A REPORT

ON

SENTIMENT ANALYSIS OF NEWS ARTICLES

BY

2016B3A70340G **Aneesh Garg**

Prepared for partial fulfillment of Course PS-1

AT

Department of Information Technology and Communication
(DoIT&C)

A Practice School - I Station of
BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

21st May, 2018 - 12th July, 2018

TABLE OF CONTENT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI
(RAJASTHAN)
Practice School Division

Station: DoIT&C **Centre:** Jaipur
Duration: 1.5 Months **Date of Start**: 21/5/18
Date of Submission: 12/7/18

Title of the Project: Sentiment Analysis of News Articles

Prepared by:
Aneesh Garg, 2016B3A70340G, Economics and CSE

Name(s) and designation(s) of the expert(s):
Arun Chauhan (Technical Director)
PS Faculty: Satyendra Kumar Sharma
Project Areas: Data analytics and Text Mining

Abstract:
Newspapers are a valuable source of information. It has information that covers some of the important and alarming happenings around us. Example: local newspapers giving information about crimes in different localities. If this information is gathered, organized and interpreted using data mining techniques, this information can be used by the police department for controlling crimes. To get most out of news articles, data analysis methods such as text mining can be used. To what portion of the newspaper the technique is applied is of great importance. Doing data mining on a whole page will not give appropriate results because each page has a lot of unrelated information covered. Doing data mining on individual news items will be a lot easier. In this project, we aim at doing text mining and analyze sentiments of individual news items taken from web. We will make use of Python and R programming languages in order to complete this project.

Signature(s) of Student(s) Signature of PS Faculty
Date: Date:

ACKNOWLEDGMENT

We would like to extend our sincere thanks to DoIT, Jaipur and to our mentor, Mr. Arun Chauhan for giving us opportunity to work on this project and for providing us with constant support and guidance throughout the project.

We would also like to express our gratitude to Mr. Viskas Sharma, Mr. Dheeraj Gaur and Mr. Gaurav for taking out time from their busy schedules to accommodate us into their teams and allotting us the project.

Furthermore, we would also like to thank our PS-1 faculty, Satyendra Sharma for his instrumental role in the smooth functioning of our project work and for providing us with this great opportunity to learn.

We would also like to thank our families for their continuous support which has been instrumental in the fulfillment of this project.

Finally, we would like to thank the Practice School Division, BITS Pilani for providing us with this unique opportunity to work at a Government Department and gain this invaluable experience.

INTRODUCTION:

Organization:

Department of Information Technology and Communication have been very helpful and generous to enhance our practical knowledge and get a hands-on experience on how to get things done. For this and several other favors that are too many to list, we are eternally grateful to this wonderful organization.

To truly know something in today's world we have to invariably delve into the history behind it. Therefore we first take a look at the history of this department.

The Department of Computers was established in 1987. In its nascent stage, it was functioning under the State's Planning Department, to provide state government a strong technical foundation to better serve its citizens and to create more accountability and efficiency through computerization. Thereafter, on 1st April 1988, the Department was reconstituted as Directorate of Computers which, with the enhancement in its scope of work form computerization to IT enablement for framing policy, planning, implementation and monitoring of Information & Communication Technologies and e-Governance projects, was further renamed as Department of Information Technology on 30th September 1997. Consequently, in the year 2002 the Department was rechristened as Department of Information Technology & Communication which is continued till date.

Throughout the recent history, this department has been among one of the most tech-savvy department of the Government. Department of Information Technology and Communication carries out all the Government schemes that help the public make their lives easier. Furthermore, it tries to inculcate use of technology among the public. The prime objective of this department can be stated as follows:

Department of Information Technology & Communication (Department of Information Technology and Communication) is working to put technology to its highest and best use throughout Rajasthan Government Departments/Autonomous Bodies to improve the administration of State

programs and services. Proving guidance on technical matters to Departments, vetting IT projects and taking Department on achieving IT Road Map are the basic jobs of Department of

Information Technology & Communication (Department of Information Technology and Communication).

The Department of Information Technology and Communication is integral in the working of all the Government schemes. The people here handle the database using Hadoop technology. From that database, the data analysts here are able to draw out useful conclusions that help the Government know better about the public. With the help of Mr.Gaurav Kumar we were able to take a look at the technologies used behind these schemes. He showed us how the data collected looks from the inside and how they extract meaningful results from that.

The Department of Information Technology and Communication performs several functions. Some of them are as stated below.

1. Formulation and Implementation of the Information Technology Policy in the state.

2. Appraisal of new technologies and prescribing uniform standards by promotion of investment in Information Technology Sector (hardware, software and services) and related activities and creation and up gradation of Information Technology infrastructure in the State.

3. Assistance in development and implantation of software packages for monitoring of key parameters and computerization of thrust areas in different departments and semi Government organizations.

4. Standardization of hardware/software platforms for the departments/organization and to ensure dynamic monitoring of their prices and minimization of wasteful expenditure.

5. Development of Information Technology related communication infrastructure.

6. Assistance to the departments/semi government organizations in creating and updating websites.

7. Planning of different IT related programme by organizing various promotional activities like national/international conferences/seminars and

participation inthe same.

8. Coordination of all IT projects in the government and follow up of Information Technology related projects/schemes posed to government of India and its agencies and also other players in this field in India as well as abroad.

9. Identification of laws and rules which need to be modified or enacted to enable legal validation and also to act as nodal agency/authority on behalf of the State Government for matters relating to Information Technology Act and similar other central and state legislations.

10. Maintenance of database for all Information Technology related material and human resources available in the state.

11. To organize training programmes for increasing IT literacy among the officers and staff of State Government.

12. Facilitate and Coordinate between various Departments for successful implementation of Mission Mode Projects (MMPs) under National eGovernance Plan (NeGP) and various other eGovernance projects in the State for efficient and effective delivery of government services for better dissemination of information on government functions.

13. Set up Core and Support ICT infrastructure for Government data storage, application hosting and Voice, Video and Data transfers over secure dedicated network for Government services delivery to citizens and businesses as well as efficient functioning of Government Departments.

14. Facilitate coordination Government Departments, Academic Institutions and Industry for achieving excellence in IT and related sectors.

15. Support various Departments for their Capacity Building Initiatives through skilled manpower recruitment and training with respect to implementation and use of ICT and e-Governance projects.

16. Support the State Government Departments in adopting best practices, guidelines, policies and standards vis-a-vis implementation and use of ICT in Government.

Therefore, it is not hard to see how important this department is to the Government and the public in general.

The exemplary vision of this organization can be stated as follows:

The Government of Rajasthan would leverage Information & Communication Technology (ICT) not only as a tool for improving governance and employment opportunities, but also more significantly as a means to enhance the quality of life and bridging the socio-economic divide in the State. The State Government intends to make conscious efforts to see that benefits of IT/ ITeS in terms of employment generation and economic upliftment percolates to all sections of the society, particularly to those living in rural and remote areas.

The e-Governance framework includes:

- Adherence to the vision of IT Policy 2007
- Timely completion of Core NeGP projects in the State
- Standardization and Security Aspects
- Capacity Building
- End-to-end Service Delivery under:
    - Government to Citizen (G2C) Services
    - Business to Citizen (B2C) Services
    - Government to Government (G2G) Services

The best part that we have observed about this department has been the dedication and desire of the employees working here to ensure that there is a digital medium between the Government and the public. The people here are not unlike the builders of a bridge, the bridge between the everyday man and the state administration. The tireless employees here have always been an inspiration for us and have made us realize what being a part of something big and special means.

Data:

Data as a general concept refers to the fact that some existing information or knowledge is *represented* or *coded* in some form suitable for better usage or processing. Data has been described as the new oil of the digital economy.

Raw data ("unprocessed data") is a collection of numbers or characters before it has been "cleaned" and corrected by researchers. Raw data needs to be corrected to remove outliers or obvious instrument or data entry errors (e.g., a

thermometer reading from an outdoor Arctic location recording a tropical temperature). Field data is raw data that is collected in an uncontrolled environment. Experimental data is data that is generated within the context of a scientific investigation by observation and recording. Data is extensively used today in all the fields of research and development.

Data is growing at an incredible rate. Gartner and IDC state that data is doubling every 18 months. Current estimate is that there is over 4 zettabytes of data in the world if the trend continues; by 2020 data will be over 40 zettabytes.

Types of data:

1.) Structured Data:
The data that has a structure and is well organized either in the form of tables or in some other way and can be easily operated is known as structured data. Searching and accessing information from such type of data is very easy. For example, data stored in the relational database in the form of tables having multiple rows and columns.

2.) Unstructured Data:
The data that is unstructured or unorganized operating such type of data becomes difficult and require advance tools and software to access information. For Example, images and graphics, pdf files, word document, audio, video, emails, PowerPoint presentations, web pages and web contents, wikis, streaming data, location coordinates etc.

Department of Information Technology and Communication, Jaipur uses Hadoop technology extensively to store data and make sense of huge amounts of unstructured data. This data is collected by the various schemes carried out by the Govt. like Bhamashah, RajSampark etc.

Data Analysis:

Data analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making.

Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, while being used in different business, science, and social science domains.

Analysis refers to breaking a whole into its separate components for individual examination. Data analysis is a process for obtaining raw data and converting it into information useful for decision-making by users. Data is collected and analyzed to answer questions, test hypotheses or disprove theories

In statistical applications, data analysis can be divided into descriptive statistics, exploratory data analysis (EDA), and confirmatory data analysis (CDA).EDA focuses on discovering new features in the data while CDA focuses on confirming or falsifying existing hypotheses.

Predictive analytics focuses on application of statistical models for predictive forecasting or classification, while text analytics applies statistical, linguistic, and structural techniques to extract and classify information from textual sources, a species of unstructured data.

Applications of Data Analysis:

- Finance – correlation and regression, index numbers, time series analysis
- Marketing – hypothesis testing, chi-square tests, nonparametric statistics
- Personal – hypothesis testing, chi-square tests, nonparametric tests
- Operating management – hypothesis testing, estimation, analysis of variance, time series analysis

USEFUL LIBRARIES IN PYTHON:

NUMPY:

Introduction

Numpy is a homogenous multidimensional array-processing package which provides a good-performance multidimensional array object, and tools for working with these arrays. It is one of the dominating Python package for numerical computations in Python. Many data mining packages, such as Matplotlib, SciPy, and Pandas are built upon Numpy.

Why it is used?

It is the fundamental package for scientific computing with Python. It contains variety of features including these:

- It is a powerful N-dimensional array object
- It has a much more natural and convenient integration of mathematical operations than lists.
- Sophisticated (broadcasting) functions used in linear algebra, Fourier transform, and random number.
- Possess tools for integrating C/C++ and Fortran code

Besides the above scientific uses, it can also be used as an efficient multi-dimensional container of generic data. Different data-types can be defined using Numpy which allows Numpy to useful in wide variety of databases.

Arrays in Numpy

Numpy's main object is the homogeneous multidimensional array.

- It is a collection of elements in tabular form (usually numbers) of homogeneous type which are indexed by a tuple of positive integers.
- In Numpy dimensions are called *axes* and the number of axes is called *rank*.
- **Ndarray** is its array class which **is** also called the alias **array**.

Importing Numpy

In order to start we need to import Numpy library by using following command:

Import Numpy asnp

Array Creation

There are various ways to create arrays in Numpy:

- One can create an array from a Python **list** or **tuple** using the **array** function. The type of the resulting array is same as that of individual elements.
- Sometimes the size if an array is known but elements of an array are initially unknown. Hence, Numpy offers several functions to create arrays with **initial placeholder content**. These minimize the necessity of growing arrays, an expensive and un-optimized operation. **For example:** np.zeros, np.ones, np.full, np.empty, etc (After importing Numpy as **np**).
- To create sequences of numbers, Numpy provides a function similar to **range** that returns arrays instead of lists:
  np.arange: It returns evenly spaced values within a given interval which returns an array of consecutive elements
  np.linspace: It returns evenly spaced values within a given interval which returns no. of elements.
  Reshaping array: One can use **reshape (np.reshape ())** method to reshape an array. For e.g. let's consider an array with shape of the form (a1, a2, a3… aN). We can reshape and convert it into another array with shape (b1, b2, b3, …, bM). The only required condition is: a1 x a2 x a3 … x aN = b1 x b2 x b3 … x bM . (I.e. original size of array remains unchanged).

Array Indexing

The knowledge of the basics of array indexing is important for analyzing and manipulating the array object. Here are some ways to do this:

- **Slicing:** Similar to lists in python, numpy arrays can be sliced (broken into pieces). Since arrays can be multidimensional, each dimension has to be sliced separately.
- **Integer array indexing:** In this method, lists are passed for indexing for each

dimension of a multi-dimensional array.

- **Boolean array indexing:** This method is used when we want to pick elements from array based upon a condition

Basic operations

A cluster of built-in arithmetic functions are present in Numpy:

- **Operations on single array:** It uses overloaded arithmetic operators to perform element-wise operation on array to create a new array of same size. For e.g. in case of +=, -=, *= operators, the existing array is modified.
- **Unary operators:** Many unary operations are provided as a method of **ndarray** class. For e.g. sum, min, max, etc. It can be applied to both rows and columns (by specifying axis parameter).
- **Binary operators:** These operations apply on array element-wise and a new array is created. One can use all basic arithmetic operators like +, -, *, /. In case of overloaded operators *+=, -=,* =, the existing array is modified.

Universal functions: Numpy provides familiar mathematical functions such as sin, cos, exp, etc. These functions are being applied to every element of the array thus, finally producing array as an output.

PANDAS:

Introduction

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in ***C*** or ***Python***.

We can analyze data in pandas using: **Series** and **Data Frames.** In order to start we need to import pandas library and then both series and DataFrame by using following commands:

import Pandas as pd

from Pandas import Series, DataFrame

Series

Series is one dimensional (1-D) array defined in pandas that can be used to store any data type. Labels need not be unique but must be of immutable type. The object supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index. Statistical methods from **ndarray** have been overridden to automatically exclude missing data (currently represented as **NaN**).

It can be created using following command:

pd.Series (*data=None*, *index=None*, *dtype=None*)

Parameters are:

1. **Data**: It contains the data stored in Series. It can be in form of one or more dictionaries, one or more series or 2D-numpy Ndarray.
2. **Index**: It is in the form of an array or a list. Values must be hashable and have same length as *data.* Non-unique index values are allowed and it will default to its index range (0,1,2….n) if value is not provided. If both a dictionary and index sequence are used, the index will override the keys found in it.
3. **Dtype**: It generates the data-type of the values entered. It is of type Numpy.

Data Frames

It is a two-dimensional structure where data is aligned in tabular fashion comprising rows and columns. Arithmetic operations are applied on both row and column labels. It can be thought of as a combination of Series objects.

Creating a DataFrame

Creation of dataframe is done by passing multiple Series into the DataFrame class using **pd.Series** method. For e.g. consider a DataFrame passed in the two Series objects, s1 as the first row, and s2 as the second row. For e.g. **pd.DataFrame ([s1, s2])**

Importing data with Pandas

The first step is to read the data. In **.csv** file the data is stored as comma-separated values where each row is separated by a new line and each column by a comma (,). In order to be able to work with the data in Python, one need to read the **.csv** file into a Pandas DataFrame where one can represent and work with tabular data.

For Example: **pd.read_csv ("IND_data.csv")**

Indexing DataFrame

Indexing can be possible using the **iloc** method. The iloc method is used for integer-location based indexing / selection by position i.e., selecting rows and columns by number.

Handling Missing Data

The phase of data analysis comprises of handling the missing data from dataset, and then filling it with appropriate values. Not surprisingly Pandas live up to that expectation as well. This can be done via **dropna** and/or **fillna** methods. While dealing with the missing data, one as a Data Analyst is either supposed to drop the column containing the null values (dropna () method) or fill in the missing data with mean or mode or other appropriate value (fillna () method). This decision is of great significance and depends upon the data and the affect would create in our results.

SENTIMENT ANALYSIS:

INTRODUCTION:

Sentiment analysis is the process of determining that whether a piece of text expresses positive, negative or neutral opinion. Because this process derives the attitude or opinion of writer/speaker, it is also known as opinion mining. Sentiment analysis makes use of advances text mining techniques. (Text Mining: is the process of deriving high quality information from text)

Sentiment analysis has many applications and benefits. It can be used by various businesses and organizations to get valuable insights into how customers feel about their product and service.

It can be applied to social media channels to get the sentiment of a given tweet. It can be used to get a heads-up by identifying any negative thread that are emerging online, which would have been difficult to identify without sentiment analysis due to huge amount of data floating on web.

It can also be applied to the emails that a corporate network receives. Sentiments of each mail can be derived and importance can be given to the mail with greater negative sentiment.

IMPORTANCE OF SENTIMENT ANALYSIS:

A company or organization or an individual makes many day to day decisions after going through what the customers or the experts have to say about the matter concerned. For an organization it may be customer review about the product or service given by organization. For an individual it may be the news of stock market or advice of an expert on finances in the form of an article. Day by day, the amount of information on web is increasing drastically and so is the need for a system which can tell sentiment of a given piece or set of information without any human efforts.

DIFFICULTIES IN PERFORMING SENTIMENT ANALYSIS:

Sentiment analysis can be applied to wide range of data but arriving to the conclusion whether a statement is positive or negative can be difficult. The complexity of human language is greatest difficulty in doing sentiment analysis. Opinions can be expressed as sarcasm or irony, which a machine learning based system cannot detect.

Example:

"After a whole 7 hours away from work, I get to go back again, I'm so lucky!"

The above comment is sarcastic and is associated with anger i.e. negative sentiment but machine learning based algorithm will not be able to detect the sarcasm and will interpret the sentiment to be positive.

APPROACHES TO SENTIMENT ANALYSIS:

There are three approaches to Sentiment Analysis:

- **Lexicon-Based Approach:**

In this method, a defined list of positive and negative words with their sentiment score is used. For example - 'nice':+2, 'good':+1, 'terrible':-1.5, etc. Then an algorithm is used to find all the positive and negative words in the text data. For every positive word, increase positive score and for every negative word, increase negative score. The final sentiment score will be positive score – negative score.

- **Machine-Learning Based Approach:**

In this method, a manually sentiments recognized set of data is required. If amount of data is huge, then machine learning techniques can be used to train the system to recognize sentiment automatically without the requirement of pre- defined set of positive and negative words.

- **Combined Approach:**

This approach is a combination of Lexicon-based approach and Machine-Learning Based approach, i.e. both ML algorithms as well as pre-defined set of positive and negative words is used for better results.

➢ We have used Lexicon-Based approach due to following reasons and limitations:

- Machine-Learning based approach requires a large set of data with sentiments recognized manually by people. More the data, better the algorithm will work. This can be done with the help of 'Amazon Mechanical Turk'. MTurk is a crowd sourcing internet market place where individuals or businesses can make use of human intelligence to perform tasks that computers are not able to perform right now, for a small price. This process may take months to get sentiment recognized data. Due to lack of time and funds, doing this was difficult.

- In Lexicon-Based approach, positive and negative sentiment words are readily available.

- There is no guarantee that MTurk sentiment results are correct.

- In ML based approach, we will never know why analyzer works as it works while in Lexicon based approach, we will know exactly why the analyzer works as it works.

LEXICON-BASED APPROACH:

Lexicon based approach is based on the assumption that contextual sentiment orientation is the sum of sentiment orientation of each phrase or each word. Lexicon means the vocabulary of a person, language, or branch of knowledge.
In lexicon based approach, a dictionary of predefined positive and negative words (vocabulary) is used to give sentiment orientation to words of a given text data.
Two sub categories of Lexicon-Based approach are:

i. Basic Lexicon-Based approach:

It is the simplest way of doing sentiment analysis. In this approach, every word of the text data is recognized and stop words are removed. (Stop words are words that don't have any sentiment value associated with them). Then the remaining sets of words are given sentiment score. Each remaining word will either be a positive, neutral or negative sentiment word. This can be done using python by checking the presence of a word in positive or negative lexicon dictionaries that are easily available online. If the word is in positive lexicon dictionary, then we increase sentiment score by 1 and if it is in negative lexicon dictionary, then we decrease sentiment score by 1. After doing this for every word in text data, we will get final sentiment score.

ii. Advanced Lexicon-Based approach:

This approach is an improvement to Basic Lexicon-Based approach. In this approach, the following aspects are considered:

- ✓ **Negations:** Negations are very important linguistics because they affect polarities of other words. Some examples of negation words are: no, not, shouldn't, etc. When a negation appears in a sentence, it is important to determine the word(s) that are affected by this term.

  For example, in sentence "this car is not nice looking but its performance is good", scope of negation is only limited to word next to it whereas in sentence "the battery doesn't work for a long time", the scope of negation is till end.

The part of sentence that negation affects is called the vicinity or scope of negation.

Negations can be divided into three classes:

a. **Syntactic negations:** those negations that completely inverts the polarity of other words.
b. **Diminisher negation:** those negations that reduces polarity of words instead of inverting them.
c. **Morphological negation:** includes all the prefixes and suffixes which can be used to form morphological negation.

- ✓ In simple sentences (i.e. sentences with only one clause) a negation may invert polarities of all the words whereas in compound sentences (in which there are multiple clauses) negation usually inverts polarity of only some of the words.
  Therefore it is little challenging to deal with compound sentences.

- ✓ Conjunctions: conjunctions do not allow the effect of negations to travel to next clause of the sentence. Example 'but', 'whereas', etc.
  Also presence of conjunctions in a sentence usually implies different sentiment scores for two parts of the sentence. For example, consider the sentence "He is not a good person but what he did the other day was really kind". This sentence is tending towards being a positive and if it is scored in normally then it will come out to be a neutral sentence.
  What can be done is to give more weightage to part of sentence after conjunction than the part before the conjunction.

- ✓ Punctuation marks: Punctuation marks such as comma ",", semicolon ";", colon ":", exclamation mark "!", braces "( )", full stop ".", question mark "?" and single and double quotation marks also limit the scope of negation.

- ✓ Exceptions in handling negations: in some cases, negations do not have any scope when used with other words such as "not just", "not only", "no wonder", "not to mention".

DATA COLLECTION:

All the data that need to be collected in order to do advance Lexicon based sentiment analysis are:

- ✓ Syntactic, Diminisher and Morphological negation words
  Source: *Paper "Negation Handling in Sentiment Analysis at Sentence Level" from Journal of Computers (JoC), Volume 12, Number 5, September 2017*
- ✓ Contrasting conjunctions
  Source: *Paper "Negation Handling in Sentiment Analysis at Sentence Level" from Journal of Computers (JoC), Volume 12, Number 5, September 2017*
- ✓ Punctuations
  Source: *Paper "Negation Handling in Sentiment Analysis at Sentence Level" from Journal of Computers (JoC), Volume 12, Number 5, September 2017*
- ✓ Stop Words
  Source: *https://www.ranks.nl/stopwords*
- ✓ Positive Lexicon word's Dictionary
  Source: *http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar*
- ✓ Negative Lexicon word's Dictionary
  Source: *http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar*

STEPS:

I. Newspaper article scraping & curation:

In order to get news articles directly from web, a package 'newspaper' can be used on python. This module scrapes (Data scraping is a technique in which a computer program extracts data from human-readable output) and curates (curation is process of gathering relevant information from scraped data) news articles. Only input required is the URL of a particular news article whose sentiment has to be analyzed.

II. Conversion of main text data of article to lower-case:

For doing sentiment analysis, every word of text has to be identified. By this conversion, words with capital letters in it will not have to be identified

separately.

III. Breaking the main text into sentences:

Next step is to break the main text into sentences that then will be given negation score i.e. negation score tells whether the sentence has negation (negation score =-1) or not (negation score=+1). To split the text into sentences, method ".split (".")" can be used which comes inbuilt in Python.
But before splitting data into sentences, some aspects have to be dealt with:

- **Dealing with Punctuation marks:**

  As was analyzed earlier that punctuations limit the scope of negation i.e. the effect of negation before a punctuation mark does not travel through. One simple solution to this is replacing all the concerned punctuation marks with "." and then split the text into sentences. By doing this sentence with punctuation mark in between will act as two sentences and negation can be accounted for both of then separately.

- **Dealing with Conjunction:**

  As was analyzed earlier that conjunctions limit the scope of negation and amplifies the sentiments. Replace contrasting conjunctions with " x.x". This is done so in order to give more weight age to sentence after the conjunction. After splitting, sentence that ends with "x" will have less relative weight age than sentences that starts with "x".

Now split the text data using ".split ()" and save them to the list Sentences.

IV. Giving Negation score to every sentence present in text:

Firstly define a dictionary (in python) "Sentences1" which stores the sentences and their negation scores. If negation word(s) are present in odd numbers then update the negation score of respective sentence to -1, and if negation word(s) are present in even numbers then update the negation score of respective sentence to 1. If a diminisher word is present in the sentence then multiply the negation score with 0.5.
Also if conjunctions were present in original sentence then 'x' will be present at the

endandbeginningoftwosentenceswhichwereasinglesentenceearlier.

Sentence with x at the end will be given less weightage (i.e. multiply negation score with 0.5) and sentences with x at the beginning will be given more weight age (i.e. multiply negation score with 1.5).

V. Declaring two variable which will store Sentiment Scores of the words:

Declare two variables 'pos' and 'neg'. Variable 'pos' will be updated with every appearance of a positive sentiment word and 'neg' will be updated with every appearance of a negative sentiment word. Initially set both of them to zero.

VI. Using Bag Of Words Technique to get final Sentiment Score:

Follow the following steps for every sentence in dictionary "Sentence1":

- ✓ Declare a dictionary "Bag_of_words" with every word (Including Stop Words) as key and value as a frequency of that word appearing.
- ✓ Declare a dictionary "clean_Bag_of_Words" with words (Excluding Stop Words) as key and value as frequency of that word appearing.
- ✓ Find all the words in text using "Regular Expression (RE)" module which is provided by python, and save them to list "Words_in_Text".
- ✓ Now update dictionary "Bag_of_words" using list "Words_in_Text".
- ✓ Now use dictionary "Bag_of_words" and list of Stop Words to update dictionary "clean_Bag_of_Words".
- ✓ Now by making use of words in dictionary "clean_Bag_of_Words" and their frequency to give sentiment scores to every word and update variables "pos" and "neg". In doing so, following things has to be done:
    - If a given word in dictionary "clean_Bag_of_Words" is in positive lexicon dictionary and the negation score of the sentence is greater than zero, then update pos (pos = pos + negation score*frequency), if negation score is less than zero, then update neg

(neg = neg – negation score*frequency).

- If a given word in dictionary "clean_Bag_of_Words" is in negative lexicon dictionary and the negation score of the sentence is greater than zero, then update neg (neg = neg + negation score*frequency), if negation score is less than zero, then update pos (pos = pos – negation score*frequency).

VII. Calculate final sentiment score using "pos" and "neg":

If: pos = 0 and neg = 0: Sentiment Score = 0;

If: either one or both pos or neg not equal to 0;

SentimentScore=(pos–neg)/(pos+neg)

CONCLUSIONS

Before starting work on the project, it was necessary to gain familiarity with various data analysis methods and techniques. The same was gained through material and guidance provided by the manager and employees.

Doing Sentiment analysis of Text Data is difficult than analyzing numerical data. Various libraries like Pandas and Regular Expression really helped a lot in Python. The sentiment analysis model that we built is an advanced form of Lexicon Based Model. There can be many improvements that can be made to the model improvising every time testing new Articles. We improvised and arrived to present model by going through about 100 tests. There is even more scope for improvising.

Sentiment analysis can be applied to more specific then generalized fields like finance or crime. Analysis models built for specific field will give more accurate results than generalized model. We would try and work on them in our coming years in college.

CODE SNIPPET

```
import pandas as pd import
nltk
from newspaper import Article import re

# Reading the article using the URL.... print('')
url = input("Enter URL:")        # Prompts Enter URL: on the screen article = Article(url)    # save
article on url in variable 'article' article.download()       # Downloads article
article.parse()                  # Reads the article

Text = article.text.lower()                          # Converting text of the article to lower case

print("\n-------------------------------------------------------------
---------------\n")
print("\nArticle's Title:\n" + article.title) print("\nArticle's Text:\n"
+ Text)
print("\n-------------------------------------------------------------
---------------\n")
Syntactic = pd.Series(
        ["needs a", "needs to", "could have", "could've", "should have",
"should've", "would have", "would've", '-less ',
          'no ', 'not ', 'rather ', "couldnâ€™t ", "wasnâ€™t ", "didnâ€™t ", "wouldnâ€™t ",
"shouldnâ€™t ", "werenâ€™t ", "donâ€™t ",
          "doesnâ€™t ", "havenâ€™t ", "hasnâ€™t ", "wonâ€™t ", 'wont ', "hadnâ€™t ", 'never ',
'none ', 'nobody ', 'nothing ',
          'neither ', 'nor ', 'nowhere ', "isnâ€™t ", "canâ€™t ", 'cannot ', "mustnâ€™t ", "mightnâ€™t ",
"shanâ€™t ", 'without ',
          "neednâ€™t ", 'de-', 'dis-', 'il-', 'im-', 'in-', 'ir-', 'mis-', 'non-', 'un-'])

Negation_Exceptions = pd.Series(["not just", "not only", "no wonder", "not to mention"])

Diminishers = pd.Series(['hardly', 'less', 'little', 'rarely', 'scarcely', 'seldom'])

Contrasting_conj = pd.Series(
        ['but', 'however', 'in contrast', 'instead', 'on the other hand', 'whereas', 'except that', 'on the
contrary',
          'conversely', 'nevertheless', 'although', 'alternatively']) Punctuations = pd.Series([';', ':',

'!', '?', ','])

stop_words = ["a", "about", "above", "after", "again", "against",
"all", "am", "an", "and", "any", "are", "as", "at",
```

```
                        "be", "because", "been", "before", "being", "below",
"between", "both", "but", "by", "could", "did", "do",
                        "does", "doing", "down", "during", "each", "few", "for",
"from", "further", "had", "has", "have",
                        "having", "he", "he'd", "he'll", "he's", "her", "here", "here's", "hers", "herself",
"him", "himself",
                        "his", "how", "how's", "i", "i'd", "i'll", "i'm",
"i've", "if", "in", "into", "is", "it", "it's", "its",
                        "itself", "let's", "me", "more", "most", "my", "myself",
"nor", "of", "on", "once", "only", "or", "other",
                        "ought", "our", "ours", "ourselves", "out", "over",
"own", "same", "she", "she'd", "she'll", "she's",
                        "should", "so", "some", "such", "than", "that",
"that's", "the", "their", "theirs", "them", "themselves",
                        "then", "there", "there's", "these", "they", "they'd",
"they'll", "they're", "they've", "this", "those",
                        "through", "to", "too", "under", "until", "up", "very",
"was", "we", "we'd", "we'll", "we're", "we've",
                        "were", "what", "what's", "when", "when's", "where",
"where's", "which", "while", "who", "who's", "whom",
                        "why", "why's", "with", "would", "you", "you'd",
"you'll", "you're", "you've", "your", "yours",
                        "yourself", "yourselves"]

# Detecting conjunctions and punctuations in main text of the Article for w in Contrasting_conj:
        if w in Text:
                Text = Text.replace(w, ' b.b')

for w in Punctuations: if w in
        Text:
                Text = Text.replace(w, '.')

# Splitting the main text of article into sentences Sentences = Text.split('.')

# Reading positive and negative Lexicon Dictionaries from a location on pc
positive_words = open('/home/aneesh/Documents/Project Finale/positive- words.txt', 'r').read().split()
negative_words = open('/home/aneesh/Documents/Project Finale/negative- words.txt',
'r').read().split()

# Declaring positive and negative sentiment score variables pos = 0
neg = 0

# Dictionary with Sentences and their Negation score(-1 if negation is there and 1 if negation is not
there)
Sentences1 = {"Sentence": "Negation Score"} # List of Positive

and Negative words found
```

```
positive_words_found = []
negative_words_found = []

# Updating Dictionay "Sentences1" with respective sentences negation score
for s in Sentences: Sentences1.update({s:
        1}) for w in Syntactic:
                if w in s:              # If syntactic negation is present
                        if Sentences1[s] == -1:                     # If sentence has -1 negation score
already, i.e. odd number of negation words has occured
                                Sentences1[s] = 1 else:
                                Sentences1[s] = -1
        for x in Negation_Exceptions:                       # Taking care of negation exceptions
                if x in s:
                        if Sentences1[s] == 1:
                                Sentences1[s] = -1 else:
                                Sentences1[s] = 1
        for w in Diminishers:                   # Taking care of Diminishers if w in s:
                        Sentences1[s] *= 0.5
        if s[0:2] == 'b ':                  # Taking care of Conjunctions in Sentences Sentences1[s] *=
                1.5
        if s[-2:] == ' b':
                Sentences1[s] *= 0.5

# Getiing Sentiment score using Bag_of_words Technique for s in Sentences1:
        Bag_of_words = {"Word": "Frequency"}
        clean_Bag_of_Words = {"Word": "Frequency"} Words_in_Text
        = re.findall(r'\w+', s)
        for w in Words_in_Text: if w in
                Bag_of_words:
                        Bag_of_words[w] += 1 else:
                        Bag_of_words[w] = 1 for w
        in Bag_of_words:
                if w not in stop_words: clean_Bag_of_Words.update({w:
                        Bag_of_words[w]})
        for w in clean_Bag_of_Words: if w in
                positive_words:
                        if Sentences1[s] >= 0:
                                pos += Sentences1[s] * clean_Bag_of_Words[w]
                                positive_words_found.append(w)
                        else:
                                neg += -Sentences1[s] * clean_Bag_of_Words[w]
                                negative_words_found.append("negation of " + w)
                elif w in negative_words:
```

```
                    if Sentences1[s] >= 0:
                        neg += Sentences1[s] * clean_Bag_of_Words[w]
                        negative_words_found.append(w)
                    else:
                        pos += -Sentences1[s] * clean_Bag_of_Words[w]
                        positive_words_found.append("negation of " + w)

if pos != 0 or neg != 0:
    Sentiment_Score = (pos - neg) / (pos + neg) else:
    Sentiment_Score = 0
positive_words_found = pd.DataFrame(positive_words_found, columns=["Words"])
negative_words_found = pd.DataFrame(negative_words_found, columns=['Words'])
print("\nPositive Sentiments/Words found:\n")
print(positive_words_found)
print("\n------------------------------------------------------------
----------\n")
print("\nNegative Sentiments/Words found:\n")
print(negative_words_found)
print("\n------------------------------------------------------------
----------\n")
print("Positive sentiment score: ") print(pos)
print("Negative sentiment score: ") print(neg)
print("\n------------------------------------------------------------
----------\n")
print("Overall Sentiment Score: ")
print(Sentiment_Score)
print("\n------------------------------------------------------------
----------\n")
```