Twitter Sentiment Analysis of the 2019 Indian Election

by

Jaydeep Motisariya

A thesis submitted in partial fulfillment
of the requirements for the degree of
MSc Computational Sciences

The Faculty of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	Twitter Sentiment Analysis of the 2019 Indian Election	
Name of Candidate Nom du candidat	Motisariya, Jaydeep	
Degree Diplôme	Master of Science	
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance August 31, 2020

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kalpdrum Passi
(Supervisor/Directeur(trice) de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Ramesh Subramanian
(Committee member/Membre du comité)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. David Lesbarrères
Monsieur David Lesbarrères
Dean, Faculty of Graduate Studies
Doyen, Faculté des études supérieures

Dr. Nilesh Modi
(External Examiner/Examinateur externe)

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

With easily available internet services in India and the rest of the world in the recent past, there is more and more traction towards social media like Facebook, Twitter, Instagram, YouTube, etc. This has enabled individuals the freedom of speech and to display their sentiments and emotions towards society. Social media has brought people closer than ever before and has provided a common platform for individuals to communicate. Some influencers promote products on social media platforms, while politicians run their campaigns online for broader reach. Social media has become the fuel for globalization. In 2019, the Indian Lok Sabha Elections saw around 360 million tweets on Twitter, giving their opinions and showing their sentiment towards the political leaders and their parties. Sentiment analysis is the computational investigation of opinions, evaluations, views, and feelings expressed in a text. The political parties have used this technique to run their campaigns and understand the opinions of the public. This also enables them to modify their campaigns accordingly. In this research text mining was performed on approximately 200,000 thousand tweets collected over four months that referenced four national political parties in India during the campaigning period for the Lok Sabha elections in 2019. The sentiments of Twitter users were identified towards each of the considered Indian political parties, Congress, Bhartiya Janata Party (BJP), Aam Aadmi Party (AAP) and Bahujan Samaj Party (BSP) using VADER (Valence Aware Dictionary and sEntiment Reasoner). A lexicon and rule-based sentiment analysis engine was created that is the principal platform to evaluate the opinions expressed in social media. The results of the analysis show that Bhartiya Janata Party (BJP) being a lead runner in the elections of 2019 received more positive intent and emotions towards their campaigns and their leader Narendra Modi as compared to the other parties and their leaders.

Keywords:

Sentiment Analysis, Twitter, Indian Election 2019, Text Mining, Data Mining, Lexical Analysis, Positive Polarity, Negative Polarity, Tweets, Word Tokenization, Word Cloud.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my thesis supervisor Dr. Kalpdrum Passi for the constant support of my master's study and research, for his patience, motivation, and extensive knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a more desirable advisor and mentor for my master's study.

I would also like to thank my committee members for reviewing this thesis and serving on the defense committee.

Last but not the least, I would like to thank my loving and caring parents, brother, my dear partner and all my friends for presenting me with constant support and continuous encouragement throughout my years of study and researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

DEDICATIONS

I would like to dedicate my thesis

to my beloved grandparents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

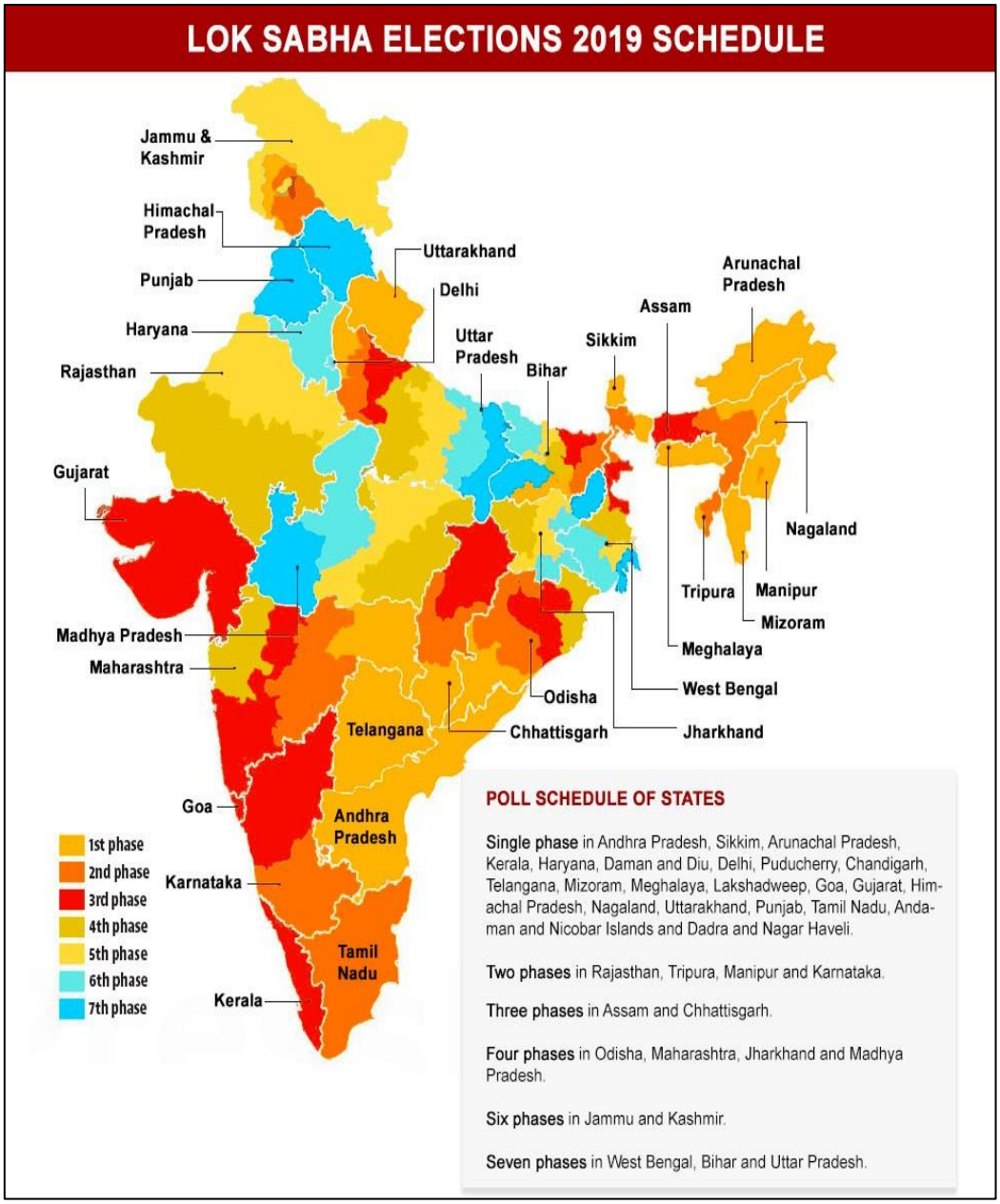
ABBREVIATIONS

NLP	Natural Language Processing
NLTK	Natural Language ToolKit
IDE	Integrated Development Environment
CPU	Central Processing Unit
BJP	Bhartiya Janta Party
INC	Congress
AAP	Aam Aadmi Party
BSP	Bahujan Samajwadi Party
API	Application Programming Interface

Chapter 1

Introduction

India is the world's largest democracy, during the year 2019, India witnessed 17th Loksabha election. The general election is the most significant event every five years to form the government at the center and elect the Prime Minister of India. The election was held for 543 parliament seats [1] all over India. A party requires to win 272 seats for the majority to be elected at the center. As the Election Commission of India, 900 million [1] people were qualified to vote, with 84.3 million voters since the last election in 2014. The 2019 general elections held the largest-ever election in the world. The Election Commission of India conducted the election in a total of 7 phases over seven weeks starting from 11th April 2019 with 1st Phase and 7th Phase commencing on 19th May 2019, with results to be declared on 23rd May 2019. A total of 600 million [1] voters polled their votes in 2019, which was the highest recorded turnout in the history of Indian general elections. In the 2019 election more than 650 parties contested, most of them were small and were targeting a specific region. The major parties were Bhartiya Janata Party (BJP) and the Indian National Congress (INC) or commonly known as congress. There were also a couple of other emerging parties like the Aam Aadmi Party (AAP), Trinamool Congress (TMC), and Bahujan Samaj Party (BSP), which gained traction in some of the states in 2019 elections. Similarly, candidates who had the highest amount of popularity and remained in the news over the election and were also the Prime Ministerial candidates of the parties mentioned earlier, like Narendra Modi from the BJP, Arvind Kejriwal from AAP, Rahul Gandhi from Congress, Mamata Banerjee from TMC, and Mayawati from BSP.



Source 2nglish.newnationtv.com

Figure 1.1: 7 Phases, where elections held for 543 parliament seats in 2019

1.1. Influence of Social Media on Elections

The growth in active online users over social media platforms has seen steep growth over the last decade. As per the world bank [2] as of 2017, around 35% of Indian population are active on the Internet. This has jumped from 7% in 2010, reporting almost 70% year-on-year growth. India has a population of 1.35 billion [3], and 35% of the population approximates to 500 million active Internet users. Twitter sees nearly 500 million tweets per day and more than 6000 tweets per second these are just the number of one social media platform, there are other social media platforms like Facebook and Instagram which has even more number of active members per day. The people are using these platforms to express their views on different topics like business products, events, marketing campaigns, etc.

It is widely pursued that Donald Trump, in 2016 US Presidential Election with the help of marketing agencies like Cambridge Analytics, used social media platforms like Facebook and Twitter to demographically target its active users by running specialized campaigns and advertisements to influence them to vote for him. Similarly, in India, in the 2019 Lok Sabha elections, major parties have tried to make their presence felt on the social media platforms with their official pages and of their leaders and the parties. During campaigning, many parties used YouTube to stream their live campaign videos for broader reach. Furthermore, campaign posters and advertisements were posted on Twitter and Facebook for increasing traction all over the world and not just India.

1.2. Motivation

With all the social media platforms, Twitter is well received by journalists, politicians, etc. for its potential political influence on its users. Twitter makes users express themselves in the limit of 140 characters, although this seems to be a limitation, looking on the shiny side, it rather enables users to precisely express them. Moreover, twitter has a provision of marking profiles of the large following verified. Therefore, this helps users to filter out tweets coming from unverified accounts of similar twitter handles. Furthermore, features like retweeting the original tweet allow for more extensive and quicker reach among its users. Such features make twitter the best choice of platform for running campaigns by the political parties and their leaders.

The 2019 Indian Lok Sabha election is the most significant democratic event every five years, and with the rise in Internet users on social media platforms, it is one of the generating sources of Big Data. Similarly, is the case of other elections in countries like the USA and UK, and in literature, several studies have been carried to analyze the public emotions and opinions of the battles fought over social media platforms like Twitter. There are more than 50 million users in the United States of America and use English as their medium to communicate through Twitter and express their emotions and opinions. India is a diverse country, and there are more than 100 languages spoken by the people, and Twitter provides support for writing the tweets in multiple languages like English, Hindi, Gujarati, etc. With more than 650 parties contesting the election, it gives a broader proposition to the people of India to express their opinions about multiple parties; therefore, the problem of understanding their inclination towards a party becomes enlarged and exciting. It was reported that the season of 2019 Indian Loksabha Election witnessed around 350 Million tweets. All these factors, along with such a more extensive base of twitter data it motivated us to conduct this study for finding interesting patterns.

1.3. Objectives

With the historical data for four months from February 2019 to 23rd May 2019, the thesis tries to relate the public opinion and emotions with the parties campaigning timelines and find out if there were any hidden patterns. The more extensive objective of this work is to find out the political orientation of the users of twitter towards the leading parties. The thesis focuses on the user tweets targeted to leading parties like BJP, INC (Congress), AAP, and BSP.

Sentiment analysis can be described as the process of extracting the emotions from the user's written text by processing unstructured information and preparing a model to extract the knowledge from it. In this thesis, the rule-based lexicon approach is implemented, which predicts the sentiments of the text by using the corpus like SentiWordNet and WordNet. The process is to obtain the text's polarity based on a set of words, each of which is annotated with the weight and extracts the information that contributes to the overall sentiment of the text. There are many approaches used for sentiment analysis on linguistic data, and the purpose to be used depends on the essence of the data and the platform selected for the purpose. The thesis aims to use Valence Aware Dictionary and Sentiment Reasoner [9], commonly known as VADER, an advanced lexicon rule-based sentiment analyzer which is specialized in analyzing social media texts.

Furthermore, the thesis aims to conduct exploratory data analysis and use data visualization techniques for analyzing tweet volume daily, weekly, and monthly for each party and its leader mentioned above. This will also help us understand the popularity of the parties and their leaders during their campaigning.

1.4. Methodology

The proposed flow for twitter sentiment analysis is shown in Figure 1.2. The system first extracts the tweets from the source, performs pre-processing and filtering based on the keywords, and then uses a lexicon-based VADER [9] sentiment analyzer to classify the tweets in positive, negative, and neutral.

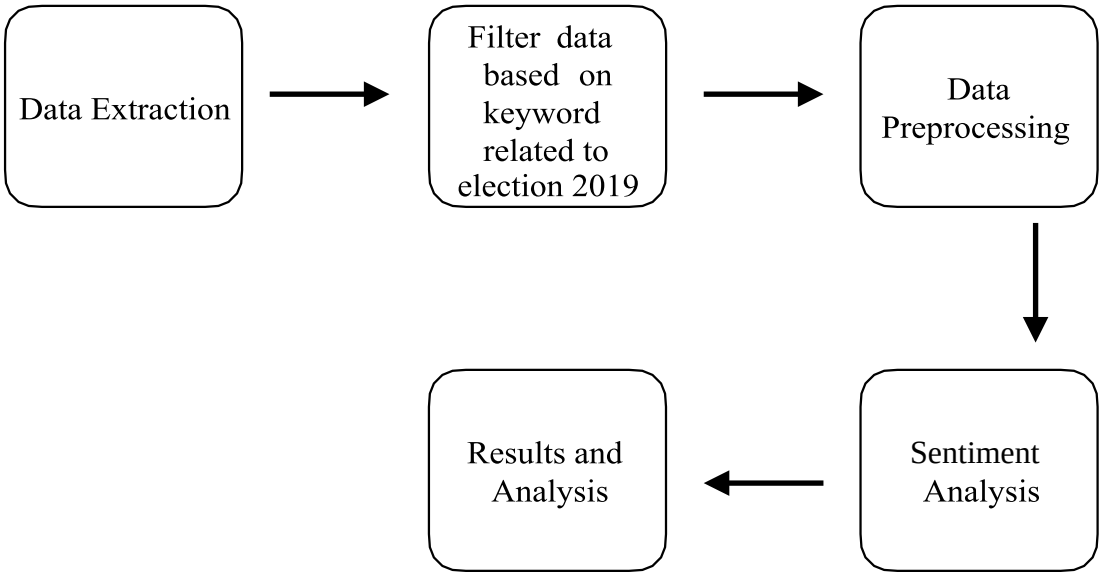


Figure 1.2: Flow diagram for twitter sentiment analysis

As per the above-presented approach, the thesis in the next section outlines the steps taken to implement the Twitter Sentiment Analysis for the 2019 Indian Lok Sabha Election.

1.5. Outline

Presented below are the steps performed while analyzing the sentiments of the user tweets for 2019 Indian Loksabha Election:

Chapter 2: Literature Review

It discusses the approaches used by other research to implement sentiment analysis on user tweets.

Chapter 3: Data Extraction & Data Preprocessing

It explains the data extraction techniques used to extract the dataset, which can be further used to perform NLP and sentiment analysis and The Data pre-processing steps for reducing the noise from the dataset as well as filtering the dataset based on the keywords. It ensures the readiness of the data for sentiment analysis.

Chapter 4: Sentiment Analysis

It uncovers the use of the VADER sentiment analyzer and NLP concepts to calculate the polarity of user tweets.

Chapter 5: Result & Discussion

It focuses on performing exploratory data analysis and visualization for analyzing tweet volume and its sentiment daily, weekly, and monthly for each party and its leader based on the campaigning strategies. Further results are discussed to compare the user opinions and sentiment for each party and its leader.

Chapter 6: Conclusion and Future Work

This section summaries the results and conclude the finding and proposes future work for performing sentiment analysis on user tweets.

Chapter 2

Literature Review

Sentiment analysis of tweets data is considered a much harder problem than standard text, i.e., review documents. This problem is slightly due to the frequent use of irregular and informal words, the short length of tweets, and the rapid evolution of language on Twitter. In Twitter sentiment analysis, a significant amount of work is carried out, followed by the feature-based approaches.

2.1. Related to work

In this section, we discuss related works about predicting and analyzing the result of an election using Twitter. We mark that researchers use a different way technique for sentiment analysis. A few researchers try to find the political preference of a user, and then compare the preferences in election, while others used judged tweets related to previous elections to predict the result of the future elections and the opinions of vote choice of the users.

Previous studies show that analyzing these sentiments and patterns will generate effective results that may be handy in determining the opinions of the public on elections and policies of the government. In [4], authors extract sentiments (positive, negative), additionally as emotions (anger, sadness, etc.) concerning the many leading party candidates, and calculate a distance measure. The distance measure shows the proximity of the political parties, with closer proximity, higher the possibilities of close political connections between those parties.

In [5] and [6], the authors additionally, use twitter data to predict election polls and derive valuable data regarding public opinions.

To conclude the sentiment of a word, four procedures were discussed by Das and Bandopadhyay [7]. The first approach for defining the sentiment was an interactive game that was proposed that explains the words with their corresponding polarity. In the second approach, a bilingual dictionary of English and Indian languages was used to provide the polarity. In the third approach, WordNet was used to indicate the polarities. In the fourth approach, they decided the polarity, using pre-annotated corpora. Das and Bandopadhyay [8] recognized given expressions in the Bengali corpus. They categorized the words in six feeling classes ((anger, disgust, fear, happy, sad and surprise) with three classes of intensities (high, general, and low) to perform sentence-level annotation.

Analysts have obtained a rule-based model for sentiment analysis called VADER [9]. They discovered that with a sentiment lexicon and several syntax rules, their model could exceed both individual human raters and machine learning methods. The VADER model resulted in an open-source application. A transposed version of the VADER application and lexicon was used in this study.

The ideas that people express in social media are also related to having an impact on people’s choices of political parties [10]. Some investigations have proved to predict political election results with sentiment analysis of tweets: using the lexicon method for the Swedish elections [11]. In the study concerning the Brazilian elections, Oliveira et al. [12] examined whether sentiment analysis of data from Twitter could appraise the citizen's political decisions, as public sentiment polls do. Their results were positive.

The Dictionary-based approach has the primary strategy to collect a small set of opinion words manually and then grow it by searching in an extensive collection of texts such as WordNet [13][14]. The new words are then added to the first set of opinion words, and the sequence is returned until there are no more words remaining to be found. The most significant downside of this method is that it relies solely on corpora, and we will not always have an extensive set of opinion words with a domain available. It is essential to consider that not all the words in a lexicon represent a positive or negative opinion regarding an entity. The Corpus-based procedure is mainly used in two situations: to discover new sentiment words from a domain corpus using a given list of known opinion words and build a sentiment lexicon from another [11]. This approach is not as practical as the dictionary-based approach because it would require a corpus with all the English words [15]. The corpus-based approach is distributed in the statically and the semantic path depending on the technique used.

Lexicon-based approach, using lexicon-based and then a chi-square analysis to classify new tweets. Our work uses a lexicon-based approach combined with a rule-based approach. There are other related approaches but using manually specified sets. The method presented by Barbosa et al. [16] classifies in subjective and objective terms, then abandons the actual tweets and organizes the first group as positive or negative.

Jose & Chooralil [17] added a new sentiment analysis method. The data collection process was implemented with the help of Twitter's Streaming API. They used a new technique for analyzing tweets, and with the help of lexical resources such as WordNet, SentiWordNet, and word sense disambiguation, they examined to extract information and facts out of tweets. Also, in order to obtain the highest efficiency possible, they also recommended a negation

handling method in the pre-processing data stage. The author's innovative vision leads them to try a variety of tools.

Tumasjan et al. [4] examined the influence of Twitter and tweet sentiment in the 2009 German national parliament election. The critics received 104,003 tweets in the weeks leading up to the election. The LIWC2007 (Linguistic Inquiry and Word Count) software used to extract sentiment from tweets. Authors observed that political debates are usually driven by a small number of heavy users (80+ tweets). This study also noted that the quantity of tweets reflects the outcome of the elections.

Taboada et al. [18] introduced a lexicon-based method to sentiment analysis. The author used dictionaries of positive or negative polarized words to do analysis tasks. A semantic orientation calculator was developed based on these dictionaries by consolidating intensifiers and negation words. The authors [19] reviewed different methods used to analyze a given piece of natural language text according to the sentiments expressed in it, i.e., whether the overall attitude is negative or positive. They performed a collection of methods for features classification and polarity identification of product reviews using machine learning, namely Support Vector Machine (SVM) combined with domain-specific lexicons.

The utilization of Twitter by politicians and their campaigns is an essential subject of study. Usage of Twitter, while the campaign cycle of 2008 in the USA by Barack Obama, produced an interest in understanding Twitter's role in political battles [20][21]. The related analysis was also conducted in examining the Twitter venture of US Congress members through their election campaigns. Investigations explained that congress members frequently posted

information on Twitter about their political opinions on several issues and issues linking with their constituencies [22][23].

A study in sentiment analysis commenced in the early 2000s. Since then, several techniques to analyze the opinions and sentiments from online opinion sources (blogs, discussions, or business websites) have been added. Nowadays, a concern has started on social networks such as Twitter, where people share their ideas about several points [24][25]. Most of those efforts are based on two main approaches, The lexical-based approach, and the machine learning approach.

Sharmistha Chatterjee [26] performed sentiment analysis on two parties BJP and Congress, using crawling twitter data through APIs. They have tried to predict different moods on parties using moods, applying standard machine learning algorithms and deep learning to do multi-class mood classification for two prominent parties in the election. They crawl tweets every week and merge them with previous weeks to have an overall prediction over a few months. They also used Sentiment Representation by WordCloud and N-gram Model. They also performed Location-wise tweet distribution and Retweet Frequency Distribution.

Parul Sharma and Teng Sheng Moh [27] tried to predict Indian election results using Sentiment Analysis on Hindi Twitter. They fetched 42,345 raw tweets in Hindi language using hashtags from Twitter Archiver. After performing the preprocessing step, 36,465 tweets were left. They labelled manually on 36,465 tweets. The author used a supervised approach such as classification algorithms, Naive Bayes, Support Vector Machine (SVM), and unsupervised approach such as Dictionary-based.

Dipak Gaikar and Ganesh Sapare [28] obtained prediction on Loksabha Elections 2019 results in India using LSTM Neural Network. They first created LSTM (Long short-term memory) classifier, a modified version of RNN (Recurrent Neural Network), for analyzing opinions about different election candidates expressed in the tweets. The training model used over 1500 labelled tweets, which are labelled as positive, negative, and neutral. They extracted a total of 40,000 real-time tweets using Twitter APIs from Jan 2019 to Mar 2019 related to names of Indian political parties. They also used word cloud for data visualization and compared results with online news channel survey ABP-C and India Today Survey for Lokshabbha Election 2019.

Twitter-based election prediction and analysis conducted by Salunkhe & Deshmukh [29]. The data gathering step is the initial phase in the research, so they collected data for the US elections and Gujarat Rajya Sabha elections through Twitter API and classified the polarity of tweets. They were classified into three categories positive, negative, and neutral. They used a dictionary-based approach for prediction using eleven feelings. Moreover, they classified tweets in eleven various feelings, namely sadness, tentativeness, certainty, achievement, anxiety, work, anger, positive word, negative words, negative hashtag, and positive hashtag. An analysis of sentiment analysis in Gujarat elections used six variables for the Gujrat Rajya Sabha election. It was noted that all the Candidates had more tweets showing joy than any other emotion.

In this thesis, we have used sentiment analysis techniques to uncover the opinions, sentiments, and emotions of user tweets for respective parties and candidates. The dataset was constructed from the tweets related to both the candidates and the parties, which was challenging to filter and process. We propose a rule-based lexicon approach that uses

WordNet's dictionary to calculate the polarity of the tweet (positive, negative, and neutral). However, just using simple WordNet or Corpus-based approach may not give accurate results when working with social media text. To overcome that, we have used VADER [5] (Valence Aware Dictionary and sentiments Reasoner), which is specifically tuned to sentiments shown in social media. VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is [30].

Chapter 3

Data Extraction & Preprocessing

In this chapter, the focus is to uncover the steps used to extract the data from twitter and data preprocessing. The thesis discusses the use of twitter API, its features, and its limitation. It also proposes ways to extract historical data for user tweets. The chapter unfolds with more details about twitter as a social media platform, the characteristic of twitter data, what features of the tweets were extracted from Twitter, how to filter the user tweets based, about python, data preprocessing methodology, data preprocessing algorithm and preprocessing steps.

3.1. About Twitter

In the year 2006, Jack Dorsey founded Twitter. Twitter's origins lie in a board meeting of the podcasting company Odeo. An undergraduate student, Jack Dorsey from New York University, to communicate within a small group, he advised the thought of an individual using an SMS service. The essential idea was to develop an SMS-based communication platform, where a group of people create their accounts, update the status, and can text using the platform. Initially, it was named "twttr," when Jack Dorsey and his contractor developed a prototype that was used as a domestic service for Odeo employees. Jack Dorsey, and other members of Odeo in October 2006, formed a new company Obvious Corporation and obtained Odeo, unitedly with its assets and investment from venture investor and shareholders formed Twitter.com.

Twitter today has become the most popular social media platform. It has more than 320 Million active monthly users across the globe. Twitter works as a platform where users can easily express their opinions, present their feelings, and pronounce beliefs and emotions (Hemalatha et al. 2012) [31].

3.2. Characteristics of Twitter Data

At heart, twitter was built as an SMS platform to send the message across people in a small group. It was developed in such a way that users, while presenting their idea, should do it concisely and effectively. Therefore, today twitter has a restriction in the number of characters that a user can write within a tweet. A user can type in maximum of 140 characters in the tweet which he/she wants to post. Although some technology enthusiasts take this as a negative implication as a user may not be able to convey all the information in one tweet, instead of looking at the bright side, it forces the user to communicate one's thoughts effectively and sentiment (Hemalatha et al. 2012) [31]. The Figure 3.1 describes the characteristics of the Twitter data.

Figure 3.1: Characteristics of the Twitter data

A user tweet posted on twitter mostly contains a set of words that may include a hashtag, which is the most important and meaningful symbol in a user tweet. Hashtags start with "#," this helps twitter to find out the most trending keyword over the globe. For example, "#NarendraModi" on twitter will put through all the current trending posts, news, videos, and photos related to the hashtag searched. The symbol "#" is the primary character in the twitter language, which can be used to know what is happening around the globe.

Another important symbol in a user tweet is "@, ", a user in it uses this symbol tag or mention a user or an account on twitter. For example, "@narendramodi" in user tweet, is a way to call out or mention the "narendramodi, ", the Prime Minister of India.

Moreover, a user tweet contains metrics that show the number of likes, number of retweets, number of dislikes, number of comments, and date-time at which the user posted the tweet. Here, in this thesis, the next part of the user tweet, which contains users' opinions and sentiments have been extracted to perform sentiment analysis using a lexicon-based approach.

3.3. Extracting Twitter Data

Twitter provides an API for developers to integrate it with their existing applications. Developers use Twitter API to extract the data of user tweets to generate insights. Twitter has both the REST API and Streaming API for data extraction. However, there are limitations to it; using API, you cannot fetch the data above 90 days from the current date, which means if a developer wants to fetch historical data of last year, it is not possible with Twitter API.

Alternatives, like Twitter crawlers, are available as browser extensions to fetch the data for specific keywords and hashtags, but using crawlers, it is challenging to get the data for keywords you are looking for. Other online archiving websites collect user tweets for all languages available on twitter as well as user tweets for the keywords, which means they do not filter the incoming twitter stream data.

In this thesis, we used one such twitter archiver [32] source to extract the historical data for the relevant keywords from 1st February 2019 to 23rd May 2019. This is the time wherein most of the events took place for the 2019 Lok Sabha Elections. The data was downloaded as a compressed ".tar" file for each day of the month. The Figure 3.2 describes the folder structure for each uncompressed ".tar" file.

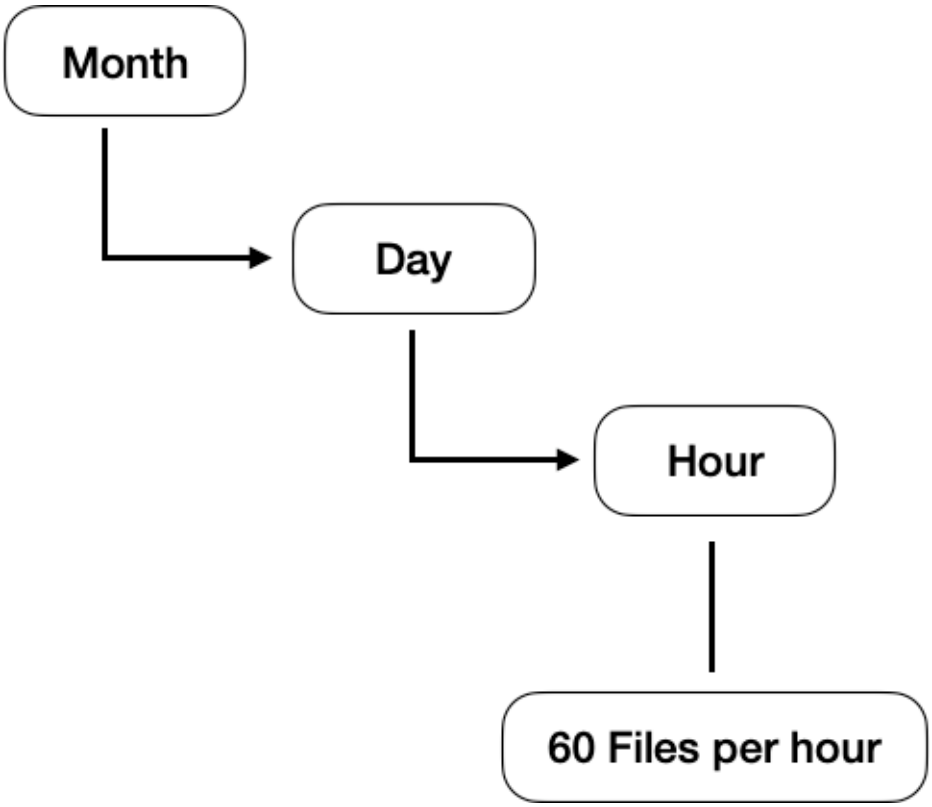


Figure 3.2: Represents the hierarchy in which data is stored in the local system.

The dataset downloaded from the source was huge since it contains the user tweets from over the world for the mentioned period; the average size of data per month was approximately equal to 50GB. Also, there were challenges to extract data from the JSON file, which is a semi-structured data file format, also popularly known as data interchange file format.

Each JSON object for the user tweet provided information such as tweet_id, tweet_text, created_at, tweet_locale, user_screen name, hashtags, location, profile image URL, and followers count, friends count, status count and various additional useful parts of information about the tweet and user profile. Python programming language was used to parse out the specific keys from the JSON objects. The Table 3.1 shows the keys/columns extracted from each JSON object.

Table 3.1: Keys/columns extracted from each JSON object.

Columns	Data Types
Tweet ID	Integer
Tweet Text	String
Tweet locale	String
Created at	Timestamp
User Screen Name	String
User Mention Screen Name	String
Hashtags	String
User Location	String
Quote Count	Integer
Reply Count	Integer
Favorite Count	Integer
Retweet Count	Integer

Tweet IDs are unique 64-bit unsigned integers, which are based on time, instead of being sequential. Tweet Text is a status update consisting of 140 characters or less on Twitter.

“Created at” indicates a UTC DateTime that the user account was created on Twitter. “User Screen Name” is typically a maximum of 15 characters long, but some historical accounts may exist with longer names. “User Mention Screen name” indicate a Twitter username of User. “Hashtags” shows number of hashtags a user posted in user tweets. “User location” defines location for this account’s profile. But it is not necessary to put in the tweets. So, in our dataset only 30% user tweets show the correct location, and the remaining tweets do not show the correct location and some have a null value. For example, a user belonging to Surat, Gujarat may enter India as the account location, so the exact location of User cannot be determined. “Quote Count” indicates approximately how many times users have quoted tweets. “Reply Count” shows the number of times this tweet has been replied. The “Favorite count” shows approximately how many times users have liked this Tweet. “Retweet Count” shows the number of times this tweet has been retweeted.

3.4. Dataset and Variables

The Twitter Data extracted from the source for the period, 1st February 2019 to 23rd May 2019, contains user tweets from all over the world. To keep track of the user tweets related to the 2019 Indian Lok Sabha Election, keywords were handpicked to filter out user tweets and store tweets related to the general election. An exhaustive list of keywords was prepared, which contains words that would help us extract both positive, neutral, and negative sentiment for each party and its leader. Below is the list of keywords used to extract user tweets related to each party and its leader.

For BJP and Narendra Modi -

[“IndiaBoleModiDobara”, “ModiAaRahaHai”, “ModiAaGaya”, “ekbaarfirmodisarkar”, “voteformodi”, “firsemodi2019”, “ModiMatBanao”, “MainBhiChowkidar”, “modichorhai”, “ChowkidarChorHai”, “gobacksadistmodi”, “modihataodeshbachao”, “bekarchowkidar”, “ModiReturns”, “AayegaToModiHi”, “armysemaafimangomodi”, “modihiNYAYhai”,

“BJP4India”, “narendramodi”]

For INC/Congress and Rahul Gandhi -

["Pappu", "ShameOnCongress", "CongressGayi", "rahulgandhichorhai", "rahulgandhi", "INCIndia", "soniagandhi"]

ForAAP and Arvind Kejriwal -

[“AamAadmiParty”, “Kejriwal”, “ArvindKejriwal”, “AAP”, “mufflerman”]

For BSP and Mayawati -

[“bspchiefmayawati”, “Mayawati4PM2019”, “NextPMBehenMayawatiji”, “BSP”]

Also, there were other common lists of keywords which would allow filtering the user tweets just specific to 2019 Indian Loksabha election, below is the list –
["ElectionResults2019", "Verdict2019", "LokSabhaElections2019", "votedforbetterindia", "Indianelections2019", "Votekar", "everyvotematters", "VijayiBharat"]

After filtering the user tweets based on the above exhaustive list of keywords for each party and its leader, the raw dataset contains approximately 200,000 tweets and is stored as a CSV file for further preprocessing. Below Table 3.2 and Table 3.3 show the distribution of tweets by the party and its candidate.

Table 3.2: Distribution of Tweets by Party

Party	Number of Tweets
BJP	1,54,667
Congress	33,260
AAP	8,497
BSP	185

Table 3.3: Distribution of Tweets by Candidate

Candidate/Party	Number of Tweets
Narendra Modi/BJP	1,35,822
Rahul Gandhi/Congress	27,868
Arvind Kejriwal/AAP	20,222
Mayawati/BSP	161

3.5. Data Preprocessing

Data preprocessing is the fundamental step in any text mining analysis. It is the required step before performing Natural Language Processing (NLP), without data preprocessing it can be difficult to calculate the polarity of any text data and we may end up getting incorrect results. The dataset extracted is a linguistic data from twitter, which means it can contain multiple dialects in a single user tweet as well as other noise like repeated words, slangs, URLs, images, extra whitespaces, etc. All this noise in the user tweet makes it difficult in performing sentiment analysis. Therefore, it is necessary to remove noise and irregular data, so that inherit, and the true meaning of a user text can be understood. Moreover, data preprocessing also reduces the overall size of the dataset as the noise is removed from the data which in turn reduces the overall execution time as well as improve the accuracy of calculating the

sentiment of the tweet. To clean the user tweets, the algorithm implemented in python programming language is discussed in the following sections.

3.6. About Python

Python is an interpreted, high-level, general-purpose programming language. It was created in the year 1991 by Van Rossum [33]. It is designed with a philosophy that focuses more on code readability. Python encourages multiple programming standards, including procedural, object-oriented, and functional programming. It is an open-source language, which has an extensive and comprehensive library. It is a dynamically typed language. Its important feature is dynamic name resolution (late binding), which allows methods and variable names binding during the execution of the program. Python is rich in open source libraries, and with a strong community, it has some of the most advanced libraries like Pandas. Pandas is a fast, powerful, flexible, and easy to use open-source library for performing data manipulations and analysis. It helps to easily summarize the data using aggregations. Pandas come inbuilt with plotting functions, which helps the user to visualize the data while doing the analysis. Moreover, python also has other libraries like NumPy, open NLP, Sklearn, Scipy, VADER, and more for exploratory data analysis and data modeling.

While implementing this thesis, Python 3.7.7 version was used for data extraction, data preprocessing, sentiment analysis, and data visualization. Open-source libraries used were pandas, NumPy, NLTK, Vader, matplotlib, plotly and wordcloud. "Pycharm IDE 2020" was used as an interface for implementation, code compilation, debugging, and error handling. It has many other features which enable ease of writing code.

3.7. Data Preprocessing Methodology

Data set available on 2019 Indian Lok Sabha Elections contains a text field which maps to the

user tweet. These tweets are full of noise and contains irregular text, special characters and can also contain extra spaces and punctuations.

Therefore, in order to calculate POSITIVE, NEGATIVE and NEUTRAL polarity of each user tweet, the pre-requisite step is to clean each user tweet. Figure 3.3 shows the high-level view for data preprocessing –

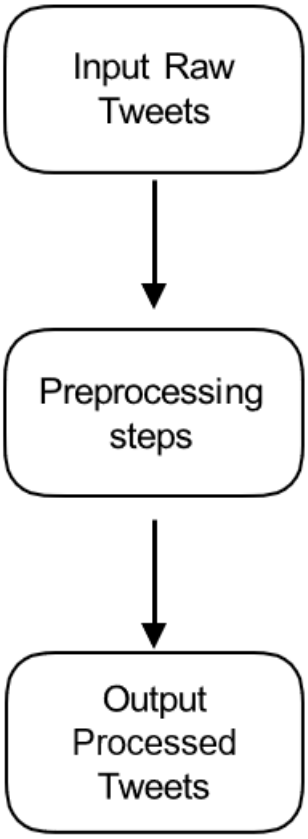


Figure 3.3 Overview of Data Preprocessing

Table 3.4 describes the algorithm implemented in python for cleaning and removing noise from each user tweet.

Table 3.4: Data Preprocessing algorithm

Input - User Tweets

Output - Processed and cleaned user tweets

For each user tweet in **dataset**:

1. Remove all https:// or URL using regular expression methods.
2. Replace all '@username' with the word 'username'.
3. Filter All #Hashtags and RT from the tweets.
4. Look for repetitions of two or more characters and replace with the character itself.
5. Filter all additional special characters (: \ ; {} - | [] + ( )? ! @ #< > % *,) from the tweets. This also includes striping the extra white spaces.
6. Removing stop words like I, am, each, the, and, etc.
7. Replace wrong contractions with custom python dictionary containing mapping of correct english contractions.
8. Replacing keywords related to political parties.

Return processed tweet

The algorithm described for each user tweet, first removes any URL present, then it removes symbols like @, # and RT plus any special characters like (: \ ; {} - | [] + ( )? ! @ #< > % *". Furthermore, it also checks for repetition of characters if present removes the repeated characters. Lastly it also removes stop words, extra white spaces, punctuations and replaces wrong contractions with correct contractions. In the following sections of the chapter we will discuss preprocessing in more detail with examples.

3.8. Preprocessing Steps

In this section of the chapter, preprocessing steps are broken down into sub steps are mentioned in the algorithm and are explained in detail with an example user tweet.

3.8.1. Removing Links/URL

The first step of cleaning the tweet removes any link/URL. The user tweets may include URL which doesn't add up to the overall context of the tweet and in our approach web content is not taken into consideration for calculating the polarity of the tweet, therefore removing any links or URLs will not affect the overall meaning of the tweet as well as reduce the overall size of the dataset. Python "re" package is loaded, which is used to clean the textual data using a regular expression. Example shown in Table 3.5 contains a link starting with "https://" and after this step, processed tweet will not contain the URL.

Table 3.5: Example removing URL from user tweet

User Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President… https://t.co/1get4yawWi
Step - 1: Processed Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President…

3.8.2. Removing hashtags and username symbols

The second step of cleaning the tweet focuses on removing @ and # symbols from the user tweets. @ in a user tweet is used for mentioning a user account, and # is used by users posting the tweet for supporting the trend in Twitter. Twitter internally uses these two symbols, @ for identifying which user account is mentioned in the tweet and # for finding

what's trending in twitter. For calculating the sentiment of the user tweet, it is found that these symbols or characters do not add up to the overall context of the user tweet. Therefore, removing such symbols or characters from tweets can significantly reduce the complexity in finding the overall polarity of the tweet. Python's regular expression is used to remove such characters and symbols. Table 3.6 shows an example of how the tweet is transformed after the second step.

Table 3.6: Example removing @ and # from user tweet

User Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President… https://t.co/1qet4yawWi
Step - 1: Processed Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President…
Step - 2: Processed Tweet	RT Sathagni: If PMOIndia narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say GoBackModi But If Vice President…

3.8.3. Removing Retweet (RT) character

The third step of data preprocessing involves removing the character "RT" which in twitter dictionary means a user tweet is retweeted. Twitter users retweet an original tweet to increase its reach across the community or to add their opinion on the original tweet. So, in- order to identify if the tweet is retweeted, twitter uses the special character "RT". The retweeted tweet by the user always starts with "RT" followed by the original tweet. In our analysis, what matters is the text of the user tweet, not these special characters; therefore, it is in the best interest to remove these special characters; this further reduces the complexity in calculating the polarity/sentiment of the user tweet. Table 3.7 shows the processed tweet after step 3.

Table 3.7: Example removing RT character from user tweet

User Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President… https://t.co/1qet4yawWi
Step - 1: Processed Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President…
Step - 2: Processed Tweet	RT Sathagni: If PMOIndia narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say GoBackModi But If Vice President…
Step - 3: Processed Tweet	Sathagni: If PMOIndia narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say GoBackModi But If Vice President…

3.8.4. Removing extra white spaces and additional special characters

The fourth step of preprocessing is cleaning extra white spaces and additional special characters. Most of the user tweet contains extra white spaces and additional special characters like brackets, +- signs, percentage signs, asterisks, and more. These characters do not have a specific meaning, and neither these characters add positivity or negativity to the overall context of the user tweet. Many times, special characters are added as a suffix to a particular emotion. For an example, the word "great" with the exclamation mark-(great!) is measured by the user to put extra weight on the same word, whereas, in other methods based on the lexicon, each and every word from the user tweet is getting compared with the corpus, there are more chances of the word “great with an exclamation mark(great!)" may not be available and not able to catch the meaning relevant to it. So even though the comment was positive, but the corpus is not able to find the meaning because of the additional special characters; it would rather decrease the polarity of the positive comment and making it a neutral comment, thereby giving incorrect results. Table 3.8 shows the processed tweet after step 4.

Table 3.8: Example removing extra white spaces and special characters from user tweet.

User Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President… https://t.co/1get4yawWi
Step - 1: Processed Tweet	RT @Sathagni: If @PMOIndia @narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say #GoBackModi But If Vice President…
Step - 2: Processed Tweet	RT Sathagni: If PMOIndia narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say GoBackModi But If Vice President…
Step - 3: Processed Tweet	Sathagni: If PMOIndia narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say GoBackModi But If Vice President…
Step - 4: Processed Tweet	Sathagni If PMOIndia narendramodi Comes to Andhra Pradesh Bicycle Buffoons Say GoBackModi But If Vice President

3.8.5. Removing repeated characters in a word

The fifth step in pre-processing is the removal of repeated characters in a word. Humans are always termed as social beings; they like to express. Users on twitter while writing a tweet are not always formal, elongation or repeated characters are very common while expressing emotions. For example, the word “Congratulationsssssss” in Table 3.9, the word contains more than one “S” and is unnecessary as it does not belong to any lexical corpuses and can mislead the overall context of the user tweet. Therefore it is essential to wisely remove the repeated characters from the word because for example “Goooooooood”, it contains more than two “Os” but if we were to remove all the “Os” and keep one, the word would transform to “God” and the overall meaning of the tweet could change.

Hence before removing repeated characters from the word, a recursive WordNet lookup is implemented. WordNet is a lexical database or corpus which provides a set of similar words of a word or called Synsets. Synsets are a group of words and their synonyms that express similar polarity. Therefore, a wordNet lookup would ensure the removal of the correct number of repeated characters.

Table 3.9: Example of removing repeated characters from a user tweet.

User Tweet	@narendramodi Opposition has to prepare for 2029 lok sabha election. Congratulations INDIA, **Congratulationsssssss** PM #VijayiBharat
Processed Tweet	narendramodi Opposition has to prepare for 2029 lok sabha election. Congratulations INDIA, **Congratulations** PM VijayiBharat

3.8.6. Replacing word with contraction

The sixth step in preprocessing is rectifying errors in text or words. Contractions like “doesn’t”, “don’t”, “couldn’t”, “shouldn’t” are common in tweets as there is word limitation of 140 characters in the twitter. Therefore, it is prevalent by the users to use short forms and contractions to communicate effectively. Usually, lexical corpus doesn’t include these types of contractions rather, they contain several bi-grams like “do not”, “not well”, “does not”, “we will”, etc. and these types of bi-grams often determine the polarity of the tweets.

Therefore, the contractions should be replaced with its equivalent bi-grams. This can be achieved by first identifying the pattern to be replaced, and then each instance of the identified pattern is replaced by the corresponding substitute string. In python, this is done using a regular expression to identify the contractions and then creating a dictionary where the key is the identified contractions in the user tweets, and value is the replacement. Table 3.10 shows the example of user tweet processed by replacing the contraction “don’t” with “do not.”

Table 3.10: Example of replacing contractions from a user tweet.

User Tweet	@narendramodi Feku **don't** be afraid of Real Chowkidar #LokSabhaElections2019 https://t.co/6trZvl2p9f
Processed Tweet	narendramodi Feku **do not** be afraid of Real Chowkidar LokSabhaElections2019

3.8.7. Replacing keywords of political parties and its leaders

In India, political parties are often written by their short forms like Bhartiya Janata Party is written as BJP, Indian National Congress is written as INC/Congress, Aam Aadmi Party is written as AAP, Bahujan Samajwadi Party is written as BSP and same is for other parties. Therefore, a user tweet during the election period can either mention BJP or Bhartiya Janata Party, INC/Congress or Indian National Congress, AAP, or Aam Aadmi Party, and similarly for other parties.

A similar trend is observed in mentioning leader names in a user tweet. During the election campaigns of 2019 Lok Sabha, it was often observed that The Prime Minister of India Shri Narendra Modi was often mentioned in the user tweet as modi or modiji or chowkidar. Similarly, Rahul Gandhi was either mentioned as rahulgandhi or pappu (this was the slang name given to him by the public during the election campaign), Arvind Kejriwal was also called by Kejriwal or Mufflerman and similarly for other party leaders.

People use such naming conventions for political parties and their leaders interchangeably in their tweets. Therefore, to maintain uniformity across tweets, all the longer forms of the party names in the tweets are replaced with its short form and similarly for leaders. Table 3.11 shows, the keyword “pappu” from the original user tweet is replaced with “rahul gandhi”, this helps to identify to which party or leader this tweet is directed towards. This also benefits in

identifying which political party or leader the tweet is referring to. Moreover, it also helps to find the distribution of user tweets and the overall trend of sentiment by the party and its leader.

Table 3.11: Example of replacing keywords in a user tweet with party names or leader name.

User Tweet	Pappu ji's face is opinion poll result of today □
Processed Tweet	Rahul gandhi jis face opinion poll result today

Table 3.11 shows, the keyword “pappu” from the original user tweet is replaced with “rahul gandhi”, this helps to identify to which party or leader this tweet is directed towards. This also benefits in identifying which political party or leader the tweet is referring to. Moreover, it also helps to find the distribution of user tweets and the overall trend of sentiment by the party and its leader.

In this chapter, the data extraction and data preprocessing were explained. The methodology implemented to extract the data using selective filtering was described in detail to only extract user tweets relevant to the election. To conduct sentiment analysis, the data set needs to go through pre-processing, wherein it cleans the raw text blob of a user tweet by removing stop words, punctuations, URLs, etc. This helps sentiment analyzer like VADER to determine the sentiment of a user tweet correctly. Data Preprocessing exclusively focused on cleaning the raw tweets extracted, 8 step algorithms implemented in earlier sections helped in filtering the tweets as well as removing unwanted noise from the data. This approach also significantly reduced the size of the dataset, which further helps in calculating the polarity of the user tweet efficiently, and execution time is on the lower side. The cleaned dataset is used as an input for performing sentiment analysis and performing exploratory data analysis.

Finally, in chapter 5, results and discussion are discussed, followed by a conclusion and future work in chapter 6.

Chapter 4

Sentiment Analysis

4.1. Introduction

Humans are termed as social beings, people live across different countries in the world, and it is a diverse environment. Since globalization and the rise of internet services and social media platforms like Twitter and Facebook have grown their user base immensely. Twitter itself has more than 300 million active users monthly. People use these platforms to communicate their opinions and emotions freely. They use common languages like English, Hindi, French, Spanish, Japanese, etc. Communication by means of language is referred to as **Linguistic Communication** (Bird *et al.* 2009) [34]. Every form of communication has its own structure, but at the core level, its structure has grammar, part-of-speech, etc. Natural Language Processing is used that allows manipulation and analysis of linguistic communication. Using Natural Language Processing, one can process and analyze large amounts of unstructured data by analyzing the structure of the sentence and calculate sentiment polarity using lexical resources or corpuses like WordNet, SentiWordNet, etc. This technique involves word tokenization, stemming, lemmatization, and stop word analysis to extract the core structure of the sentence. During this process, various parts of speech like nouns, prepositions, adverbs, pronouns, adjectives, conjunctions, verbs, and interjections are taken into consideration while breaking down the sentence as well as negations are identified carefully as they can revise the polarity of the word.

The result of this process is then compared with lexicon resources or corpuses to assign a weight to each word in the sentence and calculate the sentiment intensity or polarity. Figure 4.1 shows the proposed architecture for sentiment analysis using the Natural Processing Toolkit available in the python programming language.

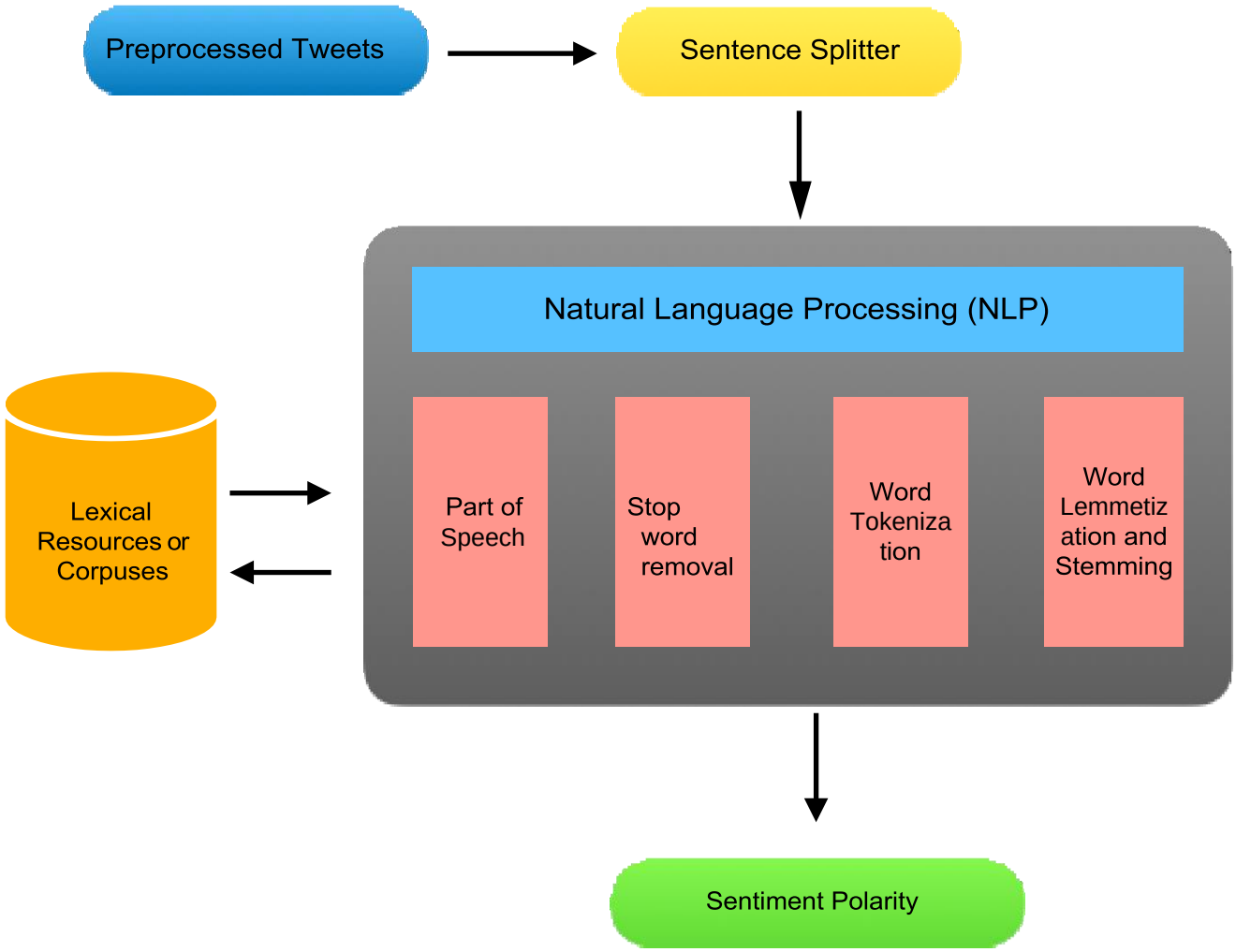


Figure 4.1: Proposed architecture for sentiment analysis using NLP

4.2. Natural Language Processing Toolkit (NLTK)

Natural Language Processing Toolkit (NLTK) in a python programming language is a framework which allows developers, researchers, and analyst to perform linguistic data processing. NLTK is a set of modules and corpora, published under the GPL open-source license, which authorizes students to learn and to conduct research in NLP (Bird et al. 2006) [35]. NLTK is accessible for Mac OS X, Windows, and Linux. Best of all, NLTK is a free, open-source, community-driven project (Bird et al. 2006).

It provides an interface to various lexical resources like WordNet, SentiWordNet, and even advanced corpuses like VADER, which is used for specifically used for processing and analyzing social media texts. In combination with corpuses and lexical resources, it provides functionality like word tokenization, stemming, tagging, parsing, stop words filtering, etc. The following subsections discuss NLTK libraries and functionality in more detail.

4.2.1. Word Tokenization

After preprocessing the extracted dataset from twitter, the raw dataset is now free of noise and unwanted text, which does not contribute to the overall context of the sentence. A sentence is a combination of multiple words, and each word individually has a meaning, and in the social media world, each word in a user tweet can have an emotion or a sentiment attached to it.

Tokenization is a process of breaking down each sentence into words. The list of words separated by a comma is the result of tokenizing the sentence into words. Natural Language Toolkit (NLTK) package in python is used for tokenization. It provides an abstract method called “word_tokenize” to split the sentences into its constituent words. Below is the algorithm used to tokenize sentences. Table 4.1 shows an example of a user tweet tokenized into list of words.

Input - Cleaned Tweets

Output - Tokenized

sentences **For** each user tweet

in **dataset**:

Tokenize the sentences into words

Return Tokenize list of words

Table 4.1: Example of tokenization of sentence.

Processed Tweet	Rahulgandhi jis face opinion poll result today
Tokenized Tweet	[‘Rahulgandhi’, ‘jis’, ‘face’, ‘opinion’, ‘poll’, ‘result’, ‘today’]

4.2.2. Word Stemming and Lemmatization

Word stemming and lemmatization are the procedures that are used to normalize the words in the sentence. It is the ability to extract the root or base of the word in the text.

Stemming

There are multiple stemming algorithms provided by Natural Language Processing Toolkit (NLTK) in python. In this thesis, the Snowball Stemmer algorithm is used for stripping the suffix from the word to extract the root of the word. It gives the root meaning to the word in text by removing various suffixes like –ED, -ING,-ION, IONS by stemming methodology and gives more abstract meaning to the word, it is the advanced version of porter stemmer. It is almost completely accepted as better than a porter stemmer.

Below is the proposed algorithm for performing stemming and lemmatization; only the words having a length greater than two is stemmed, which means words like “a”, “is”, “an”, “on”, etc. are not considered for stemming.

Lemmatization

Lemmatization is the process of transforming the structural form of a word to its root or dictionary form of the word. Natural language processing does a full morphological analysis to correctly identify the lemma of each word. Word Lemmatization in NLTK uses the WordNet database, which is basically a corpus of synonyms or a thesaurus. The significant difference between lemmatization and stemming is that lemmatization also considers the part of speech. If not passed as a parameter in NLTK, it defaults to a noun. Both processes are used to improve the quality of each sentence in the twitter dataset, which is used. Below describes the algorithm used to perform stemming and lemmatization using NLTK in python.

Input - Tokenized Sentences

Output - Stemmed and Lemmatized list of words

For each word in **tokenized_words**:

  IF word_length > 2:

    Call PorterStemmer for stemming words

    Call WordNetLemmatizer for lemmatizing words

Return Stemmed and Lemmatized list or words

Table 4.2 shows the result generated for a sample user tweet undergone tokenization, Lemmatization and stemming.

Table 4.2: Example of Word Stemming and Lemmatizing.

Processed Tweet	modi rained sops kumbhmela even build loha pool gangasagar promised gobackmodi
Tokenized Tweet	[modi, **rained**, **sops**, kumbhmela, even, build, loha, pool, gangasagar, **promised**, gobackmodi]
Stemmed & Lemmatized Tweet	[modi, **rain**, **sop**, kumbhmela, even, build, loha, pool, gangasagar, **promise**, gobackmodi]

4.2.3. Removing Stop words

some words are considered as stop words in natural language processing toolkit such as- where, when, what, he, she, it, I, we, is, the, be, etc. It is widely presumed that the removal of stop words does not have an adverse effect on determining the polarity of the text and also decreases the overall size of the data. Table 4.3 shows the example of processed tweets after removing the stop words from the original tweet.

Table 4.3: Example of removing stop words from a user tweet.

User Tweet	I have voted for strong government at centre. I have voted for @narendramodi ji. #NaMoAgain #VoteKarIndia… https://t.co/KcBh14glO
Processed Tweet	voted strong government at centre voted narendramodi ji NaMoAgain VoteKarIndia

The steps discussed in previous sections, which include word tokenization, lemmatization, stemming, and removal of stop words, make the dataset ready for performing sentiment analysis and calculating polarity for each user tweet. The steps not just ensure the readiness

of the dataset but also allows the sentiment analyzers to determine the polarity of each user tweet efficiently and accurately. Next discuss the use of VADER sentiment analyzer.

4.3. Semantic Lexicons

The majority of sentiment analysis approach fallback majorly on underlying sentiment lexicon. A sentiment lexicon is a record of lexical features, e.g., words commonly labelled according to their either positive or negative semantic orientation (Liu, 2010) [36]. It is also possible to manually create and validate lists of opinion bearing features, which many times prove to be more reliable for generating sentiment lexicons, but it can also be highly time-consuming. For this reason, most of the work done by the researchers still heavily uses predefined and modelled sentiment lexicons.

Since lexicons are the core part of any sentiment analysis procedures, the discussion will be presented for majorly two types of sentiment lexicons one polarity based and other valence based. Furthermore, the discussion will be concluded by how VADER uses both the approaches in combination to calculate the polarity of social texts efficiently and accurately, which prior two approaches are not good at handling.

4.3.1. Semantic Orientation (Polarity-based) Lexicons

LIWC (Linguistic Inquiry and Word Count) is a computer program that was built to analyze the various components such as emotional, cognitive, structural, and process which one is present in the text samples. LIWC uses its own dictionary (76 categories and around 4500 words). LIWC is one of the most reliable and validated tools; researchers use this lexicon to calculate sentiment polarity of social media text. LIWC's lexicon has been used to extract indications of political sentiment from tweets (Tumasjan- et al. 2010) [1], predicts the onset of the crisis in individuals based on text from social media (De Choudhury- et al. 2013) [37];, characterize the emotional variability of pregnant mothers from Twitter posts (De

Choudhury, Counts, & Horvitz, 2013) [37]; it secretly averages the national optimism based on Facebook

status updates (Kramer- et.al 2010) and is has been used for altering happy romantic couples from unhappy ones based on instant message conversations (Hancock, Landrigan, & Silver, 2007) [38].

However, like Gilbert, Hutto &Yardi [9] pointed out, even though is used for assessing sentiment in a social media text, LIWC does not include an application for sentiment lexical analysis parts such as "acronyms, emoticons, initialisms, or slang" which are known to be essential for sentiment analysis of the social text (Davidov, Tsur, & Rappoport, 2010) [39]. Moreover, LIWC is incapable of finding the differences in the sentiment intensity of the text. For example, “The food was awesome” conveys

more positive sentiment than “The food was okay”. LIWC would score both the sentences equally, rather such differences matter most in social media sentiment analysis as it demands a more precise look at each word in the text.

4.3.2. Semantic Intensity (Valence-based) Lexicons

The ability to calculate the sentiment intensity would help researchers and analysts to do more rigorous and in-depth study and rather not just settle down with binary polarity of a positive or negative sentiment of a text. Valence based lexicons allow us to find the sentiment intensity in text. It is very crucial and helpful for analysts and research to understand how the sentiment intensity for a product has changed over the period. Due to these reasons having a lexicon with strength valence would be helpful as it allows analysts and researchers to rhetorically analyze the change in user interest over a time.

The extension of the WordNet database is SentiWordNet, which provides the valence strength of the text. Each word in the sentence is attached with a numerical score, which ranges between 0 to 1. These scores are calculated using highly complex semi-supervised algorithms. Although it is not the gold standard resource like WordNet, LIWC, etc., it is helpful in determining the strength of sentiment for most domains. Natural Language Processing Toolkit (NLTK) in Python provides the interface for SentiWordNet for the use of analysts and researchers. But often, SentiWordNet fails to determine the sentiment intensity for social media text correctly.

4.3.3. VADER (Vader Aware Dictionary for sEntiment Reasoning)

The major short coming of the two approaches discussed was that they are unable to handle social media texts. VADER promises to leverage the benefits of rule-based modeling for calculating the sentiments of text by adding the support of social media style texts, yet able to efficiently handle textual data from other domains too. This is achieved by adding additional lexical features commonly used in lexical resources like SentiWordNet, WordNet, LIWC, etc. to express sentiment in social media text (emoticons, slang, acronyms). (**Hutto & Gilbert 2013)** [9] points out that the VADER

lexicon performs very well in the social media domain. The correlation coefficient determines that VADER (r = 0.881) performs as well as singular human raters (r = 0.888) at rivaling ground truth (aggregated group mean from 20 human raters for sentiment intensity of each tweet). Even though the corpus used in VADER has more than 9000 lexical features which is higher than other gold standard lexical resources it is still fast enough to be used online with streaming data and does not have any performance lag.

VADER sentiment lexicon administers both polarity and strength of sentiments shown in social media contents. It provides a ‘normalized, weighted composite score’ also called as

compound score which is the sum of valence scores of each word in the lexicon, and then it is normalized to a range of (-1, 1) where -1 is most negative and +1 is most positive.

In this thesis, after the comparative study of all the types of lexical resources, VADER has been used to calculate the sentiment polarity and intensity of every user tweet. VADER provides the compound scores; the user tweets are then classified into positive, negative, and neutral categories based on the threshold value. In this analysis, the value of 0.1 is used. The rules for labelling each user tweet into sentiment category based on the threshold are shown below: -.

1. *Positive Sentiment: compound score >= 0.1*
2. *Negative Sentiment: compound score <= -0.1*
3. *Neutral Sentiment: -0.1 < compound score ≤ 0.1*

The next chapter discusses the results and discussion of the data processed. The comparative study is conducted for major political parties and its leaders based on the volume of tweets, change in sentiment, etc. from 1st February 2019 to 23rd May 2019.

Chapter 5

Results and Discussion

This chapter aims to analyze the overall twitter data collected over the campaigning period of four months and present the results of the sentiment analysis conducted for each party and its leader.

5.1. Results

The election campaigns picked up from the month of March 2019 to May 2019, but to understand the transition and analyze campaign growth, the data was extracted from user tweets from 1st February 2019 to 23rd May 2019. The dataset contains approximately 200,000 tweets; this is after filtering the tweets based on the keywords listed in the data collection section. After conducting data preprocessing by cleaning the raw tweets and mapping the tweets to the party and its candidates Tables 5.1 and Table 5.2 shows the distribution of tweets by party and candidate.

Table 5.1: Number of Tweets by Party

Party	Number of Tweets
BJP	1,54,667
Congress	33,260
AAP	8,497
BSP	185

Table 5.2: Number of Tweets by Candidate

Candidate/Party	Number of Tweets
Narendra Modi (BJP)	1,35,822
Rahul Gandhi (Congress)	27,868
Arvind Kejriwal (AAP)	20,222
Mayawati (BSP)	161

The distribution of user tweets shows that during the election campaign, BJP and Congress were the two major parties for whom the public was showing more interest, followed by AAP and BSP.

5.2. Analyzing Popularity of Party and its Candidates

Analyzing the popularity of parties and their candidate in the twitter sentiment analysis is used to understand how the user sentiment changes over the period. Parties and their leaders need to understand this shift in sentiment to adjust and change their campaigning strategies over the period during the elections. Moreover, it is also important to relate this by the change in the volume of user tweets during the elections for each party and its candidate.

5.2.1. Volume Analysis

Figure 5.1, Figure 5.2, and Figure 5.3 presented is the frequency of user tweets changing over months, each week, and each day for each party considered for comparison.

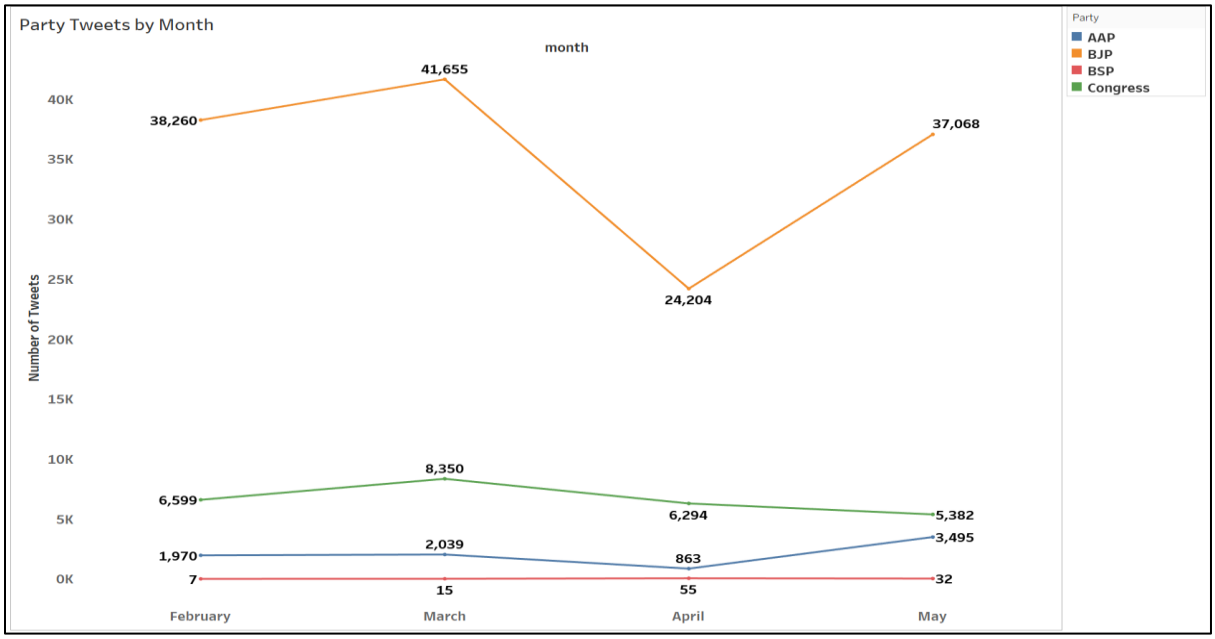


Figure 5.1: Change in frequency of tweets by month for each party

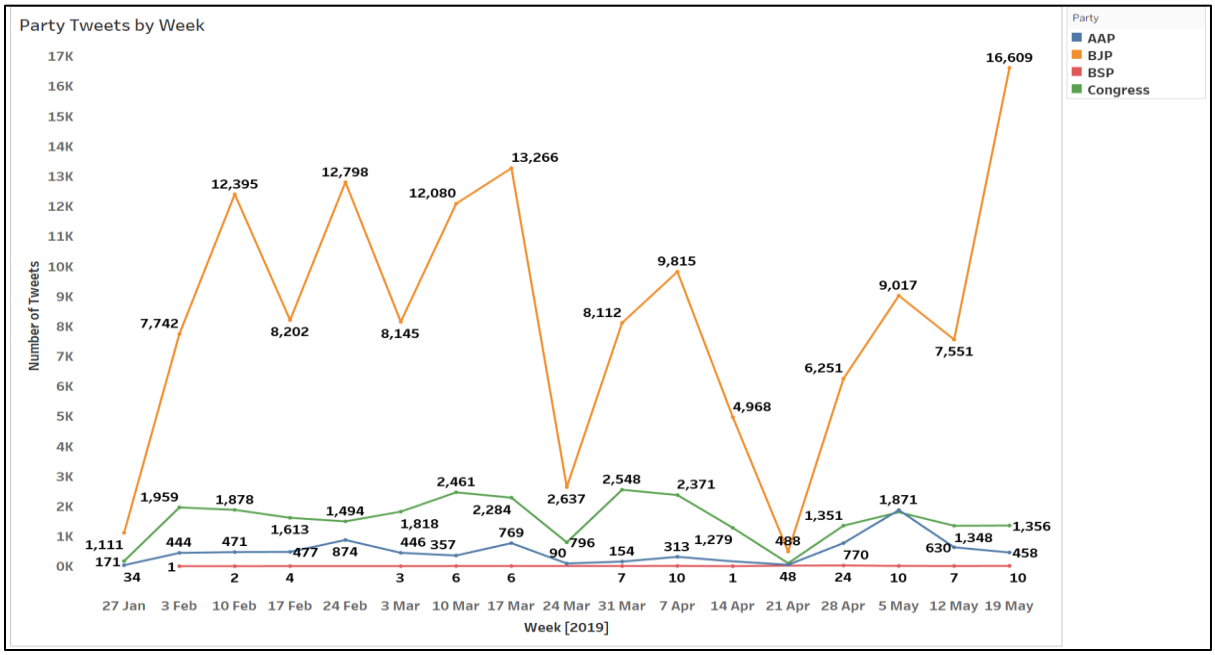


Figure 5.2: Change in frequency of tweets by week for each party

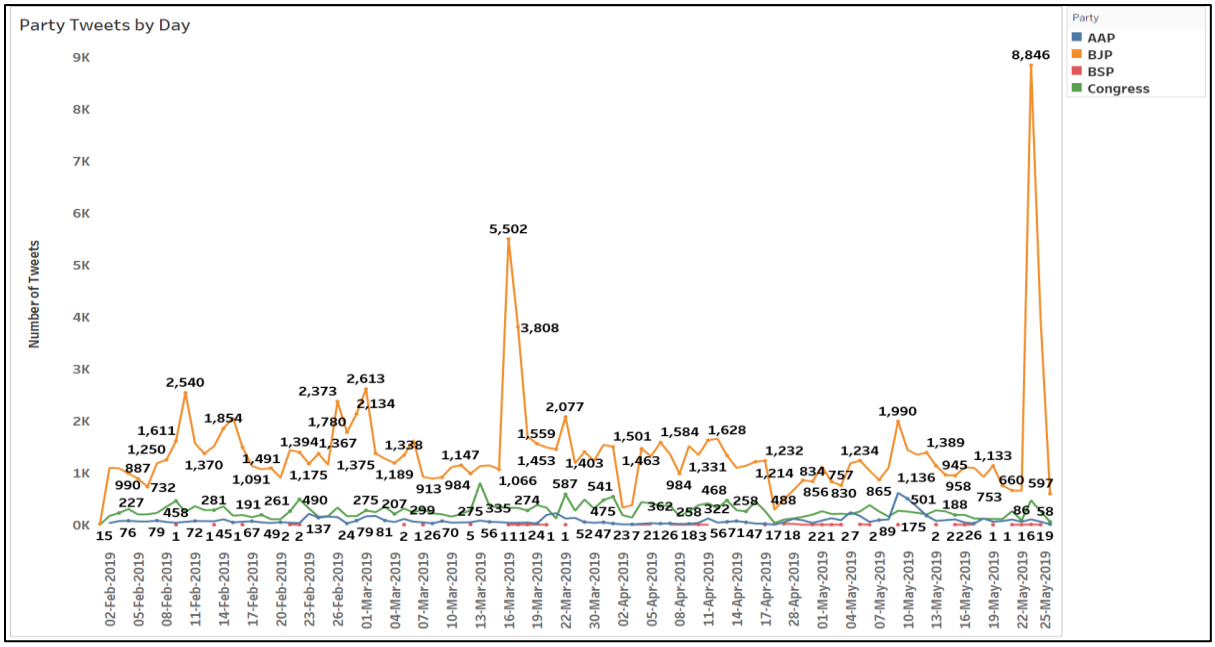


Figure 5.3: Change in frequency of tweets by day for each party

It can be clearly interpreted from Figure 5.1, Figure 5.2 and Figure 5.3 that the frequency of user tweets is more for **BJP** than its principal rival **Congress;** moreover, the frequency of user tweet has decreased for **Congress** over a period of 4 months. There is a slight increase in the number of user tweets for **AAP,** but it does not stand anywhere close to **BJP**.

A relative trend can also be seen in the frequency of user tweets for the candidate, **Narendra Modi** has far better reach over its rival candidates. Rahul Gandhi and Arvind Kejriwal user tweets very less compare to Narendra Modi. BSP tweets very lower to compare all leaders. This can be easily interpreted from Figure 5.4, Figure 5.5, and Figure 5.6.

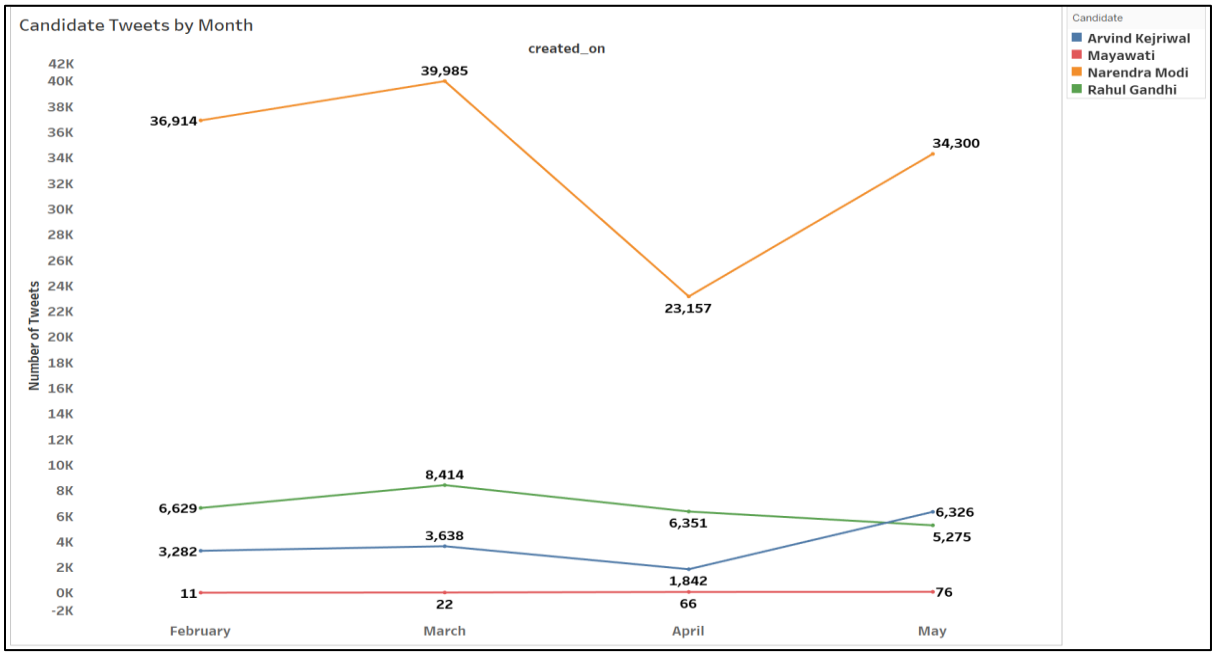


Figure 5.4: Change in frequency of tweets by monthly for each candidate

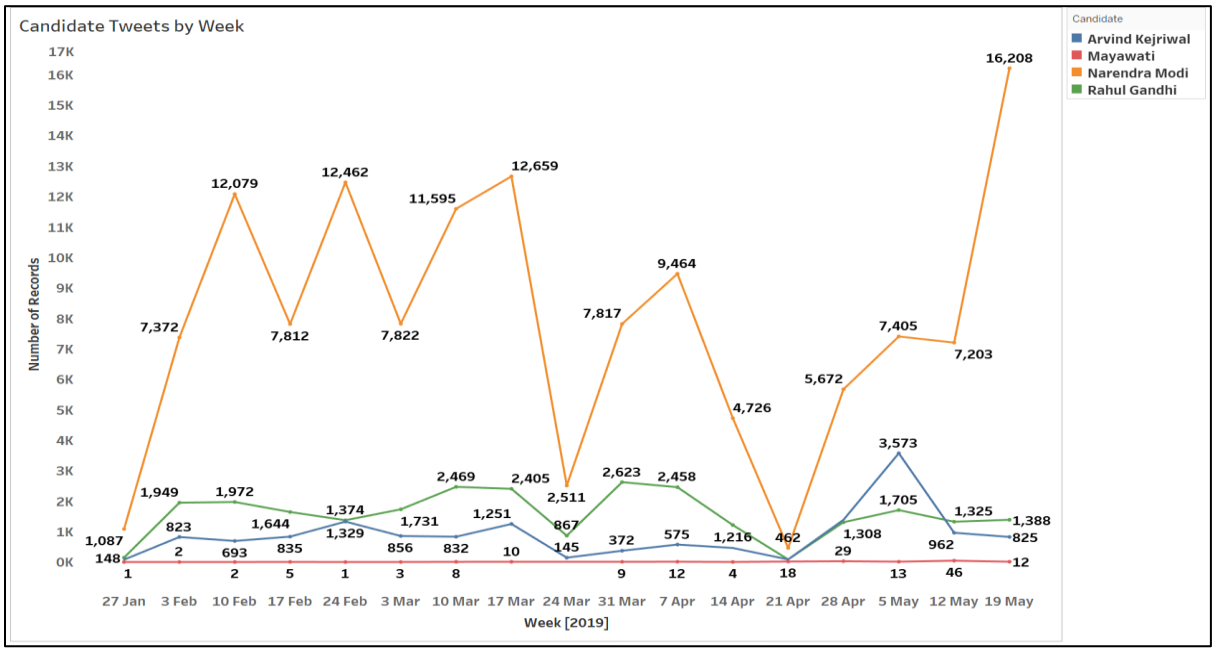


Figure 5.5: Change in frequency of tweets by week for each candidate

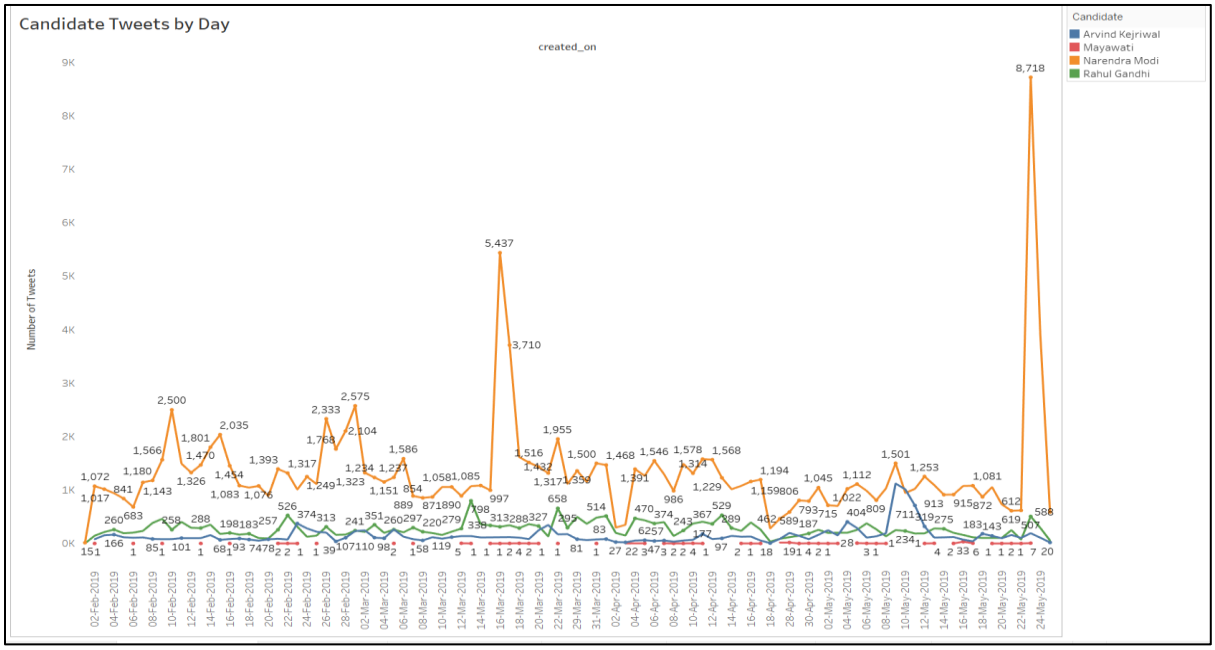


Figure 5.6: Change in frequency of tweets by day for each candidate

5.2.2. Analyzing change in sentiment

This sub-section focuses on analyzing the change in sentiment of parties and its candidate during election campaigns. As mentioned earlier, election campaigns picked up from the month of February 2019. Figure 5.7 displays the average sentiment (its compound sentiment value or sentiment strength provided by VADER) value for each month during the campaign period.

Since **BJP** was already elected at the center in the 2014 Lok Sabha elections, it was expected to have higher overall sentiment for February month. Although the plot shows that **BSP** has a higher sentiment score than all other parties, the reason being that **BSP** has a much lower number of user tweets than other parties, therefore, taking the overall sentiment score more than other parties. All the parties have seen the overall sentiment on the positive side during the initial election campaigning period, but for April and May when the actual voting started

in phases countrywide except **BJP,** all other parties has seen the shift in sentiment on the negative side, especially smaller parties like **AAP** and **BSP**. Figure 5.8 and Figure 5.9 plots displays average each party sentiment by each week and each day of campaign period.

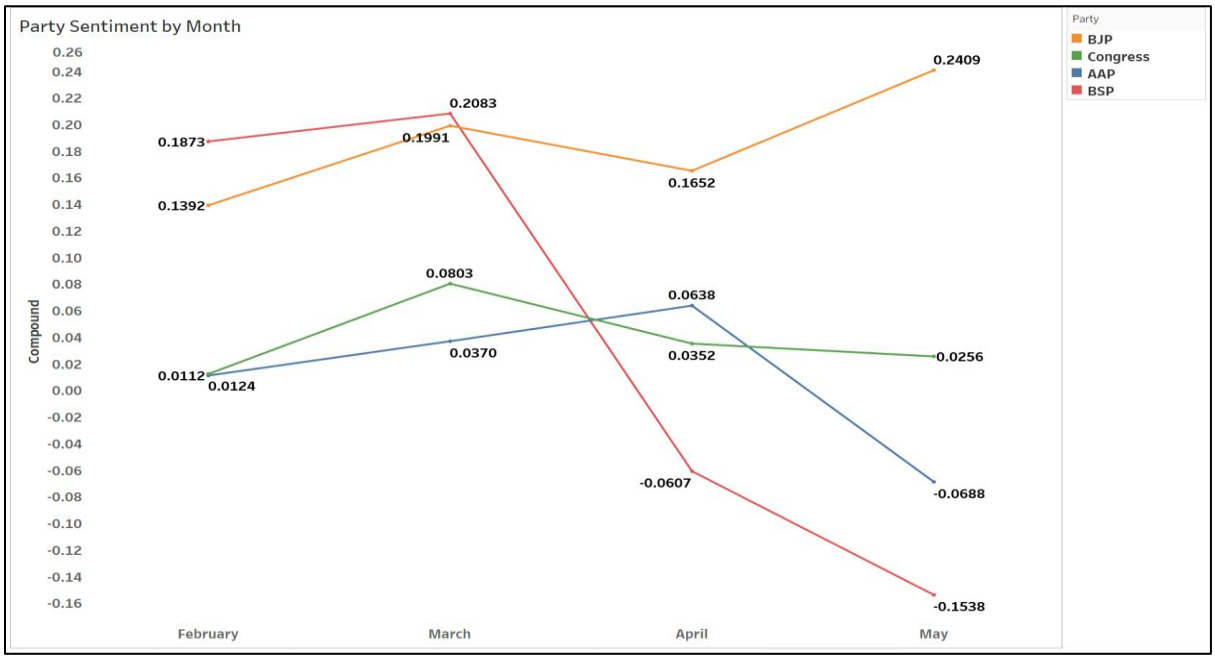


Figure 5.7: Change in Party sentiment for each month

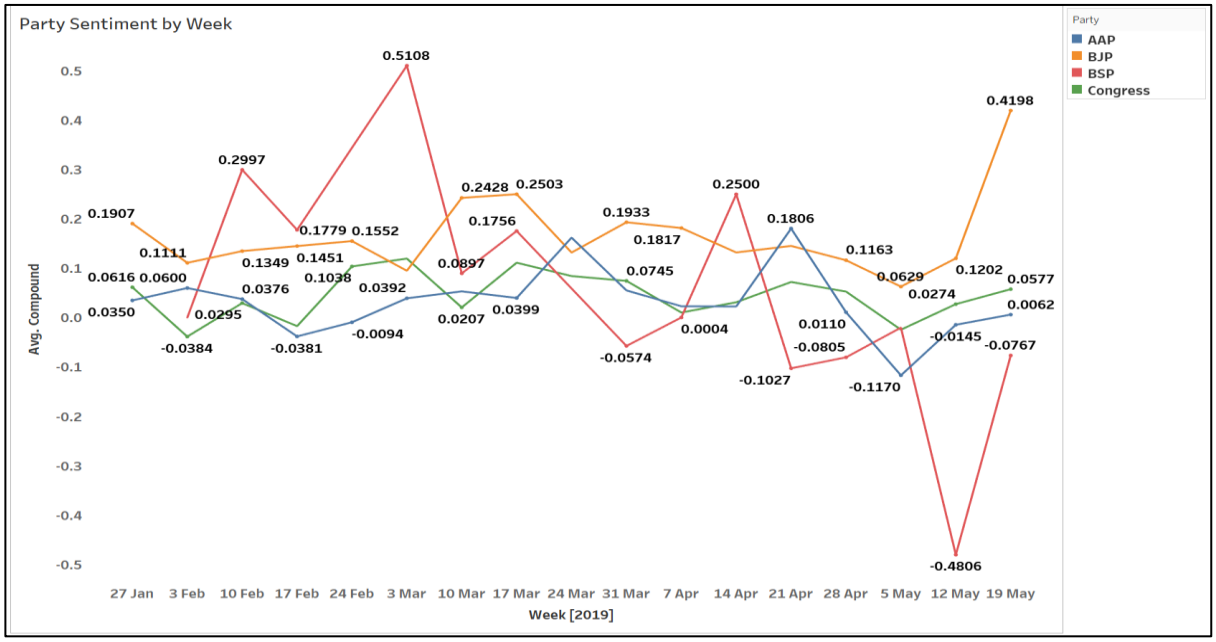


Figure 5.8: Change in Party sentiment for each week

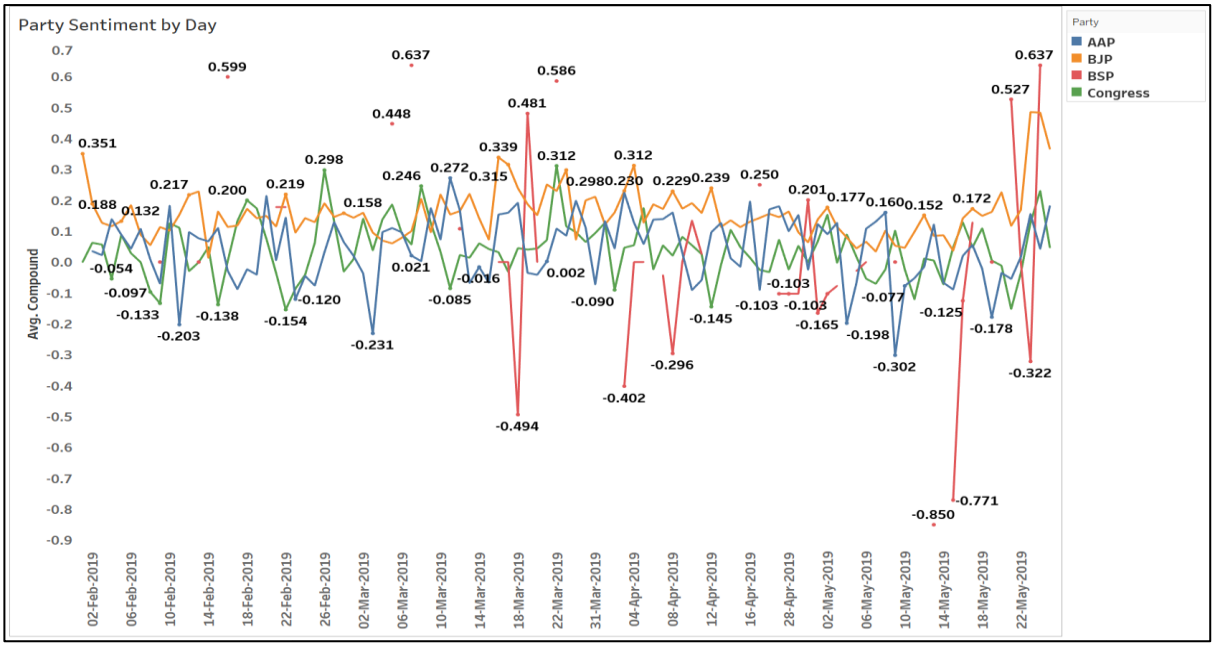


Figure 5.9: Change in Party sentiment for each day

A similar trend is observed for change in sentiment for each candidate of the parties over the campaigning period. Figure 5.10 shows that the overall sentiment of user tweets for the candidate **Narendra Modi** has doubled since elections campaigns started. The candidate appointed by Congress, **Rahul Gandhi**, has a significant drop in the overall sentiment; this is the same for **Arvind Kejriwal, AAP,** for whom the overall sentiment was negative for the month of May. **Mayawati, BSP,** although it sees a sharp rise in positive sentiment from April to May, again, the overall number of user tweets for Mayawati is much lower than other candidates; therefore, those numbers are not significant concerning other candidates. Figure 5.11 and Figure 5.12 display average sentiment of candidates by each week and each days of campaign period.

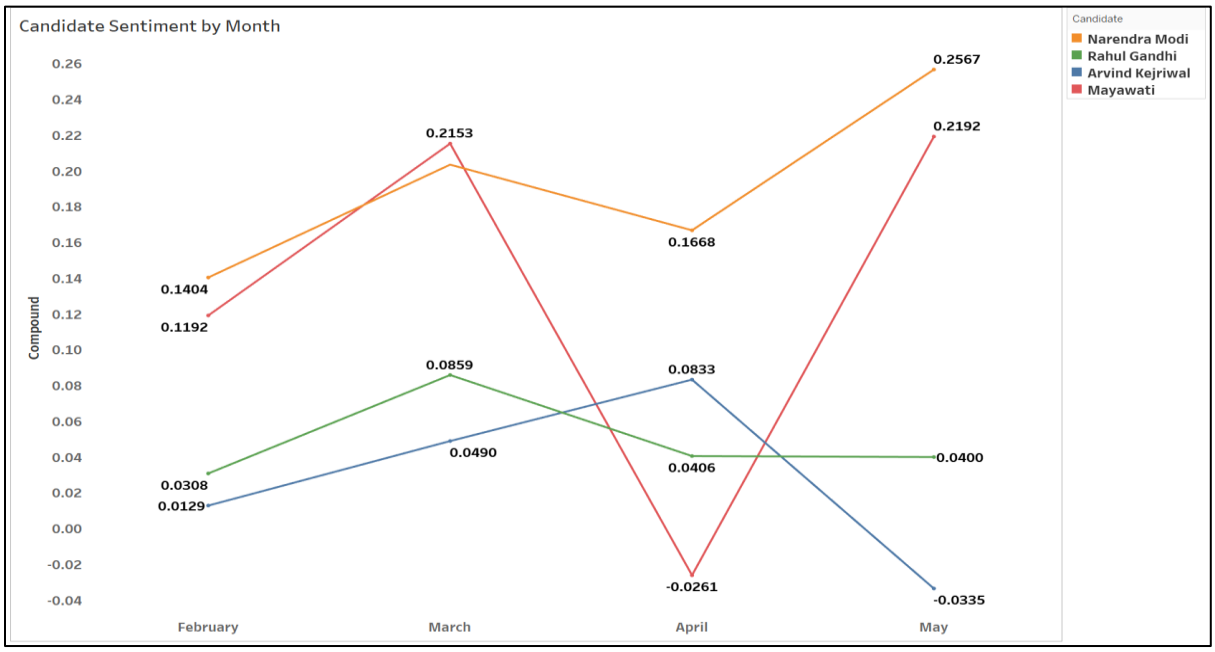


Figure 5.10: Change in Candidate sentiment for each month

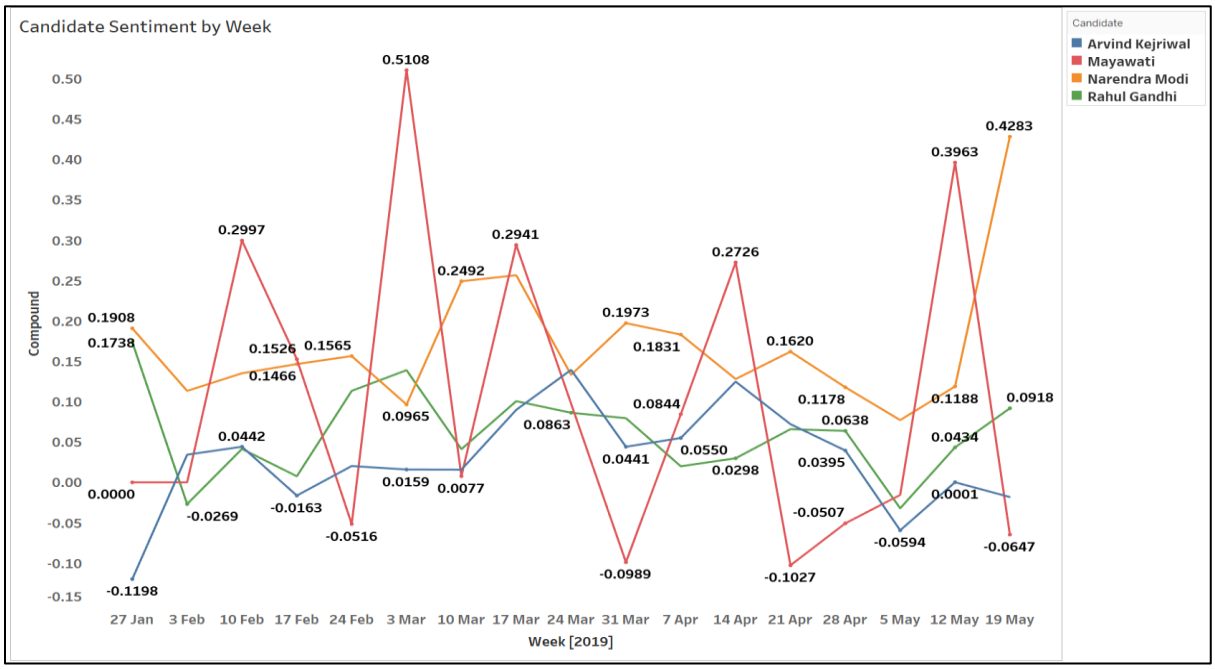


Figure 5.11: Change in Candidate sentiment for each week

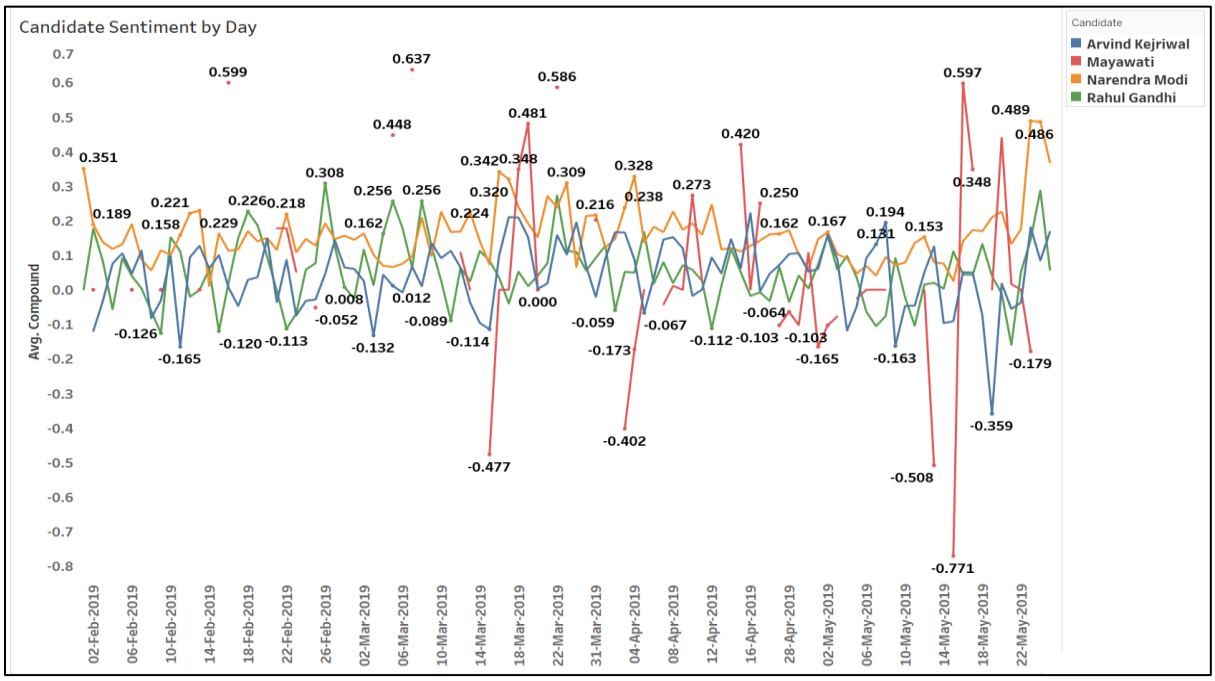


Figure 5.12: Change in Candidate sentiment for each day

From the above all plots, it can be positively stated that BJP and its leader Narendra Modi have higher overall sentiment and popularity.

5.3. Word Cloud

Word Clouds are the simplest and efficient way to understand the overall meaning of the text. Moreover, it is used to represent the frequency of various words or phrases in the user tweet. With the help of word cloud, it can easily be analyzed which words or phrases are used by the public to annotate political parties and their leaders. It can also help the political agenda of each of the parties and how the public responds to them with its opinions on twitter.

In this section, a comparison of word clouds of major parties is performed; these include BJP, Congress, and AAP. Smaller parties like BSP, DMK, TMC are not involved as the number of tweets related to these parties is not at a level with BJP and Congress, the two main rivals for 2019 Indian Lok Sabha Elections.

Figure 5.13 shown below is the word cloud for Bhartiya Janata Party, BJP and its leader Narendra Modi. The Figure reveals a positive sentiment towards BJP and its leader, phrases like "mainbhichowkidar", "best defeat", "modi ji", "citizen loves", etc. are used at frequencies than others which equates to higher popularity and following among people. "mainbhichowkidar" was one of the critical slogans among people which translates to the agenda of BJP for the 2019 elections which was strengthened Indian Army and Make in India movement.

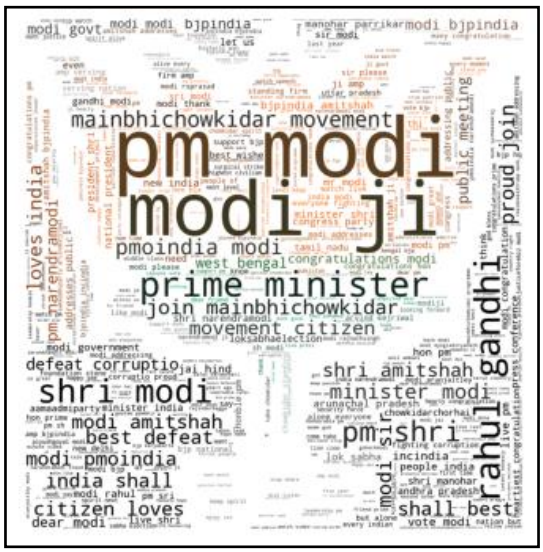


Figure 5.13: Word Cloud of BJP (Narendra Modi)

Due to the dominance of BJP over the election campaign, the word cloud in Figure 5.14, for Congress party and its leader Rahul Gandhi, does not show the high positive intensity of user

tweets.

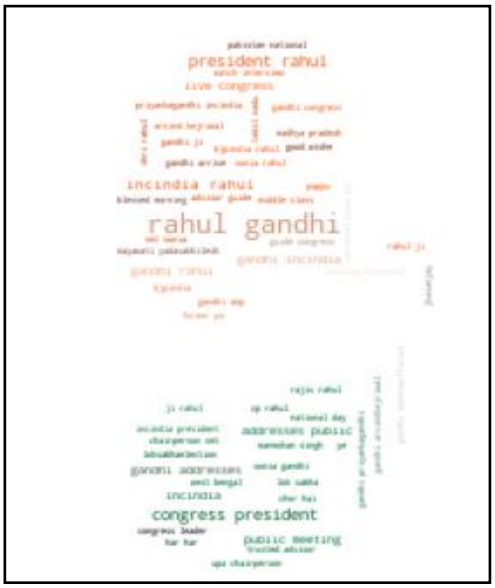


Figure 5.14: Word Cloud of Congress (Rahul Gandhi)

Similarly, for Aam Aadmi Party (AAP) and its leader Arvind Kejriwal, the word cloud in figure 5.15 shows words like “ashamed” “prove anything”, “withdraw”, etc. which highlight the negative sentiment. There are words like “cm like”, “modesty”, etc. which shows some level of positive sentiment from the public. Overall, the number of user tweets for AAP was very less than BJP and Congress, and due to those reasons overall sentiment is towards neutral which can also be seen in Table 5.3 and Table 5.4

Figure 5.15: Word Cloud of AAP (Arvind Kejriwal)

5.4. Discussion

The sentiment analysis was conducted using VADER for calculating the polarity and sentiment intensity of text. Table 5.3 and Table 5.4 shows the distribution of sentiment score for the party and its candidates in percentage since the number of tweets extracted for each party and candidate is different, so it is incorrect to compare the absolute count of tweets for each of positive, negative and neutral polarity respectively.

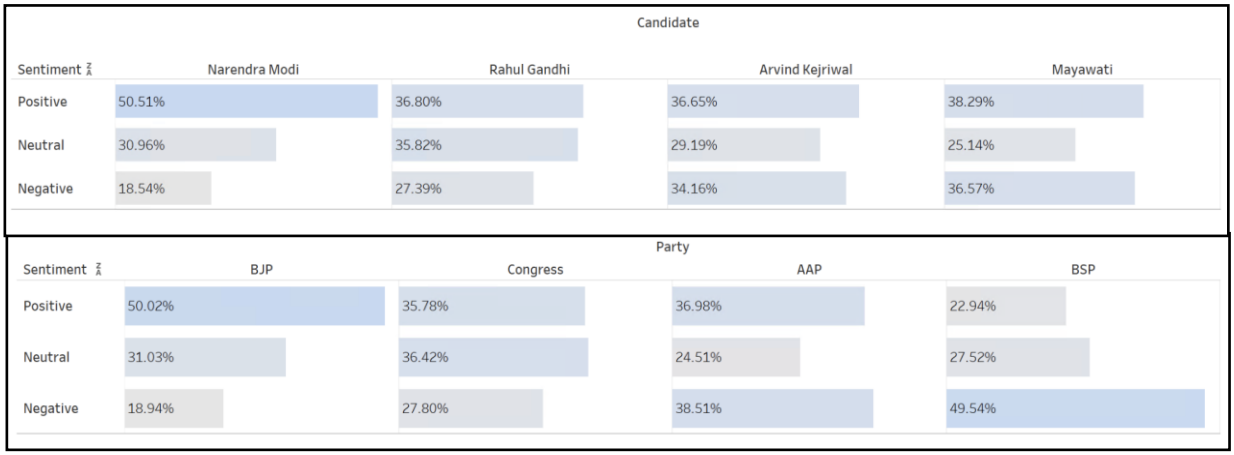
Table 5.3: Distribution of Tweet Sentiment by Party

Party	Positive	Negative	Neutral
BJP	50.02%	18.94%	31.03%
Congress	35.78%	27.80%	36.42%
AAP	36.98%	38.51%	24.51%
BSP	22.94%	49.54%	27.52%

Table 5.4: Distribution of Tweet Sentiment by Candidate

Party	Positive	Negative	Neutral
Narendra Modi	50.51%	18.54%	30.96%
Rahul Gandhi	36.80%	27.39%	35.82%
Arvind Kejriwal	36.65%	34.16%	29.19%
Mayawati	38.29%	36.57%	25.14%

In the overall results shown in Table 5.3 and Table 5.4, the positive sentiment for BSP party is 22.94%, whereas its candidate Mayawati has a positive sentiment of 38.29%. There is an approximately 15% gap between positive sentiment between BSP Party and its leader Mayawati as the number of tweets distribution for BSP and Mayawati is very low compared to all other parties. It is difficult to compare Party and its leader in this case as people show more negative sentiment for other BSP candidates except for Mayawati, as a result the overall party sentiment is more negative than its leader. That is why BSP sentiment is showing 49.54% negative and only 22.94% positive. But as people think positive for Mayawati, so candidate wise, Mayawati gets a positive sentiment of 38.29%. The tweets distribution for other parties and its leaders is more than BSP and its leader. BJP, Congress, and AAP and its leaders Narendra Modi, Rahul Gandhi, and Arvind Kejriwal have consistent results with the tweet distribution. It can be clearly inferred from Table 5.3 and Table 5.4 that both **BJP** and



its leader **Narendra Modi** have higher positive sentiment. Also, the number of tweets extracted based on keywords is more than the other parties and their leaders, which shows Narendra Modi had more popularity and positive opinion among the people. Figure 5.16 shows higher negative sentiment for the parties **AAP** and **BSP** and their leaders **Arvind Kejriwal and Mayawati**, respectively than **BJP and Congress** and their leaders **Narendra Modi** and **Rahul Gandhi**, respectively.

Figure 5.16: Distribution of Sentiment by Party and Candidate

The election results in Table 5.6 show that **Congress** won second highest number of seats in the Lok Sabha election 2019. Due to higher popularity of Narendra Modi among people and since he was the existing Prime Minister during the 2019 elections, BJP won the election with a whopping 303 seats out of contested 542 seats. Rahul Gandhi, the leader of Congress party failed to reserve the majority and in contrast, Congress was able to win merely 52 seats.

Party	No of Seats Won
BJP	303
Congress	52
BSP	10
AAP	1

Table 5.6: Distribution of seats won in parliament in Lok Sabha election 2019

Chapter 6

Conclusions and Future Work

6.1. Conclusions

In this thesis, our target was to conduct twitter sentiment analysis for the 2019 Indian Lok Sabha Election for major parties and their candidates. User tweets were extracted from twitter archiver between 1st February to 23rd May 2019 based on keywords and hashtags related to the trend observed during the election period. In order to pre-process or filter the noise from the textual Twitter data, it is necessary to perform the sequence of pre-processing steps, which included cleaning of raw tweets and mapping the tweets to referenced party and the candidate. This overall helped in generating more accurate tweet which enables precise calculation of polarity of tweets.

A model was proposed to perform sentiment analysis on a linguistic dataset, highlighting the use of the VADER sentiment analyzer specifically attuned to calculate the polarity of text expressed in social media rather than using a traditional dictionary-based approach that maintains WordNet for positive, negative and neutral polarity keywords. It was seen that people had higher positive sentiment and emotions for Narendra Modi and its party BJP than Congress, AAP, and BSP.

The results of sentiment analysis follow the actual election results which resulted in BJP winning the election by a majority. With this, we can conclude that sentiment analysis and opinion mining can help us understand how people respond to the campaign and which party and candidate are leading the election race. In this new age of internet and its ease of access,

politician and their parties are taking full advantage of it by professionally targeting user groups for running their campaigns and leading the election race.

6.2. Future Work

The future scope of this work can include more keywords for the data extraction process and increase Tweet distribution for the party and its candidates and more parties and candidates can be added in the tweet extraction. Additionally, changes in data extraction techniques can be made by using advanced big data frameworks like Apache Spark, as such sophisticated event processing engines can process vast amounts of data. Due to advances in technology and growing digitalization, the data size of social media texts is growing manifolds and using Apache Spark or Spark Streaming will help us perform the extraction, transformation, and loading in near real-time. This can be achieved by running spark jobs in a cluster environment created on AWS or Azure cloud services. This thesis mainly concentrated efforts on classifying positive, negative, or neutral sentiments correctly, but it would be interesting to distinguish between subjective and objective tweets since this would be a high initial filter. Also, the approach handles the topic of classification for tweets based on the presence or absence of a keyword that we called “main feature”, but a more elaborated method could be developed where the context or even synonyms are taken into consideration. Not only expanding the amount of manually classified datasets but also expanding the domain of the inputs would be interesting. In this study, the focus was on Twitter, but more social networks could be added. Additionally, it would be interesting to create datasets with a more balanced distribution of tweets from different classes with a balancing algorithm.

References

1. Indian General Election: https://en.wikipedia.org/wiki/2019_Indian_general_election
2. World bank internet: https://data.worldbank.org/indicator/IT.NET.USER.ZS?locations=IN
3. Demographics of India: https://en.wikipedia.org/wiki/Demographics_of_India
4. Tumasjan, A.; Sprenger, T. O.; Sandner, P.; and Welpe, I. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In Proceedings of ICWSM
5. O"Connor, B.; Balasubramanyan, R.; Routledge, B. R.; and Smith, N. A. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In ICWSM
6. Adam Bermingham and Alan F Smeaton. 2011. On using Twitter to monitor political sentiment and predict election results. In Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP 2011), pages 2–10.
7. A. Das and S. Bandyopadhyay, "SentiWordNet for Bangla," Knowledge Sharing Event-4: Task, Volume 2, 2010.
8. D. Das and S. Bandyopadhyay, "Labeling emotion in Bengali blog corpus - a fine grained tagging at sentence level," Proceedings of the 8th Workshop on Asian Language Resources, pp. 47–55, Aug. 2010.
9. C. J. Hutto and E. E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media," presented at the Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media, Ann Arbor, MI, 2015.
10. M. Eirinaki, S. Pisal, and J. Singh, "Feature-based opinion mining and ranking," Journal of Computer and System Sciences, vol. 78, no. 4. pp. 1175–1184, 2012, doi: 10.1016/j.jcss.2011.10.007.

11. B. Liu, Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers, 2012.
12. D. J. S. Oliveira, P. H. de Souza Bermejo, and P. A. dos Santos, “Can social media reveal the preferences of voters? A comparison between sentiment analysis and traditional opinion polls,” Journal of Information Technology & Politics, vol. 14, no. 1. pp. 34–45, 2017, doi: 10.1080/19331681.2016.1214094.
13. Kim, S., Hovy, E. Determining the sentiment of opinions. International conference on Computational Linguistics (COLING’04). 2004.
14. Parrott, W. G. "Emotions in social psychology: Volume overview." Emotions in social psychology: Essential readings. Ed. W. G. Parrott. Philadelphia: Psychology Press, 2001: 1-19.
15. Medhat, W., Hassan, A., & Korashy, H. Sentiment Analysis algorithms and applications: A survey. AinShams Engineering Journal, 5(4), 1093-1113. 2014.
16. Luciano Barbosa and Junlan Feng. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In Proceedings of the international conference on Computational Linguistics (COLING), 2010.
17. R. Jose and V. S. Chooralil, "Prediction of election result by enhanced sentiment analysis on Twitter data using Word Sense Disambiguation," Proc. Int’l Conf. on Control Communication & Computing India (ICCC), Trivandrum, 2015, pp. 638-641.
18. M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, “Lexicon-based methods for sentiment analysis,” Computational. Linguististics. vol. 37, pp. 267-307, 2011.
19. Chetashri Bhadane, Hardi Dalal, Heenal Doshi, “Sentiment analysis: Measuring opinions,” International Conference on Advanced Computing Technologies and Applications (ICACTA2015), Procedia Computer Science vol.no.45, pp. 808 – 814, 2015.

20. Abroms, L. C., & Craig Lefebvre, R. (2009). Obama's wired campaign: Lessons for public health communication. Journal of Health Communication, 14(5), 415–423.
21. Baumgartner, J. C., Mackay, J. B., Morris, J. S., Otenyo, E. E., Powell, L., Smith, M. M., ...Waite, B. C. (2010). Communicator-in-chief: How Barack Obama used new media technology to win the White House Edited by John Allen Hendricks, and Robert E. Denton Jr. Lexington Books.
22. Glassman, M., Straus, J. R., & Shogan, C. J. (2009). Social networking and constituent communication: Member use of Twitter during a two-week period in the 111th Congress. Library of Congress: Congressional Research Service.
23. Golbeck, J., Grimes, J. M., & Rogers, A. (2010). Twitter use by the US congress. Journal of the American Society for Information Science and Technology, 61(8), 1612–1621.
24. M. Fernández-Gavilanes, T. Álvarez-López, J. Juncal-Martínez, E. Costa-Montenegro, and F. Javier González-Castaño, "Unsupervised method for sentiment analysis in online texts," Expert Systems with Applications, vol. 58, pp. 57–75, 2016.
25. P. Nakov, S. Rosenthal, S. Kiritchenko et al., "Developing a successful SemEval task in sentiment analysis of Twitter and other social media texts," Language Resources and Evaluation, vol. 50, no. 1, pp. 35–65, 2016.
26. Sharmistha Chatterjee - https://hackernoon.com/twitter-sentiment-analysis-for-the-2019-lok-sabha-elections-b43o320e
27. P. Sharma and T. Moh, "Prediction of Indian election using sentiment analysis on Hindi Twitter," *2016 IEEE International Conference on Big Data*, Washington, DC, 2016, pp. 1966-1971, DOI: 10.1109/BigData.2016.7840818.
28. Gaikar, D.D., Sapare, G., Vishwakarma, A., & Parkar, A. (2019). Twitter Sentimental Analysis for Predicting Election Result using LSTM Neural Network.
29. Salunkhe, P., & Deshmukh, S.N. (2017). Twitter Based Election Prediction and Analysis.

30. Simplifying Sentiment Analysis using VADER in Python (on Social Media Text) - https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f

31. Hemalatha, I., Dr GP Saradhi Varma, and A. Govardhan. "Preprocessing the informal text for efficient sentiment analysis." International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) 1.2 (2012): 58-61.

32. Twitter Archieve -: https://archive.org/details/twitterstream

33. Van Rossum, Guido. "Python Programming Language." USENIX Annual Technical Conference. Vol. 41. 2007.

34. Bird, Steven, Ewan Klein, and Edward Loper. Natural language processing with Python. “O’Reilly Media, Inc.", 2009.

35. Bird, Steven. "NLTK: the natural language toolkit." Proceedings of the COLING/ACL on Interactive presentation sessions. Association for Computational Linguistics, 2006.

36. Liu, B. (2010). Sentiment Analysis and Subjectivity. In N. Indurkhya & F. Damerau (Eds.), Handbook of Natural Language Processing (2nd ed.). Boca Raton, FL: Chapman & Hall.

37. De Choudhury, M., Counts, S., & Horvitz, E. (2013). Predicting Postpartum Changes in Emotion and Behavior via Social Media. In Proc. CHI-13.

38. Hancock, J. T., Landrigan, C., & Silver, C. (2007). Expressing emotion in text-based communication. In Proc. CHI-07.

39. Davidov, D., Tsur, O., & Rappoport, A. (2010). Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. ICCL-10.

Appendix A

Source code

Twitter Tars Extraction

```
import os
files_lst = []
for root, dirs, files in os.walk("D:/TwitterTars/2019-05/"):
    for filename in files:
        files_lst.append(filename)

files_lst

import tarfile
for file in files_lst:
    print('Processing file: '+ file)
    my_tar = tarfile.open('D:/TwitterTars/2019-05/' + file)
    tar_members_names = [filename for filename in my_tar.getnames()]
    my_tar.extractall('D:/TwitterData/ExtractedFiles/2019-05/') # specify
which folder to extract to
    my_tar.close()
```

Tweets Extraction

```
import json
import os
import profile
import shutil
import datetime
import pandas as pd
from pprint import pprint
from s3_manager import S3Manager
from io import BytesIO


#get tweets from hastags

def get_hashtags(tweet):
    hashtags = []

    if len(tweet['entities']['hashtags']) > 0:
        for i in range(0, len(tweet['entities']['hashtags'])):
            hashtag = tweet['entities']['hashtags'][i]['text']
            hashtags.append(hashtag)
    else:
        hashtags = []

    hashtags = ','.join(hashtags)

    return hashtags
```

```
def get_user_mentions(tweet):
    screen_names = []

    if len(tweet['entities']['user_mentions']) > 0:
        for i in range(0, len(tweet['entities']['user_mentions'])):
            screen_name =
tweet['entities']['user_mentions'][i]['screen_name']
            screen_names.append(screen_name)
    else:
        screen_names = []

    screen_names = ','.join(screen_names)

    return screen_names

def extract_tweet_attributes(tweet, tweets_saved):

    try:
        tweet_id = tweet['id']
        tweet_text = tweet['text']
        try:
            retweeted_text = tweet['retweeted_status']['text']
            retweeted_quote_count =
tweet['retweeted_status']['quote_count']
            retweeted_reply_count =
tweet['retweeted_status']['reply_count']
            retweeted_retweet_count =
tweet['retweeted_status']['retweet_count']
            retweeted_favorite_count =
tweet['retweeted_status']['favorite_count']
        except Exception as e:
            retweeted_text = ''
            retweeted_quote_count = 0
            retweeted_reply_count = 0
            retweeted_retweet_count = 0
            retweeted_favorite_count = 0

        tweet_locale = tweet['lang']
        created_at = tweet['created_at']
        user_location = tweet['user']['location']
        user_screen_name = tweet['user']['screen_name']
        user_mentions_screen_name = get_user_mentions(tweet)
        place_type = get_place_details(tweet)['place_type']
        place_name = get_place_details(tweet)['place_name']
        quote_count = tweet['quote_count']
        reply_count = tweet['reply_count']
        retweet_count = tweet['retweet_count']
        favorite_count = tweet['favorite_count']
        hashtags = get_hashtags(tweet)

        data = {'tweet_id': tweet_id,
                'tweet_text': tweet_text,
                'retweeted_text': retweeted_text,
                'retweeted_quote_count': retweeted_quote_count,
                'retweeted_reply_count': retweeted_reply_count,
                'retweeted_retweet_count': retweeted_retweet_count,
                'retweeted_favorite_count': retweeted_favorite_count,
                'tweet_locale': tweet_locale,
                'created_at_str': created_at,
                'user_screen_name': user_screen_name,
```

```
                'user_mentions_screen_name': user_mentions_screen_name,
                'user_location': user_location,
                'place_type': place_type,
                'place_name': place_name,
                'quote_count': quote_count,
                'reply_count': reply_count,
                'retweet_count': retweet_count,
                'favorite_count': favorite_count,
                'hashtags': hashtags,
                'date_loaded': datetime.datetime.now()
                }

        tweets_saved = + 1

    except KeyError:
        data = {'tweet_id': '',
                'tweet_text': '',
                'retweeted_text': '',
                'retweeted_quote_count': 0,
                'retweeted_reply_count': 0,
                'retweeted_retweet_count': 0,
                'retweeted_favorite_count': 0,
                'tweet_locale': '',
                'created_at_str': '',
                'user_screen_name': '',
                'user_mentions_screen_name': '',
                'user_location': '',
                'place_type': '',
                'place_name': '',
                'quote_count': 0,
                'reply_count': 0,
                'retweet_count': 0,
                'favorite_count': 0,
                'hashtags': '',
                'date_loaded': datetime.datetime.now()
                }
        tweets_saved = + 1

    return data
```

```
import boto3 as bt
import pandas as pd
from datetime import datetime
import s3fs
import io
import json
import numpy as np
import re
import matplotlib
import matplotlib.pyplot as plt
import nltk, string
from nltk.corpus import stopwords
```

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import preprocessing
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem.snowball import SnowballStemmer
from textblob import TextBlob
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
from PIL import Image
```

Remove unwanted columns

```
twitter_data.drop("reply_count", axis = 1, inplace =True)
twitter_data.drop("quote_count", axis = 1, inplace =True)
twitter_data.drop("retweet_count", axis = 1, inplace =True)
twitter_data.drop("favorite_count", axis = 1, inplace =True)
twitter_data.drop("place_type", axis = 1, inplace =True)
twitter_data.drop("place_name", axis = 1, inplace =True)
twitter_data.drop("date_loaded", axis=1, inplace = True)
twitter_data.drop("created_at_str", axis = 1, inplace =True)
twitter_data.drop("tweet_locale", axis = 1, inplace =True)
twitter_data.shape
twitter_data.head()
df = twitter_data.sort_values(by=['created_at'])
df = df.drop_duplicates()
df['original_tweet'].fillna(df['tweet_text'], inplace=True)
```

Using NLTK Sentiment Intensity Analyzer

```
import nltk
nltk.download("vader_lexicon")
nltk.download("stopwords")
stemmer = SnowballStemmer('english')
vectorizer = TfidfVectorizer(use_idf = True, tokenizer =
nltk.word_tokenize,stop_words='english', smooth_idf = True)
remove_punctuation_map = dict((ord(char), None) for char in
string.punctuation)
sid = SentimentIntensityAnalyzer()
stopw = set(stopwords.words('english'))
```

Process and Clean tweets

```
contraction = {"ain't": "is not", "aren't": "are not","can't": "cannot",
                   "can't've": "cannot have", "'cause": "because",
"could've": "could have",
                   "couldn't": "could not", "couldn't've": "could not
have","didn't": "did not",
                   "doesn't": "does not", "don't": "do not", "hadn't": "had
not",
                   "hadn't've": "had not have", "hasn't": "has not",
"haven't": "have not",
                   "he'd": "he would", "he'd've": "he would have", "he'll":
"he will",
                   "he'll've": "he he will have", "he's": "he is", "how'd":
"how did",
                   "how'd'y": "how do you", "how'll": "how will", "how's":
"how is",
```

```
                    "I'd": "I would", "I'd've": "I would have", "I'll": "I
will",
                    "I'll've": "I will have","I'm": "I am", "I've": "I
"it is",
                    "let's": "let us", "ma'am": "madam", "mayn't": "may
not",
                    "might've": "might have","mightn't": "might
not","mightn't've": "might not have",
                    "must've": "must have", "mustn't": "must not",
"mustn't've": "must not have",
                    "needn't": "need not", "needn't've": "need not
have","o'clock": "of the clock",
                    "oughtn't": "ought not", "oughtn't've": "ought not
                    "that'd": "that would", "that'd've": "that would
have","that's": "that is",
                    "there'd": "there would", "there'd've": "there would
have","there's": "there is",
                    "they'd": "they would", "they'd've": "they would have",
"they'll": "they will",
                    "they'll've": "they will have", "they're": "they are",
"they've": "they have",
                    "to've": "to have", "wasn't": "was not", "we'd": "we
would",
                    "we'd've": "we would have", "we'll": "we will",
"we'll've": "we will have",
                    "we're": "we are", "we've": "we have", "weren't": "were
not",
                    "what'll": "what will", "what'll've": "what will have",
"what're": "what are",
                    "what's": "what is", "what've": "what have", "when's":
"when is",
                    "you'll've": "you will have", "you're": "you are",
"you've": "you have", "bharatiya janata party" : "bjp",
                    "inc" : "congress","@narendramodi":"modi",
"pappu":"rahul gandhi","gandhi":"rahul gandhi", "@rahulgandhi":"Rahul
Gandhi"}



def clean(text):
    text = text.lower()
    temp = ""
    for i in text.split():
        if i not in stopw:
            try:
                temp+=contraction[i]+' '
            except:
                temp+= i+' '
    text = temp.strip()
    text = re.sub(r'http\S+','', text)
    text = re.sub(r'RT @','', text)
    text = text.lower().translate(remove_punctuation_map)
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
```

```
    text = re.sub(r",", " ", text)
    text = re.sub(r"\.", " ", text)
    text = re.sub(r"!", " ! ", text)
    text = re.sub(r"\/", " ", text)
    text = re.sub(r"\^", " ^ ", text)
    text = re.sub(r"\+", " + ", text)
    text = re.sub(r"\-", " - ", text)
    text = re.sub(r"\=", " = ", text)
    text = re.sub(r"'", " ", text)
    text = re.sub(r"(\d+)(k)", r"\g<1>000", text)
    text = re.sub(r":", " : ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"\s{2,}", " ", text)
    text = text.replace("bhartiya janata party",'bjp')
    text = text.replace("indian national congress", 'congress')
    text = text.replace("aam aadmi party", 'aap')
    text = text.replace("narendra modi", 'modi')
    text = text.replace("rahulgandhi", 'rahul gandhi')
    temp=''
    for i in text:
        if i.isdigit()==False:
            temp+=i
    text = temp
    text = text.split()

#     stemmed_words = [stemmer.stem(word) for word in text]
    text = " ".join(text)
    return text.strip()

preprocessed = []
for i in df["original_tweet"]:
    preprocessed.append(clean(i.strip().lower()))
preprocessed
df.insert(4,"processed_tweet",preprocessed,True)
```

Calculate Sentiment

```
sentiment = {"Positive":[], "Negative" :[], "Neutral":[], "Compound":[]}
for i in df["processed_tweet"]:
    sentiment["Positive"].append(sid.polarity_scores(i)['pos'])
    sentiment["Negative"].append(sid.polarity_scores(i)['neg'])
    sentiment["Neutral"].append(sid.polarity_scores(i)['neu'])
    sentiment["Compound"].append(sid.polarity_scores(i)['compound'])

## Using TextBlob

def score(text):
    return TextBlob(text).sentiment.polarity

df['score'] = df['processed_tweet'].apply(score)

df['month'] = df.created_at.dt.month
df['week'] = df.created_at.dt.week
df['day'] = df.created_at.dt.day
```

Find Tweets related to different parties

```
compare = ["vijay rajnath ravishankar  yudhvirsethi patra narendra modi vijayvargiyah sadhvi bjp arun manoj reddy sushma rsprasad taneja maneka udhavthackeray gautam gambhir piyush goyal nitin gadkari gadkariji rss singh vasundhraraje bjpbengal smriti kailash gautamgambhir swamy udhav sushmaswaraj sadhvi pragya vasundhra sambitpatra shivraj arunjaitley manohar parikar subramanianswamy naredra modi manojtiwari amit modiji yogi adityanath sushma swaraj nitin vivekreddy shivrajsinghchouhan vijayrupani amit shah narendramodi pragya arun jaitley thackeray sunny deol bharatiya janata party kailashvijayvargiyah adityanath yogi jaitley piyush gadkari sambit smritiirani rajnathsingh irani swaraj gautam parikar nirmala bhartiya janta party ram nirmala sitaraman modi shivrajsingh nititngadkari manohar rammadhav smriti irani yedyurappa madhav gambhir narendra rajnath singh subrmanian goyal chouhan amitshah sitaraman manoharparikar ravishankar prasad rupani rao shah ravishankarprasad narsimha vivek vijay rupani prasad bhartiya janata party giriraj chowkidar",
          "congress rahul gandhi sonia pappu manish tiwari mani shankar aiyar amrinder singh navjot sidhu pilot sachin jyotiraditya scindia ashok gehlot ajay makhan makhen chidambaram raj babbar sheila dikshit kamal nath digvijay singh sanjay kaul ashok chavan prithviraj randeep surjaewala hooda deepender kapil sibal manmohan ahmed patil natwar gaurav vallabh pawan khera taneja reddy george antony venugopal rao raman gogoi lalu prasad yadav akhilesh ravat urmila milind deora siddaramiah shivkumar dks sandeep ashok tanwar prakash jha",
          "arvindkejriwal nota delhivote aapsweepingdelhi manishsisodia alimuddinansari raghunathkumar satyendrakumar harmohandhawan pankajgupta dilippandey atishimarlena brijeshgoyal gugansingh balbirjakhar raghavchadha prithviraj krishanagarwal naveenjaihind pradeeppadgaonkar elvisgomes petermasih jorasingh tejpalsingh sadhusingh bhagwantmann shwetasharma yogeshdahiya",
          "mayawati bsp"]

documents = df["processed_tweet"]
party = []
for i in documents:
    l = len(i.split())
    freq_bjp = 0
    freq_cong = 0
    freq_aap = 0
    freq_bsp = 0
    for j in compare[0].split():
        freq_bjp+=i.count(j.strip())
    for j in compare[1].split():
        freq_cong+=i.count(j.strip())
    for j in compare[2].split():
        freq_aap+=i.count(j.strip())
    for j in compare[3].split():
        freq_bsp+=i.count(j.strip())
    if freq_bjp > freq_cong and freq_bjp > freq_aap and freq_bjp > freq_bsp:
        party.append("BJP")
    elif freq_cong > freq_bjp and freq_cong > freq_aap and freq_cong > freq_bsp:
        party.append("Congress")
    elif freq_aap > freq_bjp and freq_aap > freq_cong and freq_aap > freq_bsp:
        party.append("AAP")
    elif freq_bsp > freq_bjp and freq_bsp > freq_cong and freq_bsp >
```

```
freq_aap:
        party.append("BSP")
    else:
        party.append("Other")


party.count("Other")
party.count("BJP")
party.count("Congress")
party.count("AAP")
party.count("BSP")

plt.rcParams["figure.figsize"] = [6,6]
labels = ["BJP","Congress", "AAP", "BSP", "Other"]
count = [party.count("BJP"),party.count("Congress"), party.count("AAP"),
party.count("BSP"), party.count("Other")]
patches, texts,autotexts = plt.pie(count,labels=labels, explode =
explode,colors = ["orange","lightgreen","lightblue", "purple",
"lightgrey"], startangle=90, autopct='%1.1f%%',shadow = False,
pctdistance=0.85)
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie
plt.axis('equal')
plt.tight_layout()
plt.show()
```

WordCloud

```
text = " ".join(tweet for tweet in df.processed_tweet)

#wordcloud = WordCloud().generate(text)

# Create stopword list:
stopwords = set(STOPWORDS)

# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white",
max_words=500, width=400, height=400).generate(text)

# Display the generated image:
# the matplotlib way:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```