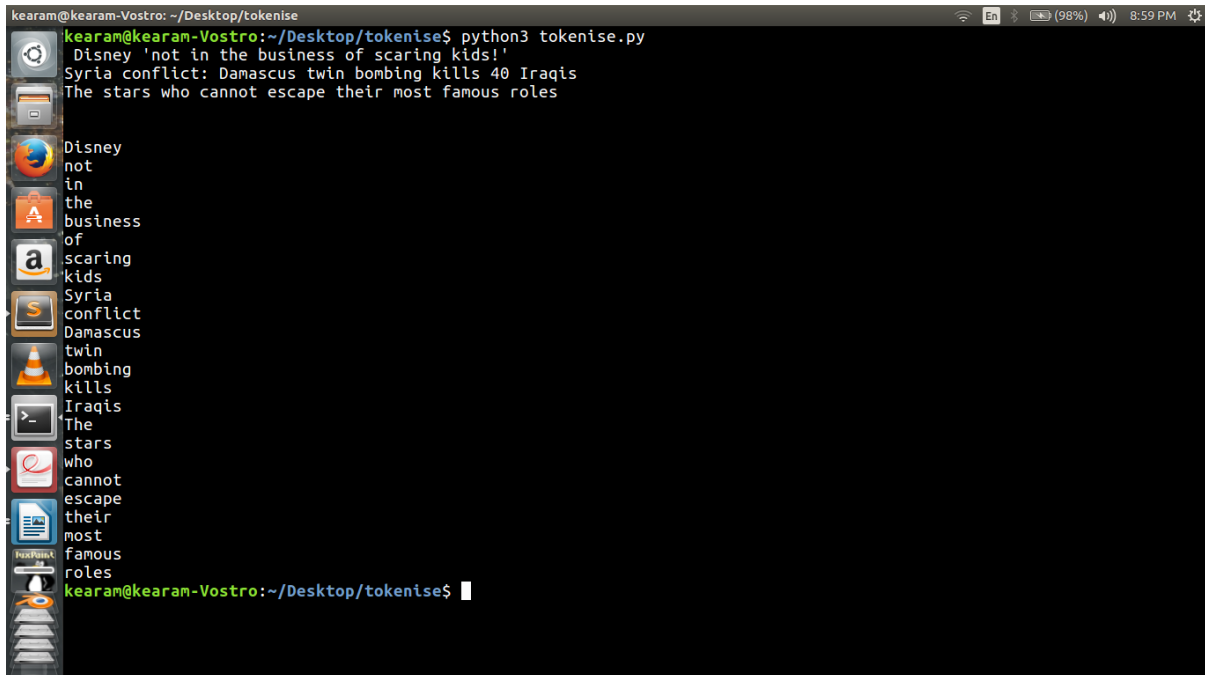
```

Fig. 7.1: scraping title, article

7.2. Text Processing

7.2.1. Tokenization

The headlines of news article are tokenized after scraping the news headline from respective website. The sample output of Tokenization is shown in fig: 7.2



```

kearam@kearam-Vostro: ~/Desktop/tokenise
kearam@kearam-Vostro:~/Desktop/tokenise$ python3 tokenise.py
Disney 'not in the business of scaring kids!'
Syria conflict: Damascus twin bombing kills 40 Iraqis
The stars who cannot escape their most famous roles
kearam@kearam-Vostro:~/Desktop/tokenise$

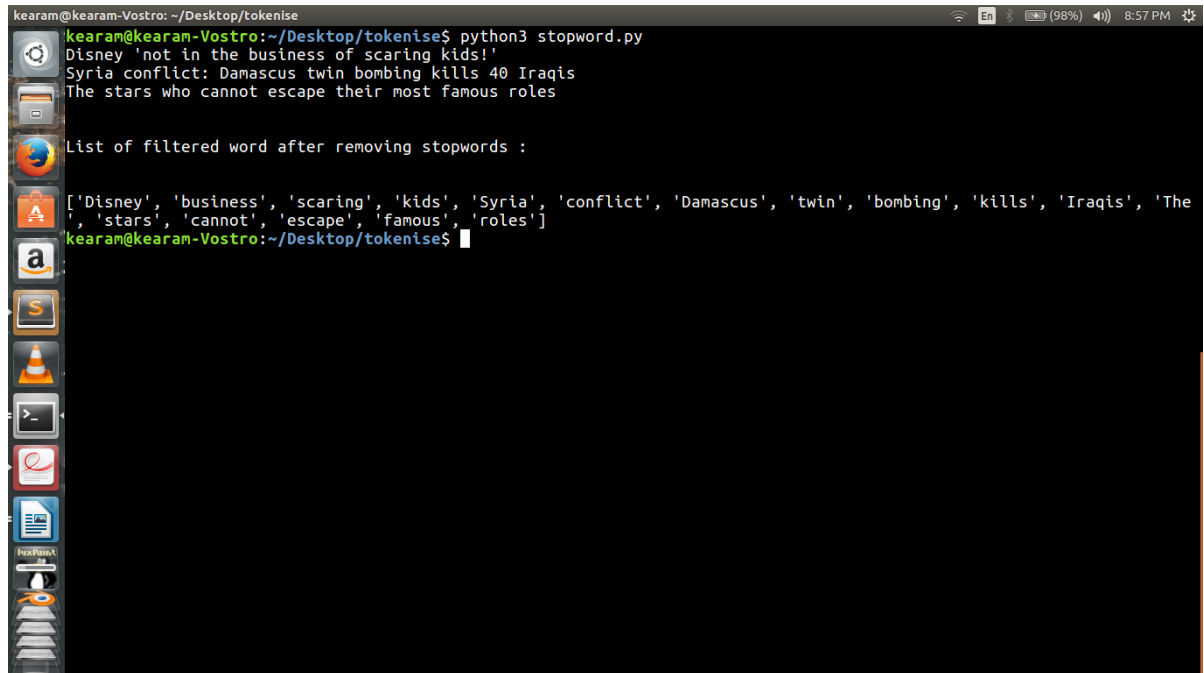
```

Fig. 7.2: Sample output of Tokenization

7.2.2. Filtering Words

Filtering is essentially the process of taking only what we need from what we have. We have used library function `stopword` from `nltk` in order to remove stopword such as `a`, `an`, `in`, `or`, `the`, `will`, etc. We have successfully remove the non-relevant word which shall not be used by classification algorithm. But still the filtered word consist non relevant word. So, we should further use POS tagging method to extract adjective, verbs, and interjection which bear semantics of the document.

Sample output of the filtration process is shown in fig: 7.3



```

kearam@kearam-Vostro: ~/Desktop/tokenise
kearam@kearam-Vostro:~/Desktop/tokenise$ python3 stopword.py
Disney 'not in the business of scaring kids!'
Syria conflict: Damascus twin bombing kills 40 Iraqis
The stars who cannot escape their most famous roles

List of filtered word after removing stopwords :

['Disney', 'business', 'scaring', 'kids', 'Syria', 'conflict', 'Damascus', 'twin', 'bombing', 'kills', 'Iraqis', 'The', 'stars', 'cannot', 'escape', 'famous', 'roles']
kearam@kearam-Vostro:~/Desktop/tokenise$

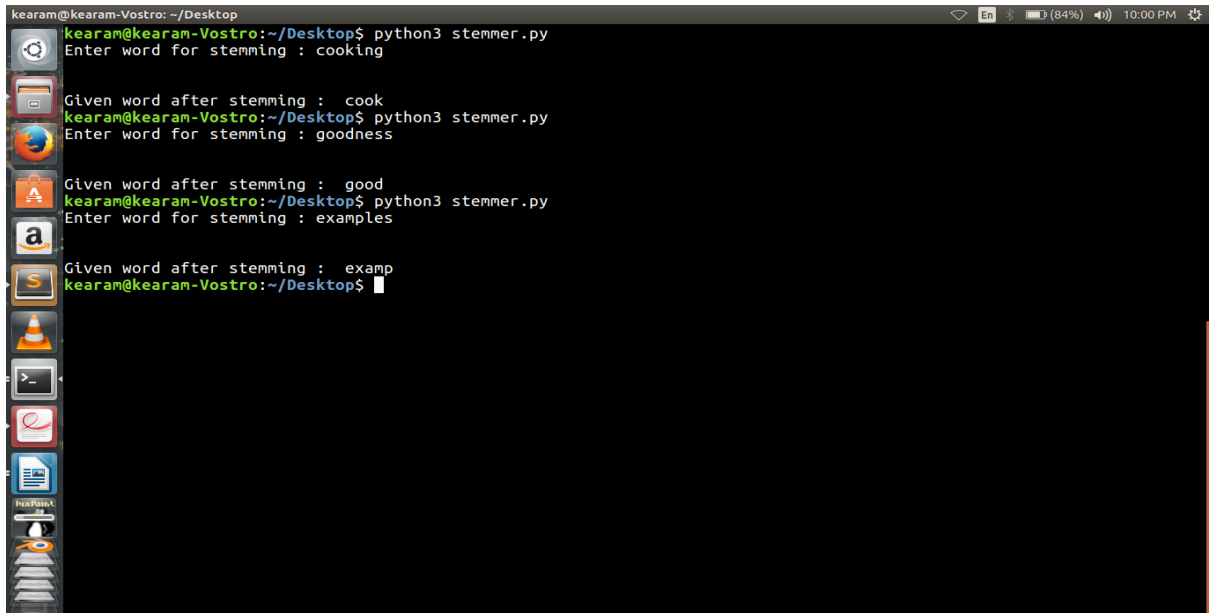
```

Fig. 7.3: Sample output of the filtration

7.2.3. Stemming

Stemming is the process of removing the suffixes of word in order to reduce into root words. We have used Porter Stemming Algorithm to implement the stemming process which takes any word as an input to return base form of corresponding word. The tokens are the just collections of words and does contain redundancy. Therefore, we would love to remove unnecessary suffixes to improve output accuracy with the decrease of word redundancy. We have seen that the word ‘cooking’ and ‘goodness’ are converted into ‘cook’ and ‘good’ without changing its original meaning.

Doing this makes easier to separate a class of words which bear a similar semantic in the classification process. The base form of words might not be meaningful words themselves in some cases. Here in output sample the word ‘examples’ is stemmed into ‘examp’ (expected: example) which is meaningless. The sample output of stemming is shown in fig: 7.4



```

kearam@kearam-Vostro: ~/Desktop$ python3 stemmer.py
Enter word for stemming : cooking
Given word after stemming : cook
kearam@kearam-Vostro:~/Desktop$ python3 stemmer.py
Enter word for stemming : goodness
Given word after stemming : good
kearam@kearam-Vostro:~/Desktop$ python3 stemmer.py
Enter word for stemming : examples
Given word after stemming : examp
kearam@kearam-Vostro:~/Desktop$

```

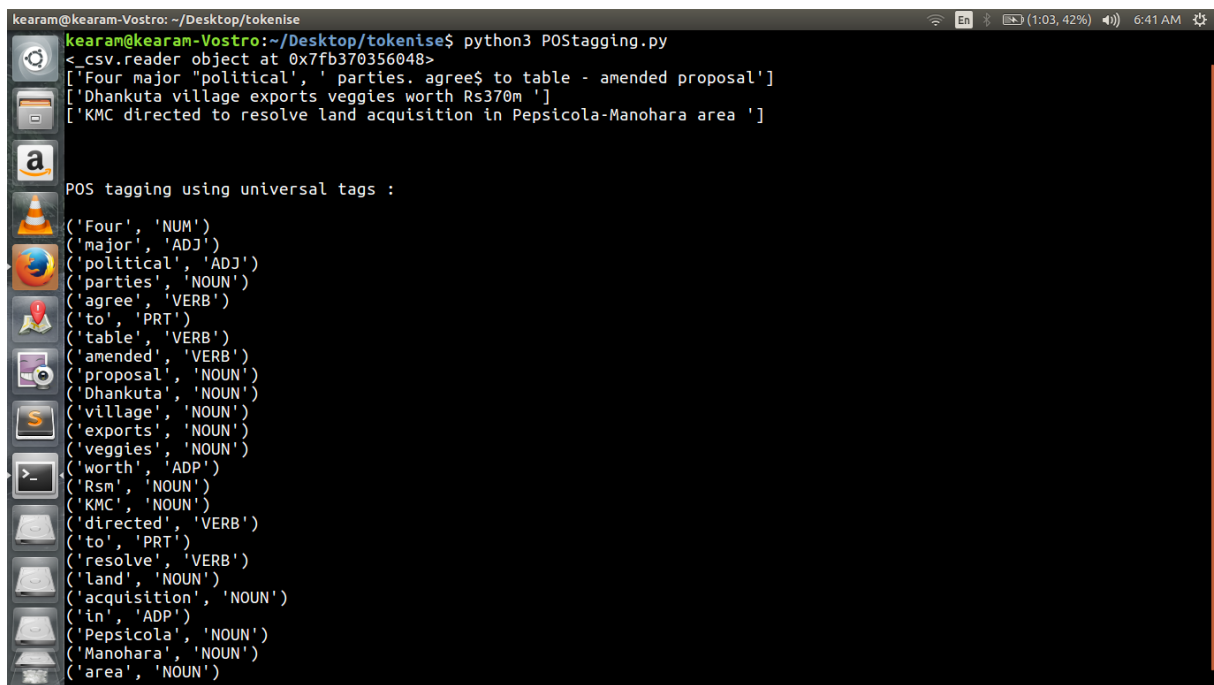
Fig. 7.4: Sample output stemming

7.2.4. POS Tagging

POS stands for Part of speech. A part of speech is a special designation provided to a word that defines its functional aspect in a sentence. Identifying the POS is essential, as certain words are more likely to bear the gist of a sentence than others. For instance, directives such as ‘A’, ‘The’ etc. are not the semantic part of a sentence. The meaning of the sentence is more often than not carried by adjectives and verbs. In this development process we have used Porter Stemmer Algorithm which we used to obtain the base forms of word, their parts of speech which mark up the structure of sentences in terms of phrases etc.

So, we were able to mark up words in a text (corpus) corresponding to its particular part of speech, based on the nature in the definition and context of the sentences. It has able to categorize them successfully by marking the acronyms, or short-forms of the parts-of-speech. Each part-of-speech has its definite acronym in the form of abbreviation which makes quite easier to know the nature of words. This methodology helps to know every form of part-of-

speech such as: noun, pronoun, verb, adverb, adjective, interjection, conjunction, preposition, and the others. But, we intend to take verbs, adjective, and interjections mostly, as they carry the meaning of the document. The Sample output of POS-Tagging can be depicted in the screenshot presented in fig: 7.5



```

kearam@kearam-Vostro: ~/Desktop/tokenise
kearam@kearam-Vostro:~/Desktop/tokenise$ python3 POS tagging.py
<csv.reader object at 0x7fb370356048>
['Four major "political", ' parties. agree$ to table - amended proposal']
['Dhankuta village exports veggies worth Rs370m ']
['KMC directed to resolve land acquisition in Pepsicola-Manohara area ']

POS tagging using universal tags :

('Four', 'NUM')
('major', 'ADJ')
('political', 'ADJ')
('parties', 'NOUN')
('agree', 'VERB')
('to', 'PRT')
('table', 'VERB')
('amended', 'VERB')
('proposal', 'NOUN')
('Dhankuta', 'NOUN')
('village', 'NOUN')
('exports', 'NOUN')
('veggies', 'NOUN')
('worth', 'ADP')
('Rs370m', 'NOUN')
('KMC', 'NOUN')
('directed', 'VERB')
('to', 'PRT')
('resolve', 'VERB')
('land', 'NOUN')
('acquisition', 'NOUN')
('in', 'ADP')
('Pepsicola', 'NOUN')
('Manohara', 'NOUN')
('area', 'NOUN')

```

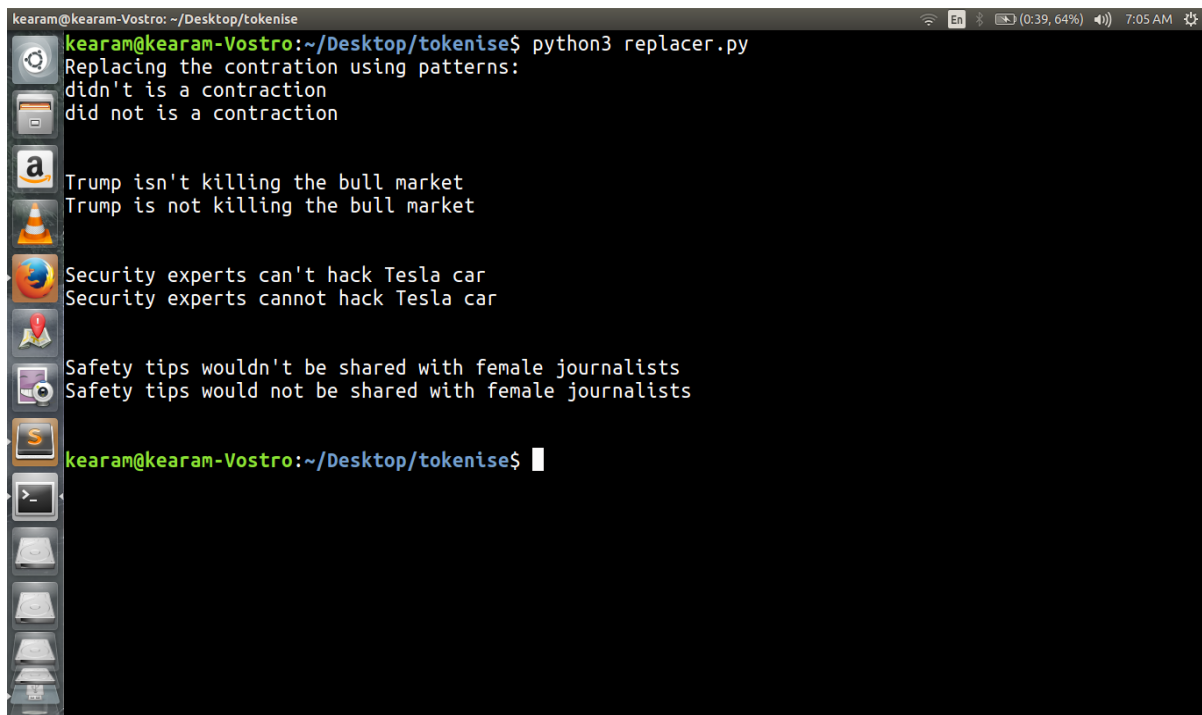
Fig. 7.5: Sample output of POS-Tagging

7.2.5. Replacing Word Matching regular expressions

Now, we are going to get into the process of replacing words. If stemming is a kind of linguistic compression, then word replacement can be thought of as error correction or text normalization. We will replace words based on regular expressions, with a focus on expanding contractions. When we were tokenizing words, Tokenizing text and it was clear that most tokenizers had trouble with contractions? This technique aims to fix this by replacing contractions with their expanded forms, for example, by replacing "can't" with "cannot" or "would've" with "would

have" how this code works will require a basic knowledge of regular expressions and the re module. The key things to know are matching patterns and the re.sub() function.

First, we need to define a number of replacement patterns. This will be a list of tuple pairs, where the first element is the pattern to match with and the second element is the replacement. The Sample output of Replacers is presented in fig: 7.6



```
kearam@kearam-Vostro: ~/Desktop/tokenise
kearam@kearam-Vostro:~/Desktop/tokenise$ python3 replacer.py
Replacing the contraction using patterns:
didn't is a contraction
did not is a contraction

Trump isn't killing the bull market
Trump is not killing the bull market

Security experts can't hack Tesla car
Security experts cannot hack Tesla car

Safety tips wouldn't be shared with female journalists
Safety tips would not be shared with female journalists

kearam@kearam-Vostro:~/Desktop/tokenise$
```

Fig 7.6: Sample output of replacing contraction

7.3. Web Interface

7.3.1. Home page

The home page has been designed using bootstrapping to make home page as simple and attractive. Bootstrapping is the powerful web page development tool for dynamic web development. The visualization section has to build. The web page demo is implemented in real web.

Our next phase for web development is done after creation of the database model (sqlite) for news classification. Then, the database model is used for the implementation of Naive Bayes classifier.

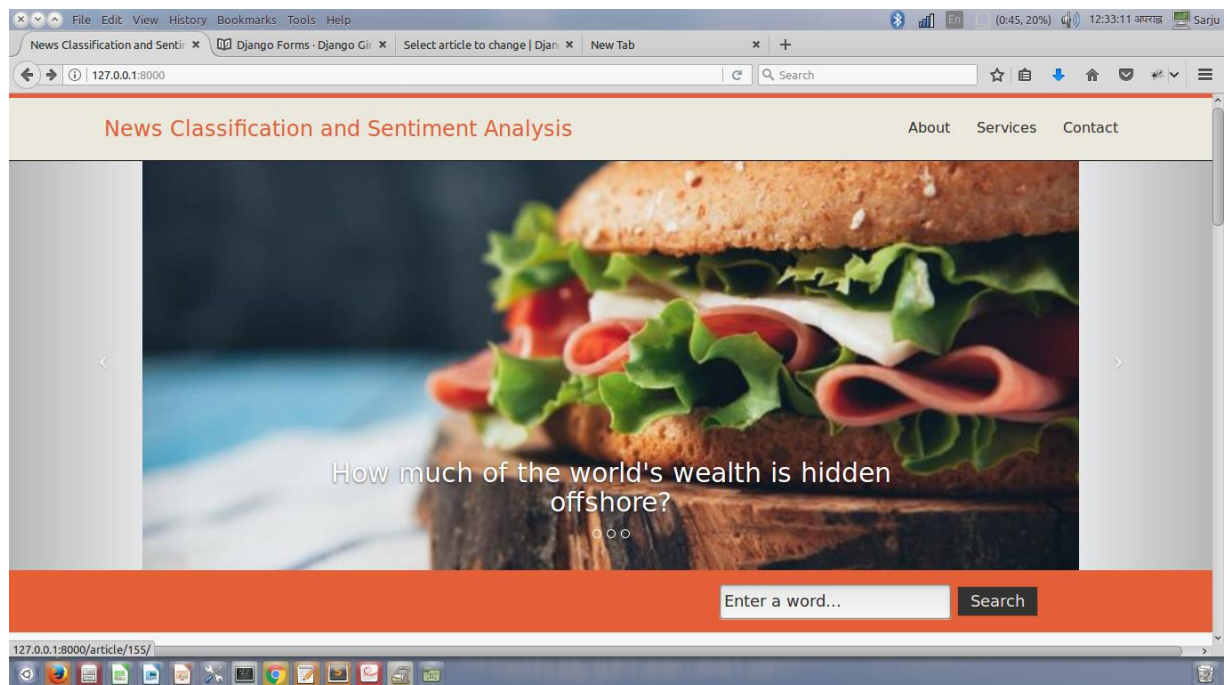


Fig. 7.7: Home page

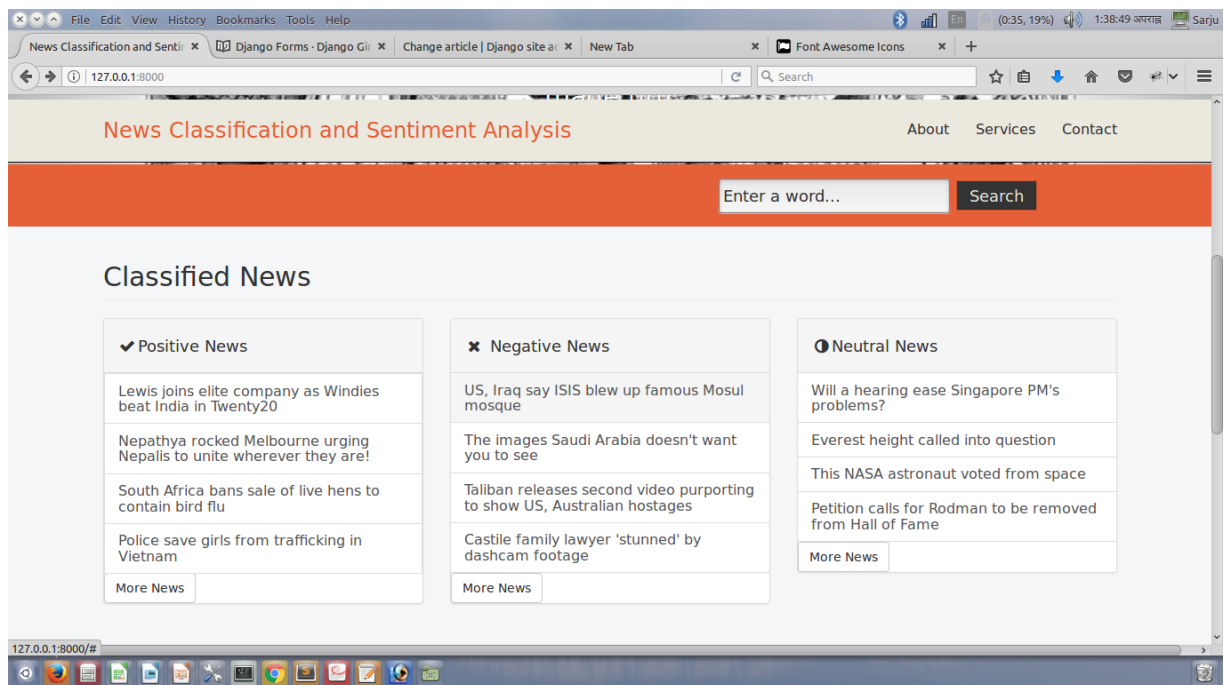


Fig. 7.8: Sample of home page showing news categories

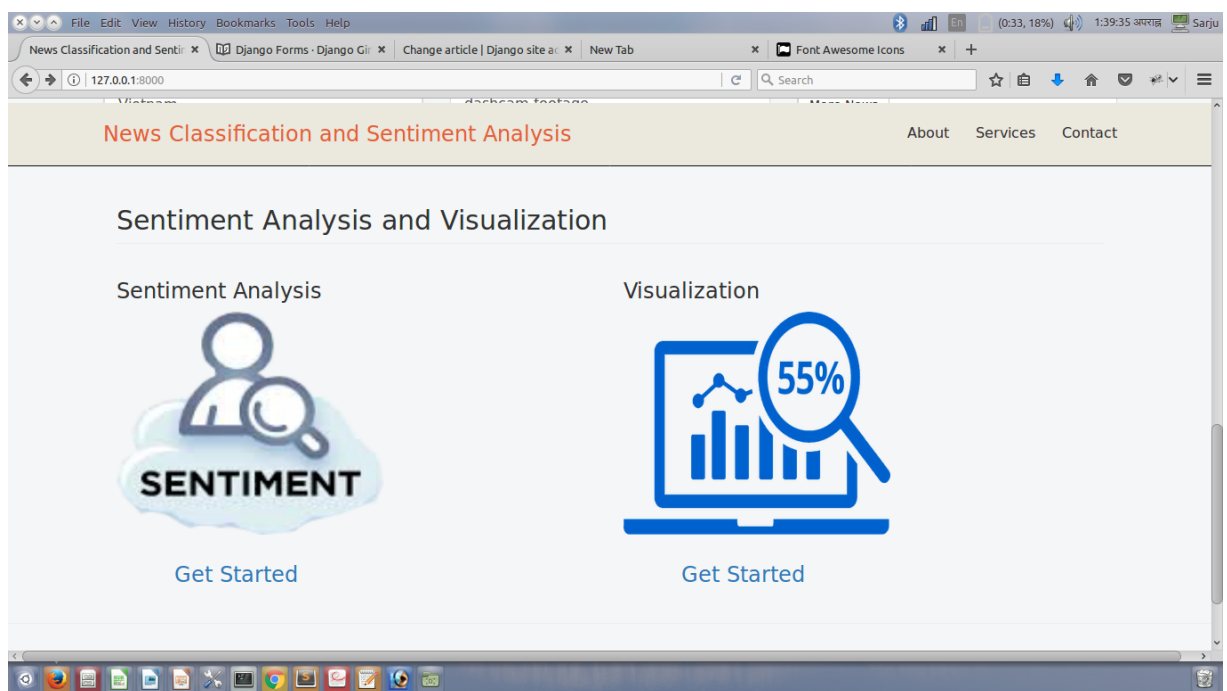


Fig. 7.9: Sample of home page showing sentiment analysis

News Display

News article scraped first classified to positive, negative, neutral classes. The classified news have shown in home page. From the home page the news can be redirected to article page. The sample is shown in fig: 7.10

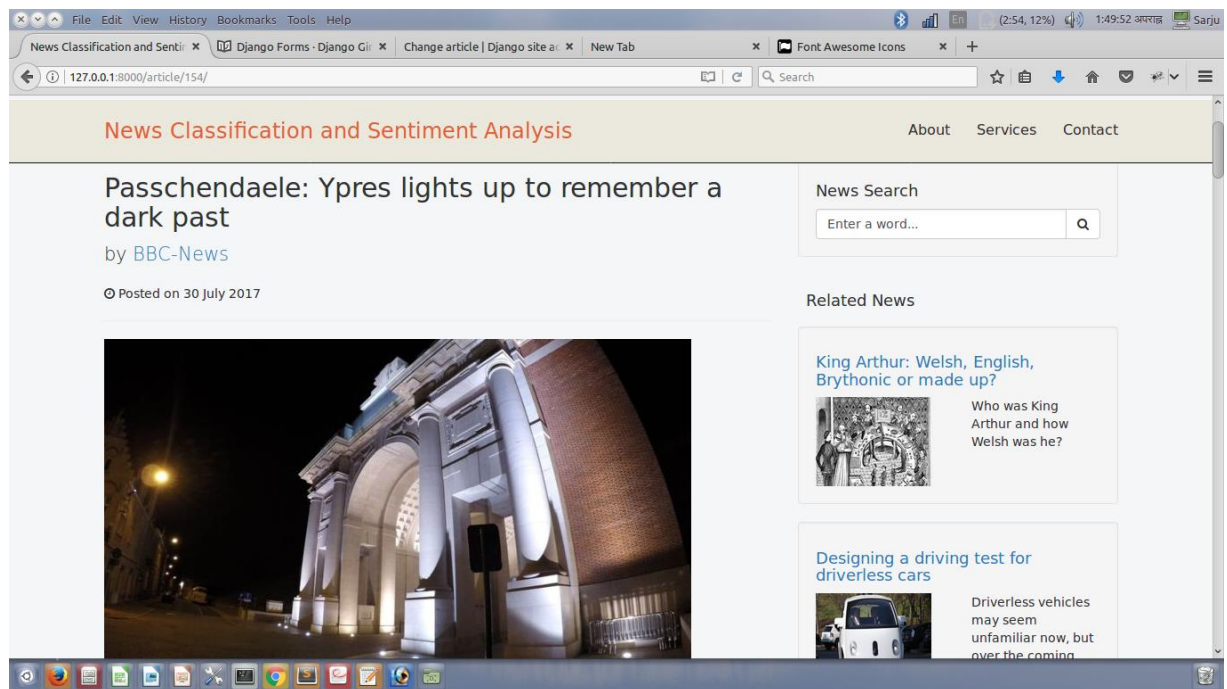


Fig. 7.10: Sample of news article display

7.3.2. Visualization

The visualization of the accuracy and the comparison between two classifier (naïve bayes and if-tdf using KNN) is displayed and sample is shown in fig: 7.11

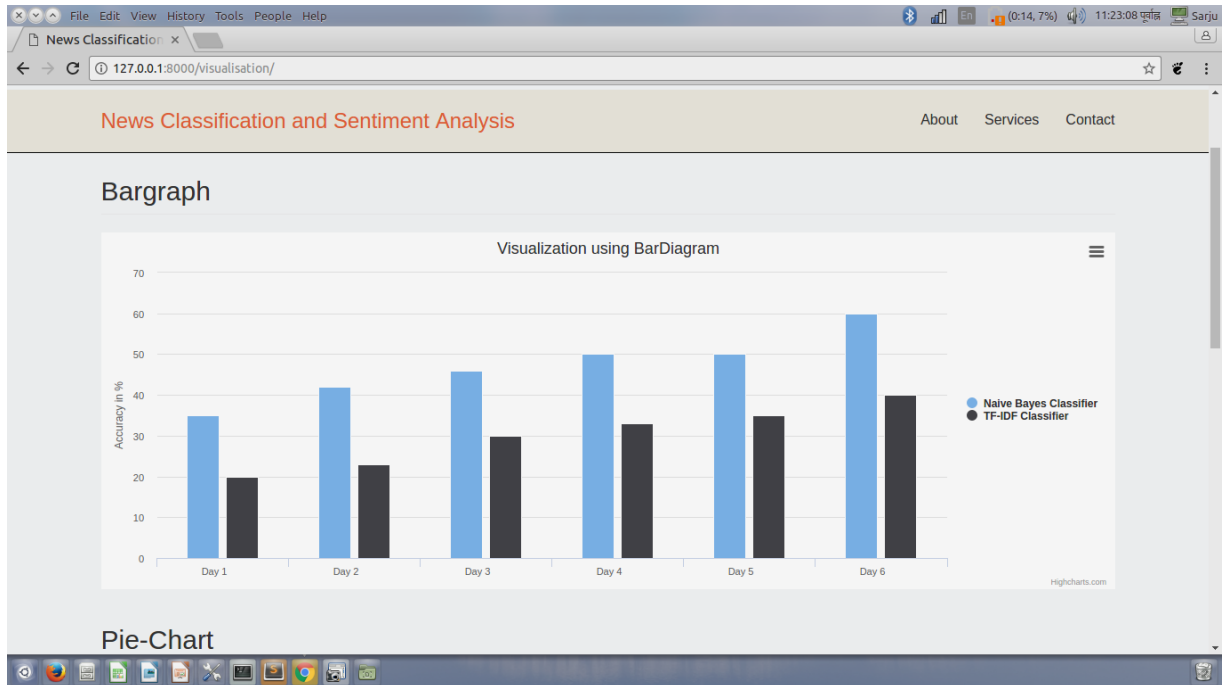


Fig. 7.11: Visualization using BarDiagram

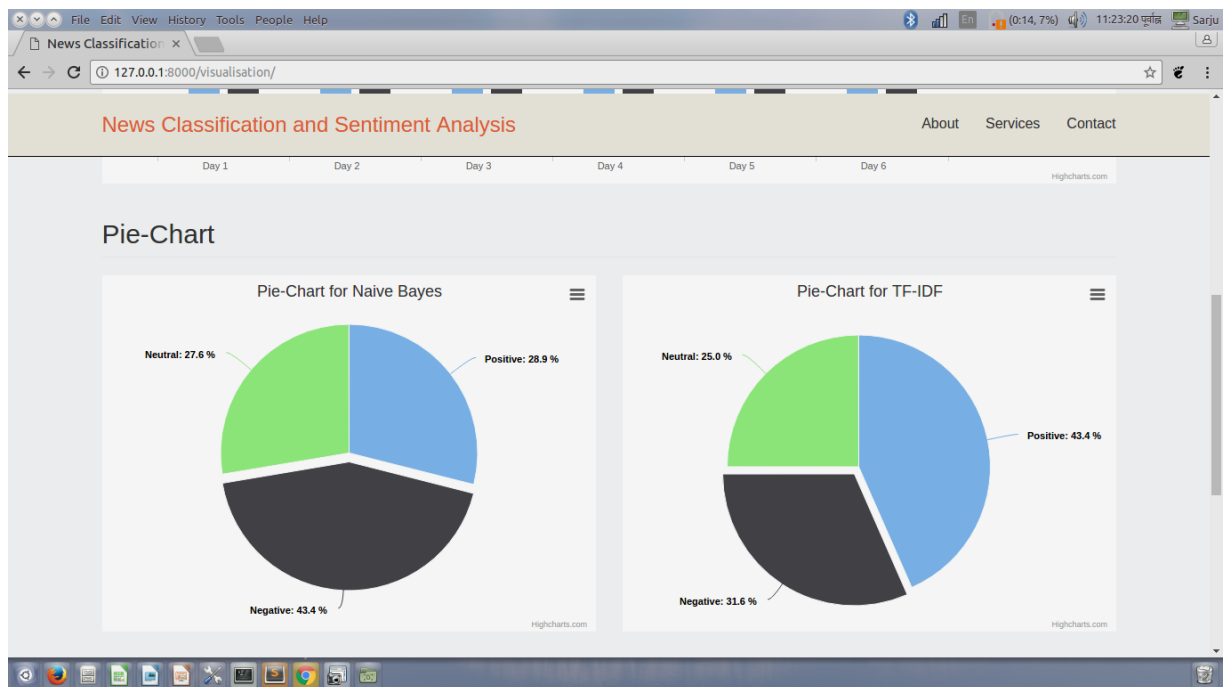


Fig. 7.12: Visualization using Pie-Char

Result

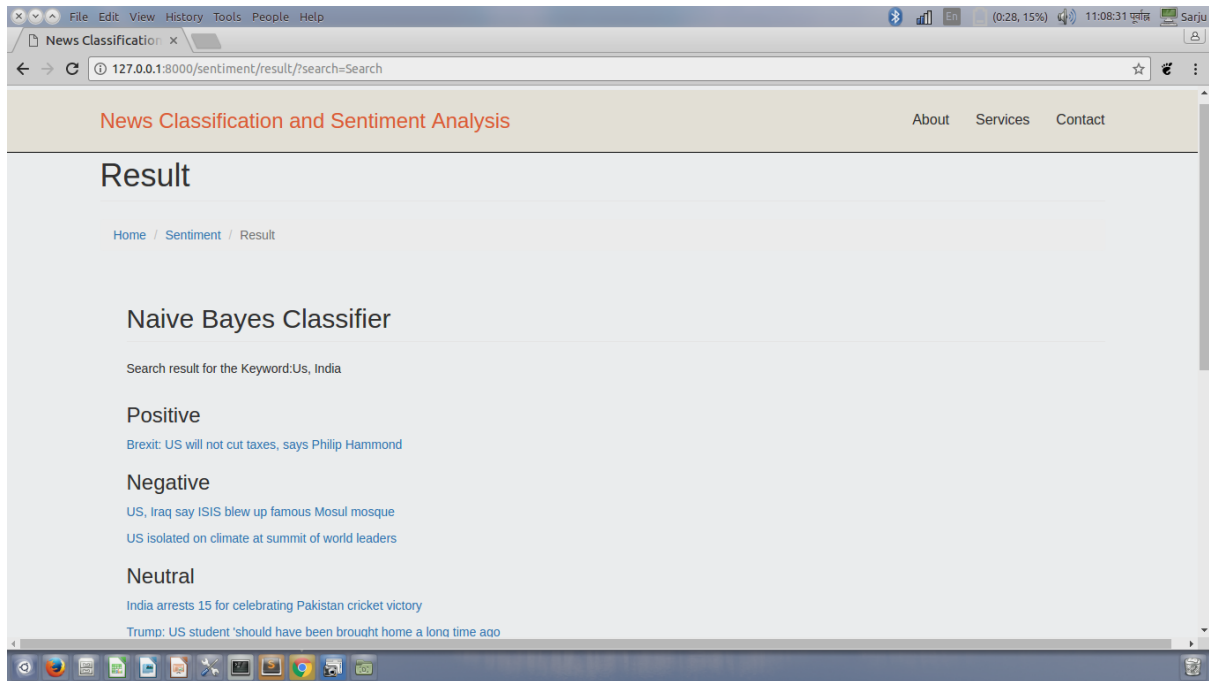


Fig. 7.13: Result display by Naïve Bayes Classifier

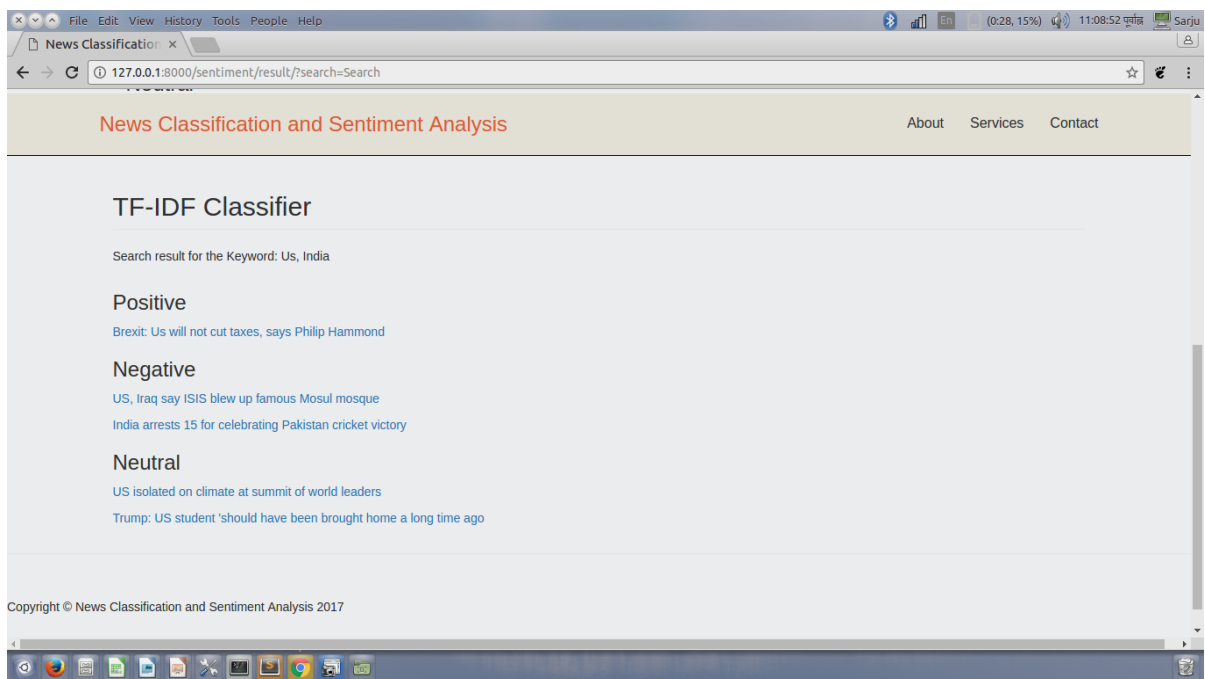
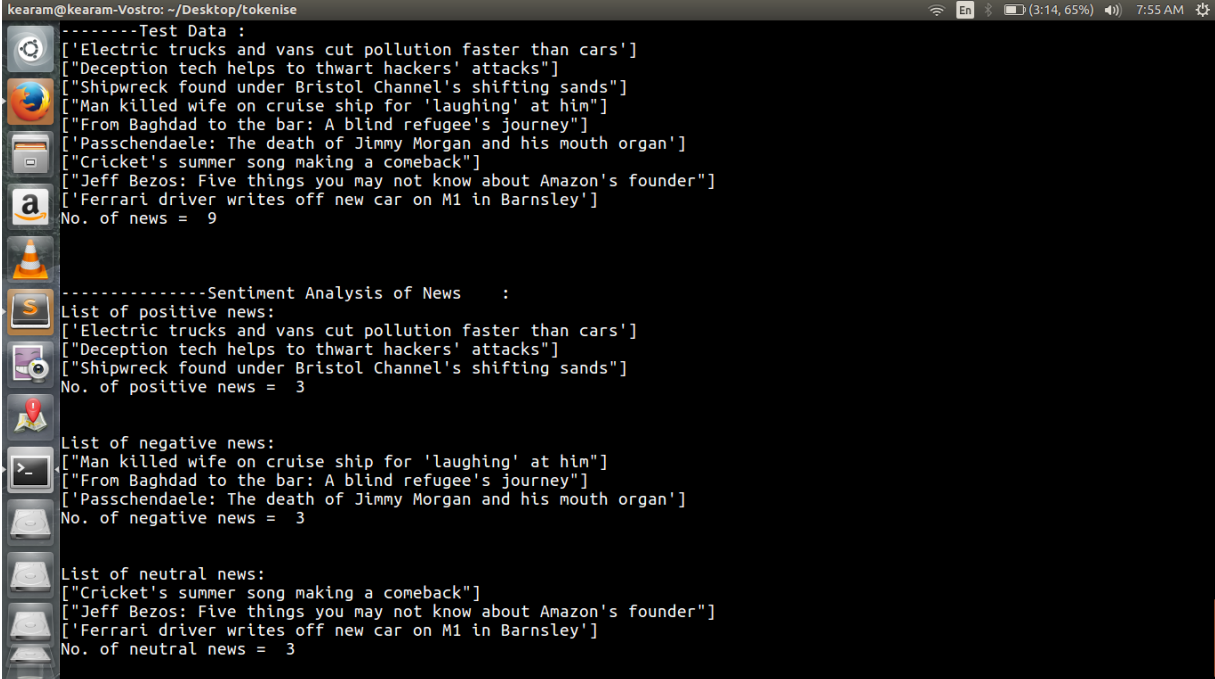


Fig. 7.14: Result Display by TF-IDF classifier

7.4. Naïve Bayes

The classifier is developed and tested in real world problem. Since the test data is not sufficient and is not so good the result from the classifier has to improve little bit more. The naive bayes classifier classifies news by its title. News titles (test data) are first tokenized, stemmed, POS tagged and classified to its related class. The verb, adverb, adjectives are extracted from the news headlines (test data) and compared with training data for classification. So there is something irrelevant result for those news which does not have any verb, adjective, adverbs.

Result of classifier for real data is demonstrated here in the form of screen shots in fig: 7.15



```

kearam@kearam-Vostro: ~/Desktop/tokenise
-----Test Data :
['Electric trucks and vans cut pollution faster than cars']
['Deception tech helps to thwart hackers' attacks']
['Shipwreck found under Bristol Channel's shifting sands']
['Man killed wife on cruise ship for 'laughing' at him']
['From Baghdad to the bar: A blind refugee's journey']
['Passchendaele: The death of Jimmy Morgan and his mouth organ']
['Cricket's summer song making a comeback']
['Jeff Bezos: Five things you may not know about Amazon's founder']
['Ferrari driver writes off new car on M1 in Barnsley']
No. of news = 9

-----Sentiment Analysis of News :
List of positive news:
['Electric trucks and vans cut pollution faster than cars']
['Deception tech helps to thwart hackers' attacks']
['Shipwreck found under Bristol Channel's shifting sands']
No. of positive news = 3

List of negative news:
['Man killed wife on cruise ship for 'laughing' at him']
['From Baghdad to the bar: A blind refugee's journey']
['Passchendaele: The death of Jimmy Morgan and his mouth organ']
No. of negative news = 3

List of neutral news:
['Cricket's summer song making a comeback']
['Jeff Bezos: Five things you may not know about Amazon's founder']
['Ferrari driver writes off new car on M1 in Barnsley']
No. of neutral news = 3

```

Fig. 7.15: Sample output of Naïve bayes Classifier

8. RESULT

Initially, 100 news titles were classified manually into positive, negative and neutral news. Then, the news title were classified using Naïve Bayes algorithm and TF-IDF classifier. These two results were interpreted to compare the accuracy of each classification algorithm. The result provide by Naive Bayes algorithm quite accurate with respect to TF-IDF classifier. Naïve Bayes make 56 per cent accuracy on the charts. Out of 35 positive documents, 22 were correctly classified. The negative and neutral news were correctly classified in fractions of 27 out of 42 and 10 out of 23 respectively.

The TF-IDF classifier was able to follow up the Naive Bayes classifier, and stays at 43 per cent of accuracy. The same positive, negative and neutral documents were fed into the TF-IDF classifier and we got 16 out of 35 positive news correctly classified. Negative and neutral news were correctly classified 19 out of 42 and 9 out of 23.

The results are listed below:

(i) **Naive Bayes:** 59/100 Correct

- For positive news: 22/35 correct
- For negative news: 27/42 correct
- For neutral news: 10/23 correct

(ii) **TF-IDF classifier:** 44/100 Correct

- For positive news: 16/35 correct
- For negative news: 19/42 correct
- For neutral news: 9/23 correct

8.1. Performance For Positive News

Naive Bayes obtained 62 per cent accuracy on positive news while TF-IDF classifier produced 45.0 per cent accuracy. With the small number of sample titles analyzed, it's hard to draw comparisons between them. However, it is easy to see that Naive Bayes has performed well as compared to TF-IDF classifier. It can be attributed to the fact that positive news are generally harder to spot and also tend to have more room for ambiguity.

8.2. Performance For Negative News

Naive Bayes performed exceptionally 64 per cent well on detecting negative news while the TF-IDF classifier performed pretty slightly 45 per cent. It can be attributed to the fact that negative news are generally easy to spot and has more number of words whose occurrence is highly related to the title being negative. For e.g.:- words like "death", "killed", etc. are very likely to be found in negative titles only. So, the bag of word model that the Naive Bayes uses suits it very well.

8.3. Performance For Neutral News

Both Naive Bayes (43 per cent) and TF-IDF classifier (39.54 per cent) performed exceptionally well in identifying neutral news. One of the main contributing factors for this seemingly high accuracy is the simple reason that most news tend to be neutral that inform us of something without evoking any positive or negative sentiment.

8.4. Overall Performance

Overall, Naive Bayes achieved an accuracy of 59 per cent while TF-IDF classifier performed poorer with an accuracy of 44 per cent. So, on a larger and random sample, the accuracy can be expected to improve to some extent.

9. LIMITATION

Sentiment analysis is the technology traditionally used to obtain such learning, sorts complex, human-generated data into positive, negative, and neutral categories. This system is too simplistic to represent human expression, and limits predictive power by misrepresenting a significant portion of data. The results of such analyses are not very reliable nor actionable. There are also no suppression of multiple emotions.

According to a random sampling of social media posts in the month of June, over 50 per cent of human data contained multiple emotions, and 30 per cent contained strongly contrasting emotions. Sentiment analysis incapable of representing data with multiple emotions, and in cases of conflicting emotions, as in the above example, it averages these polarities (positive + negative), producing a neutral signal for data that is just the opposite. Some of the drawbacks of our system are discussed here. As this project was based on supervised machine learning, human error might have affected the system. So, the accuracy of the project was 59 per cent .The project provides false results for complicated statements like sarcasm as it is difficult to train our project for such statements.

10. CONCLUSION

In this project, the proposed solution and implementation of the system justify that we can build our platform to classify news on dynamic data set of news corpus on broad range of topics. Satisfactory performance can be obtained from document classification by increasing training data set.

News articles were crawled from news website using scrapy to generate the training data for supervised learning. Stemming, tokenization and POS tagging under NLP were implemented along Naive Bayes classifier and IF-IDF classifier as machine learning approaches in order to classify the news corpse. Finally the classified news were interpreted and the information were visualize in the form of bar graphs and pie-chart.

The correctness and validity of Naive Bayes classifier and IF-IDF classifier algorithms were tested and actually quite satisfactory results were obtained. In a nutshell, this project provide a GUI based platform for serving positive news to create a positive environment and help the light-hearted news reader to avoid violent news.

11. FUTURE WORK

Sentiment of a news article varies per user. It is quite possible that news which is negative to one user can be found as positive to another user. It depends on user perception of the article and his/her thought. So to support this, we can create account for each user and maintain his/her profile. User can put a rating to each article that he read. If user finds any news article as negative, he can rate the article from one star to 5 star or mark article as negative. This user feedback will be taken into account for serving new news articles to same user in future.

It means that system would be self-learned from this feedback.

12. GANTT CHART

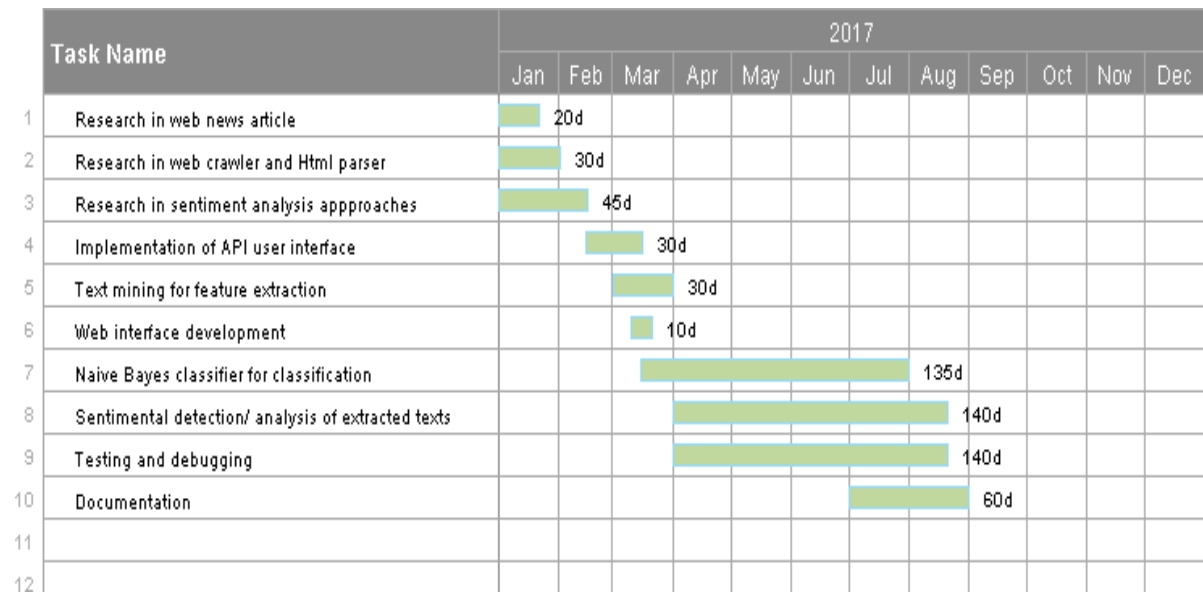


Fig. 12.1: Gantt Chart

APPENDIX

Appendix A: POS-Tagging Labels

The labels of POS-Tagging are shown in the below table:

S.N.	Label	Meaning
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP\$	Possessive pronoun
20.	RB	Adverb

21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	To
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3 rd person singular present
32.	VBZ	Verb, 3 rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WPS	Possessive wh-pronoun

REFERENCES

- [1] B. Pang and L. Lee., “Opinion mining and sentiment analysis”, Foundations and Trends in Information Retrieval, vol. 2, no. 1-2, pp. 1-135, 2008.
- [2] N. Godbole, M. Srinivasaiah, and S. Skiena., “Large-scale sentiment analysis for news and blogs,” 2007.
- [3] Simon Fong, Yan Zhuang, Jinyan Li, Richard Khoury,” Sentiment Analysis of Online News using MALLET”, 2013 International Symposium on Computational and Business Intelligence ,24-26 Aug 2013, pp 301-304.
- [4] Xujuan Zhou, Xiaohui Tao, Jianming Yong, Zhenyu Yang , “Sentiment Analysis on Tweets for Social Events”.Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design. 27-29 June 2013, pp 557562.
- [5] Kiran Shriniwas Doddi, Dr. Mrs. Y. V. Haribhakta 2, Dr.ParagKulkarni “Sentiment Classification of News Articles”, Kiran Shriniwas Doddi et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, pp4621-4623
- [6] Wikipedia http://en.wikipedia.org/wiki/POS_tagger
- [7] P. Turney., “Thumbs up or thumbs down?” Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics,pp. 417424, 2002.
- [8] Implementing News Article Category Browsing Based on Text Categorization Technique http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4740747
- [9] J. Kamps, M. Marx, R. Mokken., ”Using WordNet to Measure Semantic Orientation of Adjectives”, LREC 2004, vol. IV, pp. 11151118, 2004.
- [10] M. Bautin, L. Vijayrenu L, and Skenia., “International sentiment analysis for news and blogs”, In Second International Conference on Weblogs and Social Media (ICWSM), 2008.

[11] P. Turney., “Thumbs up or thumbs down?” Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp. 417424, 2002.