COMPUTATIONAL METHODS FOR TWEET SUMMARIZATION AND EMOTION EXTRACTION

A Thesis Presented to

the Faculty of the Department of Computer Science,

College of Natural Sciences and Mathematics, University of Houston

---

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Chair of Committee: Dr. Christoph F. Eick

Committee Member: Dr. Weidong Shi

Committee Member: Dr. Amaury Lendasse

By

Anjana Kummari

May 2020

Acknowledgement

I would like to express my gratitude to my thesis advisor Dr. Christoph F. Eick for the useful comments, remarks and engagement throughout the learning process of this master thesis. I would also like to thank the incredible team at the UH Data Analysis and Intelligent Systems Laboratory for their continual insights and motivation for this thesis. I am grateful to God and thankful to my parents, brother and dear friends for their continued support in my life.

Abstract

The process of gathering insights from social media has gained significant importance in the last decade. Since social media data is growing larger and larger, frameworks that can analyze social media content automatically are of critical importance. Twitter is a micro-blog service that generates a massive amount of textual content every day. Throughout our research, we concentrate on using Twitter for the task of sentiment analysis, the most popular micro-blogging site. We demonstrate how to compile a corpus automatically for purposes of sentiment analysis and opinion mining. Sentiment analysis classifies texts based on the sentimental orientation of opinions and emotions they contain. In this project, we are interested in evaluating popular sentiment analysis tools that automatically determine emotions in tweets and to develop computational methods that summarize the content of a large set of tweets. For the comparison of sentiment analysis tools, we created different benchmarks of manually annotated tweet datasets, and then evaluated the tools using these benchmarks. We also addressed some of the most popular sentiment analysis challenges. As far as summarization of tweets is concerned, we designed and developed algorithms that extract keywords and key sentences as a summary for a set of tweets. Finally, we developed a tool that creates a distance matrix for a set of tweets relying on the popular TF-IDF framework.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Micro-blogging websites have become a source of diverse information [1]. People from around the world would share their views on various subjects, debate current events and review their everyday products on these websites. Since micro-blogging websites are easy-to-use platforms, they have gained popularity among every age group. Twitter is one such platform that has seen tremendous growth over the years [2]. Twitter reports 330 million active users per month (as of Q1 2019). More than 40 percent of these people use the service regularly (Twitter, 2019) [3]. Twitter has a global user base, with 38% of Twitter users between 18 and 29 years of age and 26% of its users between 30 and 49 years of age [3]. Twitter, which began as an SMS-based platform, has a 140-character limit for each tweet. Nonetheless, as Twitter developed into a web-based platform, it held the character limit primarily because it fits with Twitter's brand.

Twitter broadcasts more than 450 million messages each day, providing a large

amount of information for every industry, including entertainment, sports, health, and business. Twitter data is the world's most comprehensive source of online public communication [4]. It is also very accessible; it is easy to use for both sharing and collecting information. All of the listed features mentioned above make Twitter the best platform to collect real-time and up-to-date data to evaluate and perform any analysis of real-life situations.

The information available from social networks is useful for analyzing user opinions. Some potential uses of the information include: measuring the feedback on a recently released product, looking at the response to a policy change, or evaluating the enjoyment of an ongoing event. The amount of content generated by users is too large to be analyzed manually, as sifting through such data is tedious and time-consuming. Sentiment Analysis is a relatively new field that automatically extracts emotions and opinions from text documents, such as tweets. An example of positive sentiment is "The weather is beautiful today," whereas "the movie was terrible" is an example of negative sentiment. Objective texts, such as "Facebook acquired WhatsApp and Instagram," do not convey any emotions. Topics like subjectivity and sentiments are at the core of natural language processing (NLP) and data mining research so that social network data can be leveraged to provide a deeper understanding of user opinions, experiences, and evaluations.

Studies have been performed on automated emotion extraction from texts. Pang and Lee [5], for example, used the movie review domains to experiment with

machine learning techniques in classifying sentiments. Using Support Vector Machines (SVM) and the unigram model, they achieved high accuracy in classifying the sentiments. However, because the output of the classification of sentiments is dependent on the meaning of texts, these approaches experience difficulty in determining the sentiment if the polarized words with contrasting sentiments are present in the text. Natural Language Processing (NLP) techniques have been employed to alleviate this problem. The principal tasks used to complete NLP techniques are syntactic analysis and semantic analysis. Syntax refers to the way words are arranged in a sentence to allow it to make grammatical sense. Semantics refers to the meaning expressed by a text. NLP allows the use of computer algorithms to understand the context and interpretation of words and the form of sentences.

This thesis focuses on computational tools for tweet analysis and concentrates on two different research themes. First, popular sentiment analysis tools are used to evaluate tweets. We produced a benchmark consisting of five tweet datasets that are manually annotated with a polarity ranking. Evaluation measures are introduced to compare the tool polarity score with the ground truth. Finally, the five sentiment analysis tools are evaluated and compared against the benchmarks datasets. In the second part of the study, we developed three different tweet summarization techniques that serve the following purposes:

1. To identify important keywords in a set of tweets

2. To identify important key sentences in a set of tweets

3. Computational methods obtaining distance metrics for a set of tweets

All of these methods are developed to facilitate the analysis and summarization of a large number of tweets, such as millions of tweets about Hurricane Harvey.

This thesis is organized as follows: In Chapter 2, we discuss the background and related work. Moreover, we describe Twitter data analytics in detail. In Chapter 3, we explain the data collection and summary statistics for the benchmark datasets. In Chapter 4, we provide a brief introduction to the sentiment analysis tools and methodologies. In Chapter 5, we describe the topic summarization techniques that we developed. In Chapter 6, we introduce and analyze our experimental setup for a comparison of the tools of sentiment analysis and present the results of the experiment. We also demonstrate the application of the summarization techniques. Finally, in Chapter 8 we provide a conclusion and discuss future work.

Chapter 2

Background and Related Work

This chapter presents surveys conducted on current methods for sentiment analysis. Then, we explore current approaches to information retrieval. Finally, we present related work from other researchers and scientists in the field.

2.1 Background

Text processing focuses on parsing texts to extract machine-readable information from them. Text analysis aims to generate structured data from the free text material. In this section, we discuss the two most common text analytics methodologies: sentiment analysis and information retrieval.

2.1.1 Sentiment Analysis

Sentiment analysis is the method of deciding whether a piece of text is written positively, negatively, or neutrally. A sentiment analysis program for text analysis

incorporates natural language processing (NLP) and machine learning techniques to assign sentiment scores within a sentence or a phrase. Sentiment analysis helps large-scale data analysts to gauge public sentiment, perform complex market research, determine product credibility, and understand client experience. Additionally, data analytics companies also incorporate third party sentiment analysis APIs into their customer experience management, social media tracking, or workplace analytics framework to provide their customers with valuable insights.

Existing methods for sentiment analysis can be grouped into three major categories: Rule-based techniques, statistical methods, and hybrid approaches [6]. A rule-based framework is used to store and manipulate knowledge in a useful manner to interpret the information. The term ”rule-based system” is applied to systems that contain collections of human-crafted rules [7]. Statistical approaches leverage elements from machine learning such as latent semantic analysis, support vector machines(SVM), bag of words, and deep learning. Hybrid methods use both machine learning and knowledge-based systems to identify meanings that are subtly articulated. Fig. 2.1 shows a very high-level algorithm for sentiment analysis. In this section, we discuss the subcategories of sentiment analysis in more detail.

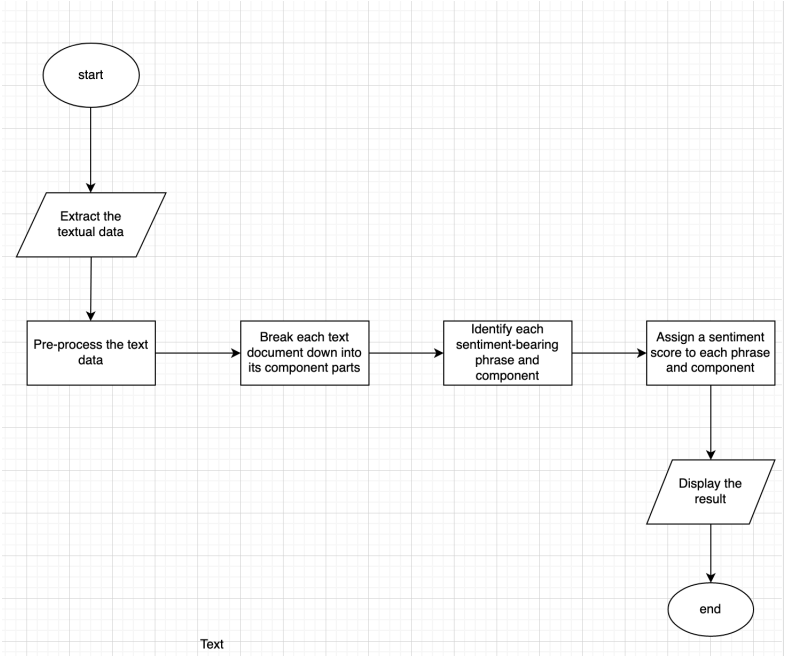

Figure 2.1: High-level flowchart for sentiment analysis

2.1.1.1 Rule-based Models for Sentiment Analysis

Rule-based sentiment analysis refers to studies carried out by language experts. The rule-based approach consists of three main phases: learning the rule for the extraction of product features, extracting opinion sentences, and identifying opinion orientation. The rule-based methodology is more intuitive and can be applied quickly, as opposed to algorithms based on machine learning. The downside, however, is that it necessitates much groundwork performed by humans, to create the set of rules. For each model, the domain expert has to draw up a set of rules, and these rules would also have to be changed each time the input changes. This, over time, can become very tedious and time-consuming. Figure 2.2 shows a flow chart for the rule-based models' algorithm.

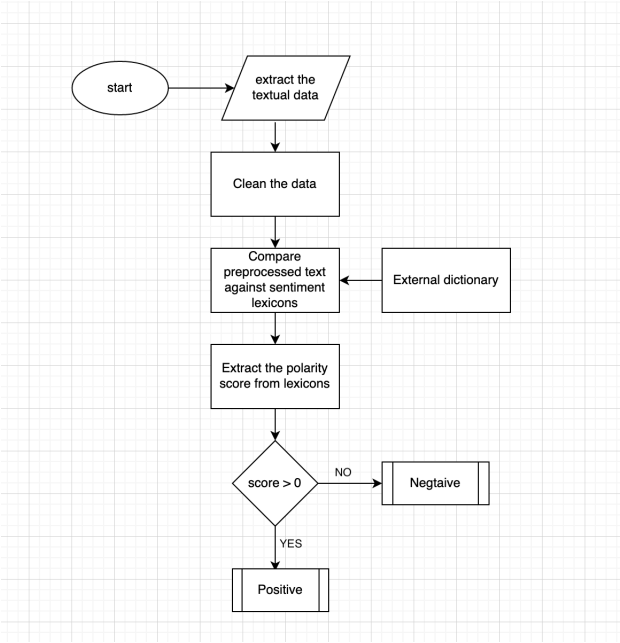

Figure 2.2: Flowchart for Rule-based sentiment analysis

The algorithm performs as follows: the documents are divided into words, known as word tokens. The word tokens match the words within the dictionaries. These dictionaries contain both positive and negative words and are created manually. Then, based on the polarized words, the score is calculated for the model. The set of rules for calculating the score is as follows:

1. Store the count of positive words into p_count
2. Store the count of negative words into n_count
3. Compute score = p_count - n_count
4. If score is greater than 0 then sentiment = positive; otherwise sentiment = negative

2.1.1.2 Machine Learning Models for Sentiment Analysis

Machine learning(ML) usually approaches sentiment analysis as a classification or prediction problem in which a model is learned and takes text as an input and returns a category, e.g., positive, negative, or neutral, or a number. The most popular machine learning models for sentiment analysis include Naïve Bayes classifiers, regression models, SVM, or neural networks. These models can be further enhanced by training, not only on individual tokens, but also on bigrams or trigrams. This enhanced training helps the classifier to pick up on negations and short sentences, which could hold sentimental information that the individual tokens do not have. As mentioned earlier, sentiment analysis is a classic case of the classification problem. The steps that would be followed in any ML model are:

1. Preprocess the text: lower case conversion, punctuation removal, word tokenization, POS tagging, and lemmatization

2. Feature extraction, in this case, the set of unique words that contribute towards sentiment score

3. Class labels for the training data

4. Split the data into training and testing data, typically 80% and 20% respectively

5. Train the model

2.1.2 Information Retrieval in Texts

The information retrieval framework is an algorithm network that facilitates the search of relevant data/documents according to user requirements. With internet technology rapidly growing, massive quantities of data are now accessible online. Information retrieval (IR) is thus becoming a critical area of study [8]. Most web documents are created as unstructured or semi-structured text. Traditional text data IR, including text classification, text clustering, and text-based search engines, are often processed on keyword-based approaches.

There are three growing conventional applications for information retrieval: document search, topic classification/clustering, and content management. Most of these applications use statistical or machine learning methods such as Term Frequency Inverse Document Frequency (TF-IDF), SVM, K Nearest Neighbor (KNN), and neural networks to assist text analysis. Figure 2.3 shows the general applica-

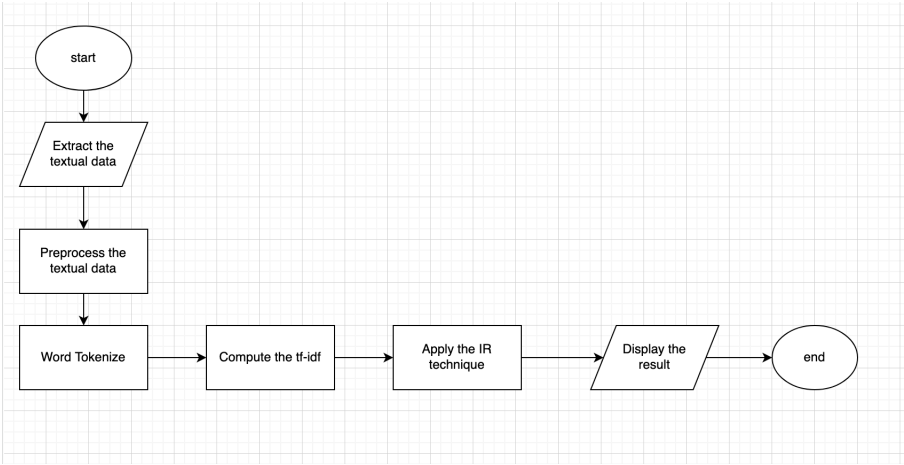

Figure 2.3: Flowchart for Information Retrieval techniques based on TF-IDF framework

tion of IR techniques. In this project, we worked with three information retrieval techniques: keyword extraction, extractive summary generation, and document similarity.

2.1.2.1 Text Summarization

Text summarization refers to the method of shortening long pieces of text. The goal is to establish a coherent and fluent overview with only the critical points outlined in the text. Automatic text summary generation is a common issue in machine learning and NLP. With such a large volume of data circulating in digital space, it is essential to create machine learning algorithms that can automatically shorten longer texts and provide accurate summaries. Text summarization can be done in two ways: An Extraction-based summary and an Abstract-based summary. The extraction-based text summary technique involves extracting key-phrases from the document and combining them to produce a summary. The abstraction technique includes paraphrasing and shortening sections of the text. In this project, we mainly focused on extraction-based summary techniques in which we extract keywords and key phrases from a set of tweets.

Example Text: Joseph and Mary rode on a donkey to attend the annual event in Jerusalem. In the city, Mary gave birth to a child named Jesus.
Extraction-based Summary: Joseph and Mary attend event Jerusalem. Mary birth Jesus.
Abstract Summary: Joseph and Mary came to Jerusalem where Jesus was born.

2.2 Relevant Work

In 1997 Kessler et al. [9] began researching sentiment analysis. The two more common approaches used to characterize the sentiments of text are lexical analysis and ML models. Lexical analysis models rely on dictionaries. In 2004 Hu et al. [10] used a lexicon-based approach to provide an opinion mining process. They extracted reviews features and listed whether the opinions were positive or negative. The opinions could automatically be extracted using machine learning-based analysis, i.e., it helps the computer to learn without explicit programming [10]. Due to their adaptability and efficiency in extracting several features, machine learning models have gained popularity.

Only a limited amount of the current research into sentiment analysis has performed a comparison between approaches of sentiment analysis, usually with confined datasets [11]. As new methods appear and are only compared to one or, at most, two other methods, using various datasets of evaluation and experimental methodologies, it is difficult to determine whether a single method triumphs over the remaining methods or even in particular scenarios. Ahmed et al. [12] conducted a comparison of sentiment analysis methods on a Twitter dataset. Their work is very similar to our efforts in this project, except we focused on five tools, whereas they worked with twenty tools. Their results showed that the tools' performance differed considerably depending on the datasets. The best-performing tools, however, maintained consistency throughout datasets.

Both supervised and unsupervised methods were explored for the execution of keyword extraction. Wu et al. [13]; Zhao et al. [14]; Bellaachia and Al-Dhelaan [15] proposed many automated keyword/keyphrase extraction methods that are non-supervised. However, the TF-IDF remains a solid, unsupervised baseline across various methods (Hasan and Ng [16]). Several methods are suggested for extracting keywords. The MAUI toolkit-indexer toolkit is a more robust baseline for Automated Keyword Extraction [17]. The program extracts from a document a list of candidate keywords and trains a decision tree over a wide range of hand-crafted features, like TF-IDF, to predict the appropriate keywords on the training set. The toolkit, once trained, extracts a list of keyword candidates from a tweet and returns a ranked list of candidates. The words with highest scores are chosen as keywords. On the development set, the parameter k is maximized.

An array of research has been conducted in the field of tweet topic detection. Phuvipadawat et al. [18] provided an example of the tweet-oriented topic detection process. Their work suggested a system that uses a specific keyword to gather tweets and then analyze them in order to find out topics relevant to that keyword. According to this process, the tweets are first extracted using queries such as #tag and are then grouped based on the similarities calculated by TF-IDF. The purpose of the grouping is to identify all the significant topics within the keyword and cluster all the tweets that fall under that topic. Teixeira et al. [19] ranked words, multi-words, and word prefixes (fixed-length: 5 characters), by using various similarity tests and assessing the results obtained, as well as the agreement between

the evaluators. Of texts in Portuguese, English, and Czech, evaluators have been required to assess 25 top-ranked words for each of the six measures. The top 23 ranked terms extracted from both TF-IDF and Phi-square or any of their new variants display better results than those obtained from Rvar or any of its variants.

Chapter 3

Data Acquisition and Analysis

This chapter is dedicated to the description of datasets and the analysis of exploratory data. First, we discuss how tweets were obtained for this project. We are working on two tasks: evaluation of sentiment analysis tools and demonstrating and evaluating the tweet summary techniques. The requirements of both of these tasks are quite different, and we present tailored specific datasets for each of them. For this project, our main objective is to compare sentiment analysis tools, and we also develop Tweet summarization techniques. Datasets required for these two tasks have been collected from Twitter. We used the R library "rtweet" to extract data from the Twitter API, and the process is discussed in more detail in later sections. We divide the chapter into three sections; the first and second sections are dedicated to data acquisition, and finally, in the third section, we discuss the preparation of the data.

3.1 Twitter API

Twitter allows users to create a developer profile and, upon access, developers are provided with a set of keys and tokens. The keys and tokens are used to collect data, and they can also track the use of the data. Users can access the Twitter API from a variety of platforms, and in our study, we used R Studio to access the data.

The API comes in both free and paid versions, and the free edition is sufficient for most academic research. There are a few drawbacks to the free edition, such as a tweet cap of 18,000 tweets a day and inability to retrieve comments, likes, and geolocations from tweets, but, even with the limitations, most analyses can still be performed. Using the free version of the API, we extracted more than 150,000 raw tweets. In the following sections, we explain the extraction procedure and data preparation

3.2 Twitter Data

This section includes a discussion of the methods used to extract Twitter data and the objective is to understand the Twitter API and the tweet datasets we used for the analyses. Data extraction is the most critical step before any analysis, as building up five individual datasets requires a lot of groundwork.

3.2.1 Raw Data

Depending on the intended use, different types of tweet data can be extracted. Tweets can be either extracted from a specific timeline, or from live tweets that day, called streaming tweets. The extracted tweets contain the following 90 columns. During the preprocessing steps, most columns are removed if they do not contribute to the analysis. Fig 3.1 displays the 90 columns that the Twitter API allows users to extract. Some of the columns that are significant are

```
> colnames(rt)
 [1] "user_id"                "status_id"              "created_at"             "screen_name"
 [5] "text"                   "source"                 "display_text_width"     "reply_to_status_id"
 [9] "reply_to_user_id"       "reply_to_screen_name"   "is_quote"               "is_retweet"
[13] "favorite_count"         "retweet_count"          "quote_count"            "reply_count"
[17] "hashtags"               "symbols"                "urls_url"               "urls_t.co"
[21] "urls_expanded_url"      "media_url"              "media_t.co"             "media_expanded_url"
[25] "media_type"             "ext_media_url"          "ext_media_t.co"         "ext_media_expanded_url"
[29] "ext_media_type"         "mentions_user_id"       "mentions_screen_name"   "lang"
[33] "quoted_status_id"       "quoted_text"            "quoted_created_at"      "quoted_source"
[37] "quoted_favorite_count"  "quoted_retweet_count"   "quoted_user_id"         "quoted_screen_name"
[41] "quoted_name"            "quoted_followers_count" "quoted_friends_count"   "quoted_statuses_count"
[45] "quoted_location"        "quoted_description"     "quoted_verified"        "retweet_status_id"
[49] "retweet_text"           "retweet_created_at"     "retweet_source"         "retweet_favorite_count"
[53] "retweet_retweet_count"  "retweet_user_id"        "retweet_screen_name"    "retweet_name"
[57] "retweet_followers_count" "retweet_friends_count" "retweet_statuses_count" "retweet_location"
[61] "retweet_description"    "retweet_verified"       "place_url"              "place_name"
[65] "place_full_name"        "place_type"             "country"                "country_code"
[69] "geo_coords"             "coords_coords"          "bbox_coords"            "status_url"
[73] "name"                   "location"               "description"            "url"
[77] "protected"              "followers_count"        "friends_count"          "listed_count"
[81] "statuses_count"         "favourites_count"       "account_created_at"     "verified"
[85] "profile_url"            "profile_expanded_url"   "account_lang"           "profile_banner_url"
[89] "profile_background_url" "profile_image_url"
>
```

Figure 3.1: Tweet Columns

1. **geo_coords** Geo location of the user
2. **is_retweet** Says if the text is a retweet
3. **reply_count** is the count of total replies to that particular tweet
4. **media_type** says what media format is attached to the tweet (image, video, gif, etc.,)

3.2.1.1 Data Arguments

The R-library "rtweet," which allows users to filter tweets as required by passing arguments, was used to collect data. Some of the arguments used in the study are

1. **n** = 17000
2. **include_rts** = FALSE
3. **type** = "recent"
4. **lang** = "en"
5. **geocode** = lookup_coords("USA"))

We extracted 17,000 recent tweets in English from the USA and did not include retweets. One other argument, "#keyword," was used to extract specific tweets containing the keyword.

3.2.2 Filtered Data

Though Twitter's API provides users with 90 columns, not all of the columns are useful for the analysis, and keeping them would take up extra space. Since one of the project's objectives is to understand the relevance of hashtags in tweets, we filtered the raw dataset and retained tweets with hashtags. After filtering more than 150,000 tweets, we found that roughly 10 - 12% of each retrieval (17,000 tweets) contains hashtags. Only the relevant data columns were stored, and the datasets were downloaded as CSV files.

3.2.2.1 Data Entities

We extracted tweets into datasets with these five variables. We refer to the datasets as tweet datasets from now on. We have presented example tweets with the five variables in Table 3.1.

1. **User_id** is twitter user information. It is not unique identification id under the factors that the same user could tweet multiple times
2. **Text**, the tweet text
3. **Created_at**, the time at which the user posted the tweet.
4. **Hashtags** contain a list of all the hashtags used in that particular tweet
5. **Location** has city and/or state information of the user.

Table 3.1: An example tweet dataset

user_id	text	created_at	hashtags	location
93096074	I already own a face mask. Here is some other directive from the @CDCgov on the #coronavirus"	2020-02-25 19:38:56 UTC	"coronavirus"	NYC
44485855	@KoreanAir_KE How is it a voluntary cancellation when the CDC is saying to avoid non-essential travel and all of the other airlines are waiving cancellation fees? #koreanair #coronavirus #travel #cdc"	2020-02-25 19:38:53 UTC	"koreanair" "coronavirus" "travel" "cdc"	Silicon Valley

3.3 Data Preparation

The tweet datasets extracted from the API described earlier were not directly used for the analysis. The raw datasets contain a varied range of tweets, and not all of them would be beneficial for the analysis. We created filters to tailor the datasets to specific tasks. Because we were working toward two very different tasks, we generated two specific datasets for each task and one common dataset that was used for both tasks. The process of obtaining and filtering datasets is been explained in this section.

3.3.1 Common Dataset

We generated multiple raw datasets from the Twitter API over a duration of six months. Of the 18,000 tweets generated each time, only 10% of them had hashtags. Therefore, we had to obtain multiple datasets to achieve a reasonable amount of data. We then performed sentiment analysis on these tweets using python-NLTK. NLTK provided us with a polarity score within the range of [-1, 1]. We created the following filter with respect to the polarity scores.

1. Tweets_score $>$ 0.5
2. Tweets_score $<$ -0.5
3. Tweets_score = 0

There were many tweets with a score of 0, so we randomly chose 100 neutral tweets. Applying these filters returned a dataset with 400 tweets, of which 100

tweets were neutral, 120 tweets were positive, and 180 tweets were negative.

Although comparison of sentiment analysis tools and evaluation of topic summarization tools have different requirements, this dataset serves well for both of them. Because it covers positive, negative, and neutral emotions, it could be used for sentiment analysis, and since the data has been generated over a duration of time, it could be used to determine the relevant topics among the data.

3.3.2 Comparison of Sentiment Analysis Tools - Datasets

Two specific and challenging datasets, "Positive Dataset" and "Negative Dataset," have been designed for the evaluation of sentiment analysis tools. The raw tweet datasets were extracted during January and February in 2020. During this time, there were three trending hashtags on Twitter, and we downloaded all tweets with that hashtag from the Twitter API, as discussed in Chapter 5.

1. #grammys, Music Award Show (1667 tweets)
2. #BTS, South Korean Boy Band (1970 tweets)
3. #impeachment, President Impeachment (300 tweets)

3.3.2.1 Positive Dataset

After combining all the raw tweets, we had a total of 3937 tweets. All the raw tweets were scored by python-NLTK and TextBlob. For the positive dataset, we filtered the tweets that were scored more than 0.7 by NLTK and TextBlob. A total

of 70 tweets were registered in all three datasets, with a score of more than 0.7 by both tools. After the extremely positive tweets were screened, we performed a manual assessment of the 70 tweets. This was a very challenging dataset because it included only tweets that fit one form of emotion (positive) and showed how each tool scores tweets.

3.3.2.2 Negative Dataset

The negative dataset consisted of filtered tweets that were scored less than -0.7 by NLTK and TextBlob. Once applied, these filters resulted in 120 negative tweets from all 3 datasets. After the data was filtered, we conducted the manual assessment. This was even more challenging than analyzing the positive dataset, as there were many tweets that expressed sarcasm. Very few tools performed well at detecting sarcasm (to be discussed in later chapters). Many tweets that scored negative by the tools were corrected to actually score positive during the manual assessment. This highlights one objective of creating complex datasets: to gauge the accuracy of the tools.

3.3.3 Tweet Summarization Datasets

The motivation behind creating multiple datasets to test different tweet summarization approaches was to understand the impact of randomness. Randomness, in this project, refers to how varied the datasets were. If there was a wide range of topics being discussed within the dataset, it would mean that the data is very random. As randomness increases, it becomes more difficult to summarize the

data. The common dataset contained random tweets collected over a duration of time (2 months). However, there were only 400 tweets in that dataset, and they had been assigned scores, which contributes towards the sentiment aspect of the summarization. The two datasets designed for Tweet summarization techniques are going to be called "Randomized dataset" and "Topic-specific" dataset from now on.

3.3.3.1 Randomized Data

A set of 2200 tweets were downloaded from Mar 2019 to January 2020 from the API. None of them were scored, and they focused on a wide range of topics, as they were all extracted over 10 months. This dataset was created to understand how the threshold varies as the randomness of the data increases.

3.3.3.2 Topic Specific Data

We used the "#impeach" tag for analysis, as this was one of the most commonly discussed topics. We used the raw data that had not been manually annotated, containing approximately 250 tweets.

Chapter 4

Sentiment Analysis Tools and Methodologies

Various tools for sentiment analysis are available on the market. They are used to analyze reviews, feedback, and reports. This saves time and gives customers high-level insight into what to focus on and what improvements to make. Sentiment analysis tools classify pieces of writing as positive, neutral, or negative and represent the classification either numerically or graphically. Some tools visualize their results and present feelings as graphs, while some tools return polarity scores to express emotion. Sentiments are typically quantified with either a positive or a negative value. The overall sentiment is often inferred as positive, neutral, or negative by the sign of the polarity score. Specifically, we looked for tools that present an emotion score within the range of [-1, +1].

The primary motivation behind the evaluation of sentiment analysis tools is to

understand how the tools perform in comparison to manual assessments performed by humans. We designed one medium-difficulty and two challenging datasets, which were introduced and discussed in Chapter 3, to assess the performance of the sentiment analysis tools. We divide this chapter into three parts: the first section describes the development of dataset benchmarks, the second section presents and explores in detail the tools of sentiment analysis that we have chosen for this research, and the third section explains the normalization of scores.

4.1 Manual Assessment

We manually created three benchmarks of tweets that serve as the ground truth to evaluate sentiment analysis tools. While many tools perform sentiment analysis, there are specific complex text categories that the tools do not work well with. A few of these domains are sarcasm, irony, negations, jokes, and exaggeration. Our goal was to build a benchmark that covers these complex domains, as well. We would use these benchmarks to analyze how accurately these tools work with medium-difficulty datasets and with challenging datasets that include these complex domains. For sentiment classification, the essential task is to exploit the words that contribute towards the polarity score, e.g., adjectives, verbs, and adverbs. Different studies suggest the importance of various forms of adverbs and adjectives in sentiment classification tasks [20]. Sentiment analysis research has focused on using adjectives to indicate the semantic orientation of text (Hatzivassiloglou and McKeown [21]; Wiebe [22]; Hu and Liu [8]). Adjectives and adverbs contribute the most toward sentiment scoring. Adjectives describe nouns and

pronouns, while adverbs emphasize adjectives and verbs. In our case, we were looking for a combination of adjectives and adverbs, which impacts the sentiment score. Hyperbolic sentences depict exaggerated emotions. Usually, emotions are directed towards something (the subject), and subjectivity differentiates the positive/negative texts from neutral texts, as they lack subjectivity.

This section is further divided into two subsections: in the first subsection, we discuss the limitations of sentiment analysis tools, and in the second subsection, we explain the scoring mechanism for the manual assessment.

4.1.1 Complex Text Classifications

As discussed earlier, sentiment analysis has its limitations. Here, we list some domains that the sentiment analysis tools often find challenging and misclassify:

1. **Hyperbolic sentences** are extreme exaggerations used for emphasis or effect. It is figurative language and should not be taken literally. **Example**: "I am dying of laughter."

2. **Superlative sentences** are sentences with adjectives that are used to describe an object that is at the highest or lowest limit of a quality. It is used when the subject is compared to a group of objects. **Example**: "This is the best movie I have ever watched!"

3. **Negations** are sentences which have the word "not" in them. They could be positive or negative but most of the time, they are misclassified as negative.

Example: "I don't hate it, actually."

4. **Sarcasm/irony** refers to the use of words that mean the opposite of what is intended. These are often used in a negative manner. **Example**: "I am absolutely enjoying the 4-hour wait time."

We included all four domains in our challenging datasets. The positive dataset contained many sarcastic tweets that were classified as positive by most of the tools, while they were actually negative. Superlative sentences were included in all of the datasets. Hyperbolic sentences and negations were included in the negative dataset, while not all of them were actually negative.

4.1.2 Scoring Mechanism: 7-Class Method

Very high-level classifications of the sentiments are: Positive (+1), Negative (-1), and Neutral (0). Based on this high-level classification, we built a 7-class method for our benchmark. Table 4.1 shows the seven classes of sentiments. Benchmark creation is a 3-phase process:

1. Obtain the tweet datasets.

2. Manually score the tweets using the 7-class method.

3. Evaluate sentiment analysis tools.

Obtaining the datasets was discussed in Chapter 3. We discuss the manual scoring in this section, and we describe evaluation of sentiment analysis tools in detail in Chapter 6.

Table 4.1: 7-Class Method

Score	Sentiment
+1	Very Positive
+0.8	Fairly Positive
+0.5	Positive
0	Neutral
-0.5	Negative
-0.8	Fairly Negative
-1	Very Negative

In the following subsections, we explain the seven classes and all of the tweet categories that fall under each of these classes

4.1.2.1 Very Positive: +1 / Very Negative: -1

+1/-1 denotes that the text has a very extreme representation of the sentiments. The categories +1/-1 are typically associated with exaggerated versions of adjectives and adverbs. In particular, we looked for the following attributes in the text:

1. **Superlative Adjectives**: Sentences with superlative adjectives are usually extreme cases of positive or negative. **Example**: ”Best,” ”worst,” ”tallest,” etc.,

2. **Hyperbolic sentences** have exaggerated emotions present in the text. **Example**: ”He’s as skinny as a toothpick.”

3. **Superlative Adverbs**: Adverbs are used to describe adjectives or verbs. In

the cases where the adjectives or verbs are with a leading superlative adverb, the text is considered very positive or very negative. **Example**: "I am really happy."

Table 4.2: Tweet Examples for Very Positive/Negative

Text	**Score**
It was a great night of concert. Absolutely amazing! Keep Jazz alive!	+1
So sad. What a tragedy! RIP Kobe, his daughter, and the 9 others that lost their lives.	-1
I like this movie better than Avengers	+0.8
It keeps getting sadder.	-0.8
Hope everyone is having a nice Monday!	+0.5
The weather is gloomy today.	-0.5
Learn more about remote classes here!	0
Is anybody home today?	0

Table 4.2 gives examples of positive and negative tweets. In the first example, "absolutely amazing" is an adjective with a leading superlative adverb, and hence it is classified as very positive. In the second example, along with an adjective with a leading adverb, there is also a superlative adjective, and thus it is classified as very negative.

4.1.2.2 Fairly Positive: +0.8 / Fairly Negative: -0.8

+ 0.8/-0.8 denotes a moderate degree of sentiment in the text. We looked for comparative adjectives and adverbial versions. We searched through the text for

the following factors:

1. **Comparative Adjectives** are usually used to compare a subject with another item. **Example**: ”The iPhone’s camera works better.”

2. **Comparative Adverb**: If we want to compare one verb action with another, we can use a comparative adverb. This puts more emphasis on the sentiment. **Example** ”A ran fast, but B came first because B ran faster.”

Table 4.2 gives examples of relatively positive and negative tweets. Both instances have comparative forms of adjectives, and example 1 has a comparative adverb, as well.

4.1.2.3 Positive: +0.5 / Negative: -0.5

+0.5/-0.5 denotes the text having a given opinion, which can contribute to the polarity score. We searched through the text for the following factors:

1. **Adjectives/Adverbs**: The presence of adjectives or adverbs within the text that represent emotions. **Example**: ”I like iPhones.”

2. **Emotions**: The presence of definite but not overbearing emotions directed towards a subject. **Example**: ”This movie makes me sad.”

Table 4.2 shows examples of positive and negative tweets. Both of the examples have first-degree adjectives directed toward a subject.

4.1.2.4 Neutral: 0

A lack of subject matter characterizes neutral texts. People tend to tweet their daily routines online, which could be very generic and could lack the presence of emotions. Although these tweets do not have anything to contribute information-wise, they can be used in datasets to create a balance among the other emotions and to understand how the tools score neutral texts. Some characteristics of neutral texts are:

1. There are no adjectives that contribute towards an emotion.
2. No emotions are explicitly mentioned within the text.
3. There is no definitive subject in the text.

As discussed in Chapter 3, we had already scored these using NLTK before manually annotating the benchmark datasets. This simplified the process, as we did not have to annotate thousands of tweets. Most of the tweets fell into the seven categories that we described; thus, it was not very challenging. In the case of tweets that did not fall into these seven categories, we depended solely on polarized words that defined emotions and gave the text meaning. It took nearly two weeks to build each dataset.

4.2 Survey of the Compared Sentiment Analysis Tools

We selected a total of five tools that present their results within the range of [-1,+1]. We define each of the tools in the following five subsections and display an

example tweet and its polarity score from the tool.

4.2.1 Natural Language Toolkit

NLTK is also one of the most common Python libraries used for text analytics. It was first developed in 2001 as part of a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania [23]. We used NLTK.sentiment.vader to extract the sentiment score of the tweets. It has functions like polarity_scores and sentiment_valence, which represent the sentiment in various ways. For this project, we used polarity scores, which return the compound sentiment score within the range [-1.0, 1.0]. The polarity score is represented as pos, neg, neu, and compound, where each document has these scores. We have considered only the compound score for this analysis, as it is the aggregate score. Table 4.3 shows the compound score for the tweet example.

Table 4.3: Tweet Example Scores for Sentiment Analysis Tools

Text	**NLTK**	**TextBlob**	**SA**	**SentiR**	**Lexicon**
Coursera is an amazing platform to learn data science.	0.789	0.672	0.721	0.571	+3

4.2.2 TextBlob

TextBlob, also known as TB, is one of the Python libraries that offers API access to perform fundamental NLP tasks. The TextBlob Naive Bayes Analyzer is based on

the Stanford NLTK and has a rather simple and easy-to-use interface when compared to NLTK. Polarity in sentiment analysis represents sentiment orientation and, in simpler terms, classifies them as positive, neutral, or negative. The polarity score [24] that is calculated using the naive Bayes model is a float value within the [-1.0, 1.0] range. TB sentiment also identifies if the documents are subjective or objective. Objective statements are hard facts, whereas subjective statements express opinions. The subjectivity [24] is also a float value within the [0.0, 1.0] range, where 0.0 represents that the document is very objective, and 1.0 represents that the document is very subjective. In this study, we ignored the subjectivity score and only used the polarity score.

TextBlob uses a Naive Bayes model to predict the polarity. The naive Bayes algorithm converts word scores to probabilities to make predictions. TextBlob assigns three scores, polarity, subjectivity, and intensity, to each word. When calculating the sentiment for a single word, TextBlob uses a sophisticated technique known as "averaging." With these assigned scores for each word, TB calculates the overall polarity score of a text. Table 4.3 shows the TB score for the given example.

4.2.3 Sentiment Analysis - R

The Sentiment Analysis Package in R uses dictionary-based sentiment analysis, which is performed on textual contents in R. This execution uses different existing lexicons, which are defined as the vocabularies of a language. Additionally, the Sentiment Analysis package allows one to generate tailored dictionaries. These

are customized to a specific domain and improve prediction performance compared to pure dictionaries [25]. Dictionaries are used to determine the polarity score of the text, as they have a predefined list of positive and negative words. The Sentiment Analysis Package comes with four default dictionaries, which are the Harvard-IV dictionary, Henry's Financial dictionary [26], Loughran-McDonald Financial dictionary [27], and QDAP dictionary from the package "qdapDictionaries." In this study, we used the QDAP dictionary.

Sentiment Analysis works very similarly to how the lexicon-based model works. It relies on dictionaries, and it calculates the sentiment score based on the positive and negative word count in the text.

4.2.4 SentimentR

The SentimentR package offered by R is designed to compute the polarity score of a text at the sentence level. The sentence level relates to grammar, content, and punctuation. SentimentR works with factors like valence shifters [28], which are words that alter or intensify the meaning of polarized words. Some of the valence shifters are:

1. **Negators** are sentences containing "not" in them. Example: "I **don't** like it."
2. **Amplifiers** emphasize polarized words more by using adverbs for positive sentences. Example: "I **really** like it."
3. **De-Amplifiers** emphasize the polarized words more by using adverbs for

negative sentences. Example: I **hardly** like it.

These valence shifters are limitations of sentiment analysis, and SentimentR handles these domains. Hence, we picked this tool for comparative analysis. We show a sample text with a polarity score computed using sentimentR in Table 4.3.

The algorithm that assigns value to the polarity of each sentence first utilizes a sentiment dictionary [29] to tag polarized words. Each paragraph composed of sentences is broken into element sentences, and each sentence is broken into an ordered bag of words. The words in each sentence are searched and compared to a dictionary of polarized words (Rinker's augmented Hu and Liu [10] dictionaries in the lexicon package). Positive and negative words are tagged with +1 and 1, respectively. Valence shifters assign an added weight to these scores, and the final polarity score is computed based on the polarized words and valence shifters.

4.2.5 Lexicon Based Model

Lexicon-based models use the placements of words and linguistic methods to discover the patterns to determine the polarity score (Hu and Liu [10]; Kim and Hovy [27]) and dictionaries of words annotated with the word's polarity, i.e., positive or negative. Dictionaries for lexicon-based approaches can be created manually (Toni [30]) or automatically, using seed words to expand the list of words (Hatzivassiloglou and McKeown [31]). For our project, we modified the existing dictionaries for positive and negative words. We primarily worked with social media

data, and the language used on social media is quite different from traditional dictionaries. Hence, we extended the existing dictionaries by adding Twitter-specific words.

A fundamental implementation of the lexicon model with modified dictionaries is that this model does not consider context; each word is considered literally, and then it calculates the score. Algorithm 4.2.5.1 gives a high-level pseudo code for the Lexicon-based model. This model might have disadvantages when negations and sarcasm are used in a text. The results are discussed in Chapter 6. Although Lexicon-based models are not very complicated, they are a fundamental approach to sentiment analysis.

4.2.5.1 Algorithm

Algorithm 1 Lexicon's Based Model

```
1:  procedure LEXICON_SCORE(text)                    ▷ The lexicon score for text
2:      pos_count = 0, neg_count = 0
3:      preprocess(text)                             ▷ Preprocess the text
4:      for each positive word in the dictionary do
5:          pos_count += 1
6:      end for
7:      for each negative word in the dictionary do
8:          neg_count += 1
9:      end for
10:     SentenecScore = pos_count - neg_count
11:     return SentenceScore                         ▷ The score for text
12: end procedure
```

4.3 Normalizing Scores and Class Labels

In this section, we discussed how the scores have been normalized, and then we discussed the class labels. Although all the tools' scores were within the range of [-1, +1], it is not possible to assign class labels if all the tools do not have the same scoring scale. Class labels are needed to calculate classification error, which is one of the evaluation methods. However, we retained the raw scores as well for comparison purposes. We devised a normalization technique that converts scores

in [-1,1] into a 7-class scale that is followed by the manual benchmarks.

Table 4.4: Converting numerical scores into class labels

Tool Results	**Score**	**Classification**	**Class Label**
$>= 0.8$	+1	Very Positive	1
$0.6 < x < 0.8$	+0.8	Fairly Positive	2
$0.3 < x <= 0.6$	+0.5	Positive	3
$-0.3 <= x <= 0.3$	0	Neutral	4
$-0.6 <= x < -0.3$	-0.5	Negative	5
$-0.8 < x < -0.6$	-0.8	Fairly Negative	6
$<= -0.8$	-1	Very Negative	7

Table 4.4 summarizes how the numerical polarity scores of the tools we evaluated were converted into the seven categories of the 7-class method. All the raw scores provided by the tools are translated to the scale set out in Table 4.4. In the next chapter, we will introduce and discuss in detail the Tweet Summarization techniques that we have created.

Chapter 5

Topic Summarization Tools and Methodologies

This chapter centers on computational methods that create summaries for a given set of tweets. Section 5.1 discusses a keyword extraction algorithm, Section 5.2 introduces a key sentence generation algorithm, and Section 5.3 introduces similarity assessment techniques that produce a distance matrix for a set of tweets.

5.1 Keyword Extraction Algorithm

Keyword extraction is defined as the process of retrieving the most important words or expressions from a single document or set of documents. The primary motivation for keyword extraction is to understand a large set of text documents without needing to read them. Although there are many ways to do this, we proposed three methods to retrieve keywords:

- TF Approach
- TF-IDF Approach
- Hashtag Approach

TF and TF-IDF are a few of the most common approaches used in text analysis. Since the input datasets are tweets, we decided to introduce an algorithm that is specific to Twitter data, called the Hashtag Approach. The motivation behind this is to study the relevance of hashtags in summarization techniques.

5.1.1 Term Frequency Alogorithm (TF)

The Term Frequency algorithm converts documents into a set of words and computes the frequency of each of these words in the entire document. The input for this algorithm is a set of preprocessed tweets, and the output is a Python dictionary with words as keys and their frequency as values. The main goal is to find the most commonly appeared words.

$$TF(t) = \frac{WordCount(t)}{TotalCount} \tag{5.1}$$

where
WordCount(t) = Number of times the word t has appeared in the corpus and
TotalCount = Total number words in the corpus

Example:

1. Monday morning is lazy

2. Today is Monday

3. I feel lazy on Monday morning

After stopwords (words like are, is, etc.,) are removed, the results look as follows,

Table 5.1: Example TF scores

Word	**Frequency**
Monday	0.3
Morning	0.2
Lazy	0.2
Today	0.1
Feel	0.1
I	0.1

Table 5.1 shows the frequencies of the words for the given example. To identify the keywords, we can either choose the top 'k' frequency scores or define a threshold and return all of the words that have a frequency above that threshold. For this example, the top 3 keywords are "Monday," "morning," and "lazy."

5.1.2 Term Frequency Inverse Document Frequency Approach (TF-IDF)

Term Frequency Inverse Document Frequency (TF-IDF) is widely used for information retrieval and summarization. This algorithm assigns a weight to each

word, and these weights represent the importance of the word. The significance increases relative to the number of times a word shows up in the document yet is counterbalanced by the recurrence of the word in the corpus.

The TF-IDF weight of a word is a combination of terms: first, the standardized TF is computed; then, the subsequent term is the IDF, which calculates the logarithmic division of the total length of the corpus and the number of documents in the corpus where the word vector appears. TF computes the frequency of each word in the entire dataset, and IDF computes the number of documents that contain the word. When TF is computed, all words are given equal importance, and hence to weigh down the most frequent words that may contribute very little, IDF is computed. TF-IDF will be referred to in this chapter as TDF-weight.

$$TDF(t) = TF(t) * IDF(t) \tag{5.2}$$

$$IDF(t) = log_e(\frac{N}{DF(t)}) \tag{5.3}$$

where

TDF(t) is TF-IDF weight of the word t

TF(t) is the frequency of the word t

IDF(t) Inverse Document Frequency of the word t

N is the total number of documents in the corpus

DF(t) is the number of documents containing the word t

The input for this algorithm was a set of preprocessed tweets. The output was a Python dictionary with words as keys and their TDF-weights as values. We extracted the words with the highest TDF-weights, then we could either select the top 'k' keywords or define a threshold and choose the words that have a TDF-weight above the threshold.

Example

For the same example as 5.1.1, the TDF weights are shown in Table 5.2.

Table 5.2: Example TDF weights

Word	**TDF-weight**
Monday	0.51
Morning	0.315
Lazy	0.368
Today	0.2
Feel	0.2
I	0.04

If the top 3 keywords are to be chosen, then we have ”Monday,” ”morning,” and ”lazy” as the keywords.

5.1.3 Hashtag Frequency Algorithm

Since our datasets were primarily tweets, we decided to implement a tweet-specific approach. Twitter has a 140-character limit for each tweet, and therefore users use

features like hashtags to represent their content. Most of the time, hashtags are to indicate the topic the user is discussing. Here, we present some example tweets with hashtags for better understanding.

Example Tweets

1. We're all in this together. Be kind. **#COVID19 #COVID19OhioReady #CoronaVirus**

2. 48 states and D.C. have now **#CoronaVirus**. Alabama reported its first **#Covid19** case

3. Let's dispel the notion that "novel coronavirus is just like the flu". Sharing this striking comparison of **#Flu**, **#COVID19**, **#SARS** and **#MERS** made by @BioRender

For this approach, we computed the frequency of the hashtags. Twitter's API allows users to download hashtags, but only about 10% of tweets contain hashtags. However, all of the datasets we used throughout this project contain tweets with hashtags. The hashtag frequency (HF) of a tweet is calculated using the following formula:

$$HF(h) = \frac{HashtagCount(h)}{TotalCount} \tag{5.4}$$

where
HF = Hashtag Frequency (weight) of the hashtag h
HashtagCount = No. of times the hashtag h has appeared

TotalCount = Total number of hashtags in the entire dataset

The input for this algorithm is a set of hashtags, and the output is a Python dictionary with hashtags as keys and HF as values. This is a very straightforward approach; therefore, we computed the frequency of hashtags and returned the hashtags with the highest frequencies. The frequencies of the hashtags from the example tweets after preprocessing are shown in Table 5.3.

Table 5.3: Example HF scores

Hashtag	**Frequency**
COVID19	0.375
CoronaVirus	0.25
COVID19OhioReady	0.125
Flu	0.125
SARS	0.125
MERS	0.125

The top 2 keywords for the above example would be COVID19 and CoronaVirus.

5.2 Extractive Text Summarization Using TF-IDF

Extractive Text Summarization is used to generate a summary using representative tweets for the dataset. This algorithm aims to identify a list of representative tweets for large documents. While the keyword extraction algorithm returns a list of the most important topics of the dataset, summarization would include more

significant phrases and sentences. The input for this algorithm is a set of preprocessed tweets, and the output is a list of representative tweets. This algorithm has five essential phases, which we explain in the following subsections.

5.2.1 Sentence Tokenization

Sentence tokenization can be defined as converting a sentence into a list of words. This is the first step before performing any other functions. We used the Python dictionary to represent the sentence and the word tokens. All the documents were preprocessed before tokenization. During the preprocessing, all the unwanted words, such as "is," "are," and "to," were removed. The sentence tokenization process for the following example would be as follows:

1. {'Monday morning is lazy': [Monday, morning, lazy]}
2. {'Today is Monday': [Today, Monday]}

5.2.2 TDF Score Assignment

We used the same mechanism from Section 5.1.2 to compute the TDF weights. We created a TF matrix and an IDF matrix; then, we multiplied the scores to generate the final weight. For example we mentioned in Section 5.1.1, we presented the sample results below in the same way the algorithm returns.

1. {'Monday morning is lazy': {{'Monday': 0}, {'morning': 0.03}, {'lazy' : 0.03}}}
2. {'Today is Monday': {{'Today': 0.047}, {'Monday': 0}}}

5.2.3 Sentence Scoring and Ranking

Based on the TDF weights of all of the word vectors, we determined the sentence scores. All of the sentence scores were also represented through Python dictionaries, in which the key is the tweet, and the value is the sentence score. Sentence scoring follows the following mechanism:

1. Count_words_in_a_sentence : It counts the number of words in each document.

2. Total_score_per_sentence : this would denote the sum of all scores.

$$Sentence\_score : \frac{total\_score\_per\_sentence}{count\_words\_in\_a\_sentence} \tag{5.5}$$

We then find the average sentence_score which would serve as a threshold for summary generation.

5.2.4 Hashtag Normalization

Hashtag normalization is a feature that may or may not be implemented along with the algorithm, depending on the datasets. If the datasets include hashtags, this function can be included, and if the dataset does not have hashtags, we can exclude this function. From the Twitter API, we already had received a list of hashtags that are present in the dataset. Using the Hashtag Frequency algorithm mentioned in Section 5.1.3, we received a list of top hashtags that were very relevant in the dataset. These top hashtags will be referred to as keywords.

We looked for sentences that contained these keywords and increased the score of the sentences by n. The goal of this modification was to give special importance to the sentences that contained the keywords, as we assumed that keywords were the most important topics within the dataset.

5.2.5 Generating Summary

The arguments passed through this function are: tweets, sentence_value, and threshold. The initial threshold is the average of the sentence_scores we calculated. Based on the threshold, phrases from all the tweets that have a score over the threshold value would be put in as the summary. Depending on factors such as randomness and summary size, we changed the threshold value.

1. **Randomness**: When the data does not have a common base or if the tweets are of wide variety, the randomness of the data increases. Random data would result in very large summaries. In such scenarios, we increased the threshold.

2. **Summary Size**: If the user would like to have 3-4 lines of summary, the threshold can be increased accordingly.

5.3 Document Similarity

Document similarity identifies similar documents, and it can be achieved by creating distance functions (like Euclidean distance). We proposed two distance

functions that can be used to identify plagiarism and mirrored websites. In the following subsections, we discuss cosine and Jaccard similarity, two widespread measurements of computing text similarity.

5.3.1 Cosine Similarity

Cosine similarity measures how similar two vectors of an inner product space are [31]. It is measured by the cosine angle between two vectors, and it determines whether two vectors are pointing in the same direction. Cosine similarity is mostly used in text analytics for information retrieval and analyzing document similarity. Cosine similarity is a good measure to identify plagiarism and is given by the following formula:

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{5.6}$$

where

te = dot product of the two vectors

Example:

We will show the implementation of cosine similarity for the following four documents.

The similarity algorithm is implemented as follows:

1. Preprocess the text

Table 5.4: Example Tweets for Similarity index algorithm

Example Texts
Computer Science is the study of computers and computational systems
Artificial Intelligence is the branch of computer sciences that emphasizes the development of intelligence machines, thinking and working like humans
Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data
While artificial intelligence (AI) is the broad science of mimicking human abilities, machine learning is a specific subset of AI that trains a machine how to learn.

2. Compute the TDF-weights for each word in the documents

3. Compute a sparse matrix

4. Compute the cosine/jaccard similarity matrix from the sparse matrix

5. Use the similarity matrix to retrieve the similarity indices

For the examples mentioned in Table 5.4, the sparse matrix is shown in Figure 5.1 and for document 3 in Table 5.4, ”Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data”, the similarity indices are given in Table 5.5.

```
4 x 4 sparse Matrix of class "dsCMatrix"
          3         4         1         2
3 1.0000000 0.5458690 0.3000000 0.4341216
4 0.5458690 1.0000000 0.3119251 0.3868955
1 0.3000000 0.3119251 1.0000000 0.4341216
2 0.4341216 0.3868955 0.4341216 1.0000000
>
```

Figure 5.1: Sparse matrix for Cosine similarity

Table 5.5: Example Cosine similarity indices

Document	Similarity Index
"While artificial intelligence (AI) is the broad science of mimicking human abilities, machine learning is a specific subset of AI that trains a machine how to learn"	0.5458
"Computer Science is the study of computers and computational systems"	0.4312

5.3.2 Jaccard Similarity

Unlike in cosine similarity, where the total number of common attributes is divided by the total number of attributes possible, in Jaccard similarity, the number of specific attributes is divided by the number of attributes existing in at least one of the two objects. The cosine index can be used to detect plagiarism, but it is not an efficient index to detect internet mirror sites. Although the Jaccard index is a proper index for finding mirror sites, it is not as useful for capturing copy-paste plagiarism (within a larger document). Based on the individual problem statement, one of the two algorithms can be chosen for each project. The Jaccard

Similarity Index for vectors A and B is calculated using the following formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.7}$$

For the same example as in Table 5.4, the sparse matrix for Jaccard similarity is shown in Figure 5.2 and for document 3, ”Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data”, the similarity indices are presented in Table 5.6.

```
4 x 4 sparse Matrix of class "dgCMatrix"
          3         4         1         2
3 1.0000000 0.2702703 0.1290323 0.2000000
4 0.2702703 1.0000000 0.1428571 0.1818182
1 0.1290323 0.1428571 1.0000000 0.2272727
2 0.2000000 0.1818182 0.2272727 1.0000000
>
```

Figure 5.2: Sparse matrix for Jaccard similarity

Table 5.6: Example Jaccard similarity indices

Document	Similarity Index
"While artificial intelligence (AI) is the broad science of mimicking human abilities, machine learning is a specific subset of AI that trains a machine how to learn"	0.2702
"Computer Science is the study of computers and computational systems"	0.1290

Chapter 6

Experimental Results

This chapter is divided into two sections; in the first section, we discuss the results of sentiment analysis tools, and in the second section, we demonstrate the Tweet summarization techniques. We present the implementation and the results of our experiments with three types of summarization techniques in this chapter.

6.1 Comparison of Sentiment Analysis Tools

We compared and evaluated five tools against the manually annotated benchmark datasets. The scores assigned by all of the tools were within the range of [-1,1], but for assigning the labels, we normalized the scores into the seven categories of the 7-score method. The benchmark served as ground truth to assess the accuracy of the five tools, and we calculated metrics that assess prediction error and ranking error. The evaluation metrics are introduced and explained in the following subsections. We discuss the classification error and its metrics in the first subsection,

and in the second subsection, we discuss the ranking error and its metrics.

6.1.1 Classification Error

We computed the Standard Accuracy and Mean Absolute Error (MAE) for all of the tools against the benchmark, respectively. As discussed in Chapter 4, the scoring mechanism for the benchmark categorizes emotion scores into seven classes, and the scores for the five tools have also been normalized into these seven classes. For MAE, we converted the ordinal classes into numbers: one through seven.

6.1.1.1 Classification Accuracy

The accuracy of a measurement system is the degree to which a quantity's measurements is close to the quantity's true value The formula for accuracy is given as follows:

$$Accuracy(\%) = \frac{TrueValues}{TotalValues} \tag{6.1}$$

Where True values = Correctly predicted scores and Total Values = Total number of scores

6.1.1.1.1 Similarity Matrix Accuracy was computed for all of the tools against each other and put together in a similarity matrix. The scores in red indicate the evaluation against the benchmark in percentages. The next three figures show the accuracy measures for the three datasets that have been created for Sentiment Analysis Evaluation:

1. The Common Dataset consists of 400 tweets, and all of the tweets were extracted randomly over two months. Figure 6.1 shows the results, in which Python NLTK performed the best for the given set of tweets.

lexicon_score	tb_score	manual_score	nltk_score	sa_score	score_sentiR
[[0.	45.11278195	**32.33082707**	40.10025063	54.13533835	63.15789474]
[45.11278195	0.	**42.35588972**	45.61403509	56.14035088	56.14035088]
[32.33082707	**42.35588972**	**0.**	**56.89223058**	**24.56140351**	**35.58897243]**
[40.10025063	45.61403509	**56.89223058**	0.	36.09022556	44.86215539]
[54.13533835	56.14035088	**24.56140351**	36.09022556	0.	65.6641604]
[63.15789474	56.14035088	**35.58897243**	44.86215539	65.6641604	0.]]

Figure 6.1: Similarity matrix for Common Dataset

2. The Positive Dataset consists of 75 tweets, most of which are positive. It is a very challenging dataset, as there are tweets that contribute majorly towards one sentiment, and they are all overtly high-polarity tweets. Based on the results shown in Figure 6.2, the lexicon-based model performed best for this set.

3. The negative dataset consists of 120 tweets, and most of them were scored negatively by the tools. However, during the manual assessment, many of them were scored positive. This dataset covers the aspects of sarcasm and irony, and it explores the extremely challenging aspects of Sentiment Analysis. Figure 6.3 shows the results for the negative dataset, and it is safe to assume that the tools did not perform effectively for this particular dataset.

lexicon_score	tb_score	manual_score	nltk_score	sa_score	score_sentiR
[[0.	32.89473684	43.42105263	11.84210526	9.21052632	39.47368421]
[32.89473684	0.	34.21052632	35.52631579	0.	26.31578947]
[43.42105263	34.21052632	0.	22.36842105	10.52631579	38.15789474]
[11.84210526	35.52631579	22.36842105	0.	0.	7.894736848]
[9.21052632	0.	10.52631579	0.	0.	31.57894737]
[39.47368421	26.31578947	38.15789474	7.89473684	31.57894737	0.]]

Figure 6.2: Similarity matrix for Positive Dataset

lexicon_score	tb_score	manual_score	nltk_score	sa_score	score_sentiR
[[0.	50.83333333	19.16666667	23.33333333	50.83333333	60.]
[50.83333333	0.	15.83333333	25.	60.	49.16666667]
[19.16666667	15.83333333	0.	25.	10.83333333	20.83333333]
[23.33333333	25.	25.	0.	17.5	29.16666667]
[50.83333333	60.	10.83333333	17.5	0.	55.83333333]
[60.	49.16666667	20.83333333	29.16666667	55.83333333	0.]]

Figure 6.3: Similarity matrix for Negative Dataset

6.1.1.1.2 Mean Absolute Error In statistics, mean absolute error (MAE) is a measure of difference between two continuous variables [32]. As an example, assume two continuous variables X and Y that represent the same phenomenon, with one being the actual truth (X) and the other being predicted value with alternative techniques (Y). If X and Y need to be compared, the MAE would compute the average vertical or the horizontal distance between each point and the identical line [32]. The identity line is the line that is often used as a comparison measure between two sets of data that are expected to be identical in a 2-dimensional

scatter-plot [33]. The mean absolute error is given by

$$mae = \frac{\sum_{i=0}^{n} abs(y_i - \lambda(x_i))}{n} \tag{6.2}$$

where y_i is the predicted value and x_i is the actual value.

As discussed in previous chapters, we followed a 7-class and a 3-class evaluation method. MAE was computed to evaluate the class identification error. The tables below show the mean absolute error of each tool in comparison to the benchmark datasets. As seen from the results in Table 6.1, NLTK and TextBlob performed the best, with error values of 0.63, 0.93 for 7-class evaluation on the common dataset. However, on the positive and negative datasets, shown in Table 6.2, and Table 6.3, we concluded that TextBlob maintained consistency on challenging datasets.

Table 6.1: MAE for 7-class evaluation for Common dataset

Tool	**MAE against Benchmark**
NLTK	0.6397
TextBlob	0.9323
Sentiment Analysis	1.3959
SentiR	1.0401
Lexicon Based Model	1.0350

Table 6.2: MAE for 7-class evaluation for Positive dataset

Tool	MAE against Benchmark
NLTK	1.1711
TextBlob	0.8684
Sentiment Analysis	1.5789
SentiR	0.93421
Lexicon Based Model	0.7763

Table 6.3: MAE for 7-class evaluation for Negative dataset

Tool	MAE against Benchmark
NLTK	1.3631
TextBlob	1.283
Sentiment Analysis	1.3756
SentiR	1.1916
Lexicon Based Model	1.2

6.1.2 Rank Error

In this section, we discuss how correlated the scores are to the benchmark. We used the Spearman correlation test and Kendall's rank correlation coefficient, because Spearman correlation works best with ordinal data, and Kendall's coefficient gives a degree of agreement among the tools. Both measures can deal with ties and further description, and examples are provided in the following sections.

6.1.2.1 Spearman Correlation Test

The Spearman Correlation Test is similar to the Pearson Correlation Coefficient, but instead is applied to the rank orders [34]. Rank order involves classifying items according to preference or in a specific order. In this project, the scores were ranked under the assumption that the most positive score (+1) has the highest rank and the most negative score (-1) has the lowest rank. The Spearman Correlation Test is one of the most reached out tests, as it can be used to describe the direction of the relationship (positive or negative) between two variables. The Spearman correlation coefficient is calculated using the following formulae:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{6.3}$$

$$\rho = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2 \sum_i (y_i - \overline{y})^2}} \tag{6.4}$$

where

Equation 6.3 is used for normal correlation test and equation 6.4 for ties.

d_i = difference in paired ranks, n = number of cases, and i = paired score.

The basic interpretation of the Spearman Correlation coefficient is that the closer the coefficient is to +1/-1, the stronger the relationship is between the two variables in the positive or negative direction, respectively. If the Spearman coefficient is 0 or close to 0, then there is likely no significant relationship between the two

variables. However, the fact that two variables correlate cannot prove any conclusions, and only further analysis and research can reveal more intricate details. To compute the Spearman Correlation coefficient, we used the built-in Spearman test within R that can handle ties.

6.1.2.1.1 Example Correlation Test Let's see the following 2 values as examples and their Spearman test results.

Actual = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Test_1 = [1, 2, 2, 3, 3, 3, 4, 5, 6, 6]

Test_2 = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

We computed the correlation using R, and the input could be raw values, as mentioned above, or we could calculate the ranks and use them as input. Even if the input was raw values, they were automatically converted into ranks by the correlation test function. As shown in Table 6.4, Test 1 and Actual have relatively closer rankings (with ties) when compared to Test 2, which had completely opposite rankings.

Table 6.4: Example Spearman Correlation Results

Test	**Rho Value**
Test_1	0.981
Test_2	-1

The Spearman Correlation Coefficient was computed both for normalized scores (using the 7-score method) and raw scores. The class labels were converted into ranks and used as the input for the test. The ranking mechanism was also explained in Chapter 5, and below are the rho values for all of the tools. Table 6.5, Table 6.6, and Table 6.7 show the Spearman Correlation results for the Common dataset, Positive dataset, and Negative dataset respectively. As shown in the tables below, TextBlob performed more consistently with a coefficient value closer to +1.

Table 6.5: Spearman Correlation Test Results for Common dataset

Tool	**Rho Value (Normalized)**	**Rho Value (Un-normalized)**
NLTK	0.8120	0.7194
TextBlob	0.6818	0.6652
Sentiment Analysis	0.4276	0.6213
SentiR	0.6728	0.6396
Lexicon Based Model	0.6536	0.6679

Table 6.6: Spearman Correlation Test Results for Positive dataset

Tool	**Rho Value (Normalized)**	**Rho Value (Un-normalized)**
NLTK	0.0201	0.0527
TextBlob	0.0450	0.0802
Sentiment Analysis	0.0115	0.0032
SentiR	0.0712	0.0712
Lexicon Based Model	0.0518	0.0518

Table 6.7: Spearman Correlation Test Results for Negative dataset

Tool	Rho Value (Normalized)	Rho Value (Un-normalized)
NLTK	0.3052	0.3085
TextBlob	0.1502	0.3570
Sentiment Analysis	0.2842	0.0762
SentiR	0.3469	0.3216
Lexicon Based Model	0.3246	0.3380

6.1.2.2 Kendall's Rank Coefficient

In statistics, Kendall's Rank Coefficient is used to measure the ordinal association between two measured quantities [35]. A τ test is a non-parametric hypothesis test for statistical dependence based on the τ coefficient [35]. We computed Kendall's Tau-B correlation coefficient (T_b) for the analysis because it is known to being able to handle ties. T_b is considered the non-parametric alternative to the Spearman Correlation Test when there are numerous tied ranks. The Kendall's Rank Coefficient is defined as:

$$\tau_b = \frac{n_c - n_d}{\sqrt{(n_0 - n_1)(n_0 - n_2)}} \tag{6.5}$$

where

n_0 = n(n-1)/2

$n_1 = \sum_i t_i(t_i - 1)/2$

$n_2 = \sum_j u_j(u_j - 1)/2$

n_c = Number of concordant pairs

n_d = Number of dis-concordant pairs

t_i = Number of tied values in the i^{th} group of ties for the first quantity

u_j = Number of tied values in the j^{th} group of ties for the second quantity

Kendall's coefficient always falls within the range of [-1,1] where -1 shows the negative correlation and +1 shows the perfect positive correlation. While tests using the standard Pearson correlation coefficient naturally assume a normal distribution, Kendall's Tau-b makes no assumptions about what kind of distribution the data might have, and it can handle any number of distinct outcomes.

6.1.2.2.1 Example Kendall's Coefficient Test For the same example as 6.1.2.1.1, the results are as follows.

Table 6.8: Example Kendall's Tau Results

Test	**Rho Value**
Test_1	0.9428
Test_2	-1

Similar to the Spearman Correlation Test, Kendall's Rank Coefficient is also performed in R, and the values are automatically converted to ranks. In our experiments, Kendall's Tau-b Coefficient was computed both for normalized values and raw values. The class labels and the raw scores were used as the input for the

test. Table 6.9, Table 6.10, and Table 6.11 show the Kendall's Tau results for the Common dataset, Positive dataset, and Negative dataset respectively. As shown in the tables below, NLTK and TextBlob performed best, with a coefficient value closer to +1.

Table 6.9: Kendall's Tau Results for Common dataset

Tool	**Tau Value (Normalized)**	**Tau Value (Un-normalized)**
NLTK	0.7185	0.5824
TextBlob	0.5899	0.5046
Sentiment Analysis	0.3763	0.4764
SentiR	0.5827	0.4927
Lexicon Based Model	0.5605	0.5533

Table 6.10: Kendall's Tau Results for Positive dataset

Tool	**Tau Value (Normalized)**	**Tau Value (Un-normalized)**
NLTK	0.0181	0.0378
TextBlob	0.0418	0.0661
Sentiment Analysis	0.0106	0.0081
SentiR	0.0678	0.0673
Lexicon Based Model	0.0387	0.0446

Table 6.11: Kendall's Tau Results for Negative dataset

Tool	Tau Value (Normalized)	Tau Value (Un-normalized)
NLTK	0.2481	0.2215
TextBlob	0.1257	0.0230
Sentiment Analysis	0.2533	0.0544
SentiR	0.2912	0.2371
Lexicon Based Model	0.2707	0.2773

6.2 Demonstration of Tweet Summarization Techniques

We worked on three Tweet Summarization techniques: Keyword Extraction, Extractive Summary Generation, and Document Similarity. Section 6.2.1 presents the results of the keyword extraction algorithm for all three datasets, Section 6.2.2 shows the results for summary generation, and Section 6.2.3 displays the document similarity results.

6.2.1 Keyword Extraction Algorithm

Keyword Extraction was performed using three methodologies: Term Frequency (TF), Term Frequency Inverse Document Frequency (TF-IDF), and Hashtag Approach (HF). In this section, we present the results of each of these methodologies for all of the datasets.

6.2.1.1 Common Dataset Keywords

We selected the top 10 keywords for this dataset using the three keyword extraction methodologies. We also extracted the keywords manually and present the results in Table 6.12. In Table 6.13, we show the similarity matrix for the three techniques in comparison to the manual keywords and each other. Based on the similarity matrix in Table 6.13, TF and HF performed best, with an accuracy of 70%.

Table 6.12: Keyword Extraction Algorithm Results on Common dataset

TF keywords	**TF-IDF Keywords**	**HF words**	**Manually Extracted Keywords**
sad	terrible	sad	sad
angry	angry	angry	angry
amazing	talented	amazing	amazing
crazy	dominos	crazy	crazy
happy	happy	happy	happy
kobe	kobebryant	kobebryant	kobebryant
love	love	love	love
monday	mondaymotivation	mondaymotivation	monday
rip	rip	rip	died
life	politics	life	politics

Table 6.13: Similarity Matrix for the Keyword Extraction Algorithms on Common dataset

TF keywords	TF-IDF Keywords	HF words	Manually Extracted Keywords
0	0.4	0.8	0.7
0.4	0	0.6	0.5
0.8	0.6	0	0.7
0.7	0.5	0.7	0

6.2.1.2 Randomized Data Keywords

The randomized dataset contained nearly 2200 tweets collected over six months. This dataset was created to understand how the results vary concerning randomness. Because the data was collected over a long time period, it explored various topics. Therefore, there may have been multiple significant topics or none at all. We did not perform manual analysis on this dataset due to its size. The results are presented in Table 6.14, and Table 6.15 shows the similarity matrix for the three keyword extraction techniques. We concluded that, within this dataset, there was very little agreement among the techniques.

6.2.1.3 Topic Specific Dataset

For the topic-specific dataset, we extracted 250 tweets with the tag "impeach." This dataset contained one significant topic, and all the keywords ideally were relevant to the topic, "impeach." We also manually extracted the keywords for this dataset. The results for the topic-specific dataset are displayed in Table 6.16.

Table 6.14: Keyword Extraction Algorithm Results on Randomized dataset

TF keywords	**TF-IDF Keywords**	**HF words**
maga	maga	maga
trump	trump	trump
itjobs	uber	itjobs
hiphop	inktober	hiphop
tbt	thursday	tbt
good	good	nationalboyfriendday
great	polling	retail
october	earrings	nyc
people	nct	shopmycloset
thanks	atlanta	atlanta

Table 6.15: Similarity Matrix for the Keyword Extraction Algorithms on Randomized dataset

TF keywords	**TF-IDF Keywords**	**HF words**
0	0.3	0.5
0.3	0	0.3
0.5	0.3	0

Table 6.17 shows that the HF identifies keywords with the closest accuracy to the manual keywords, followed by TF. We concluded that, based on the similarity indices of the algorithms, all of them have relatively similar keywords for this dataset.

Table 6.16: Keyword Extraction Algorithm Results on Topic Specific dataset

TF keywords	**TF-IDF Keywords**	**HF words**	**Manually Extracted Keywords**
realdonaldtrump	realdonaldtrump	realdonaldtrump	realdonaldtrump
impeachmenttrial	hypocrisy	impeachmenttrial	impeachmenttrial
impeachment	ken	impeachment	impeachment
trump	trump	trump	trump
defense	defense	impeachtrump	defense
ukraine	ukraine	ukraine	ukraine
potus	potus	potus	potus
maga	vote	vote	vote
people	indictment	politics	politics
senate	senate	senate	senate

Table 6.17: Similarity Matrix for the Keyword Extraction Algorithms on Randomized dataset

TF keywords	TF-IDF Keywords	HF words	Manually Extracted Keywords
0	0.6	0.7	0.8
0.6	0	0.6	0.7
0.7	0.6	0	0.9
0.8	0.7	0.9	0

6.2.2 Extractive Summary Generation

The extractive summary generation algorithm retrieves the key phrases from the entire corpus, which was the set of tweets, in our case. The extractive summary generation technique is especially useful to summarize extremely large datasets. As explained in Chapter 5, the input for this algorithm was a set of tweets, and the output was to be a list of key phrases. We applied this algorithm on the three datasets, and we present the results in the following subsections. For each dataset, we show the summary generated by the algorithm.

6.2.2.1 Extractive Summary for the Common Dataset

The results presented in Figure 6.4 are for 400 tweets and 1.5*threshold value. The increase in threshold value returns a smaller summary

6.2.2.2 Extractive Summary for Randomized Data

The results presented in Figure 6.4 are for 2200 tweets and 4*threshold value. We had to dramatically increase the threshold value because the data did not contain

Figure 6.4: Extractive Summary for Common Dataset and Randomized Dataset

```
consistently amaze radio podcast
simply amaze
gopherbeaver lakeshowyo wow incredi
thesandisquirts look amaze
ohjnuelz amen
wear tv
haha nigga
dokinana excellent always
prlncesscherry nah poor
flynfriday seeeecret source powah
check cartoon via theweek
hollysmorgan
pm monday january useastern
inuwije thank
mkbhd foldable oled display
rorondgo amen
johnpavlovitz truth
write want livejournal
voteforgmo thoughts well
acasablanca aw thank youu
bfresh boy dont git
ashlieweeks change channel
thomvsu beetlecream darn dang flabb
far cm
akiyusav neet lol
chris august hes still
eliehonig thats get
want lately chicken
ibkjohnson kpmgus dear sisthank
gravy beverage food angry gravy
davidcorndc mr mulvaney liar
pittsburgh winter sad
fuck crazy
hour flight
sincerelydon fuck people
ken starralready lose interest
hate fall people
```

```
autumnleaves vermont burlington vermont
view rowoakridge
important burisma
happy meangirlsday
happy nationalboyfriendday best boyfriend ever love endlessly drm
dansen loveeee
gentleman poet
maraj thats mmvwear babyyyyyyy
remember throwbackthursday
mog mog
happy nationalboyfriendday look back iconic heartthrobs
hes allll easyforhim
thread impeachtrump
god colonizer god colonize blackexcellence blackskinhead
fanfact accord rethinkresearch
hire attorney
bell nationalpoetryday
cnco duoogrupofavorito latinamas
happy meangirlsday congregation
happy nationalboyfriendday love u sm idiot
baseball
happy nationalboyfriendday
coreylajoie thats fakenews
youre ugly believe
hate whatreallypissesmeoff
nationalbullyingpreventionmonth
bowl bowl type bowl librarybowls
time tmi
happy meangirlsday
remind thatswheniblewthewhistle
gorgeous burcuzberk afiliak
jesus versefortoday
happy nationalboyfriendday bestfriend alelx love beech lt
class lhsspirit
ever nfhopeconcert
meantake advice putinmypants
```

multiple tweets that fell under the same topic. Hence, each topic was considered significant.

6.2.2.3 Extractive Summary for Topic-Specific Data

The results presented in Figure 6.4 are for 250 Topic-Specific tweets. All the tweets belong to the topic, "#impeach."

```
wheresthewitnesses impeachmenttrials impeachment realdonaldtrump senatecoverup hide
nadler lose lot weight point impeachment
whitehouse ivankatrump realdonaldtrump indictment please gift impeachment
resistbot make easy part royblunt hawleymo duty impeachment witnessesanddocuments
hypocrisy see ken starr impeachment
break news bolton testify impeachment
prayer time like normal trial impeachment
whens draft gon na start coronavirus impeachment wwiil netanyahu
senators talk reporters capitol right impeachment
impeachment democrat media hatecrime
back call overturn election impeachment
give senalexander courage tennesseans impeachment
impeachmenttrial impeachment
let see witness trumpsremovaltrial impeachmenthearings impeachment
whitehouse presssec impeachment word lindseygrahamsc senatemajldr moscowmitch circa presssec grisham circa til today
mittromney kinda rebel impeachment impeachmenttrials
bar president drop far lower bar impeachment repandybiggsaz
intellectual dishonesty impeachment trial perfect example fragile democracy become impeachment
well fast impeachment
weak presentation defense impeachment
republicans impeachment
impeachment aint work
presidents men come home roost karma baby guilty impeachment
political cartoon significance moment read understand know cartoon tomtolestoons impeachment
vote impeachment party line vote america vote impeachment vote america democratstheenemywithin democratshateamerica democratsarecorrupt
attempt bribery bad bribery impeachment make america respectable
go freedom wire impeachment trump kag schiff
instead argue low bar impeachment argue low bar presidency trump impeachment senatetrial
defend unconstitutional without witnessesanddocuments danger danger danger tuesdaythoughts impeachment boltonmusttestify
bolton must testify impeachment trialeditorial
john bolton crash impeachmenttrial
impeachment impeachmenttrial hey man fuck framers
impeachment bolton factor like
npr unsourced mean regard boltons book impeachment
crime mitt romney senate floor chocolate milk impeachment romney bust contraband chocolate milk bottle senate floor
alandersh nonsense sheer drivel please stop impeachment harvard
lose space allusion impeachment bingo card im impress
```

Figure 6.5: Extractive Summary for Topic Specific Dataset

6.2.3 Document Similarity

In this project, we created distance functions to gather all of the tweets that belonged to one topic. These algorithms take the index of any tweet as an input and return all tweets that are similar to that particular tweet. Using the keywords that were produced in Section 6.2.1, we selected our key tweet. In the following subsections, we produce the results for all three datasets using Cosine similarity and Jaccard Similarity.

6.2.3.1 Similar Documents for Common Dataset

Based on the list of keywords in Table 6.12, we chose the keyword "sad" for the demonstration. We then chose a key tweet at random in which the keyword "sad" appears, and we extracted all of the tweets that were similar to that key tweet.

Key Tweet 1:

"Pilot: "Maintain special VFR at or below 2,500"
Pilot: "request flight following," (controllers in regular contract with aircraft/pilot")
Controller: "2 echo X-ray, you're still too low level for flight following at this time."
Dense fog = flying blind.
#sad #RIP9"

Table 6.18 shows the results for the key tweet by Cosine Similarity, and Table 6.19 shows the results for the key tweet by Jaccard Similarity. As shown in Tables 6.18 and 6.19, the Cosine similarity algorithm looks for texts with similar meaning, while the Jaccard similarity algorithm looks for texts with similar wording.

6.2.3.2 Similar Documents for Randomized Dataset

For the dataset with 2200 random tweets, shown in Table 6.14, we chose the keyword, "trump." We then selected a key tweet at random in which "trump" appears. The results of Cosine similarity are presented in Table 6.20, and the results of Jaccard similarity are presented in Table 6.21.

Table 6.18: Similar Tweets for the key tweet 1 by Cosine Similarity

Similar Tweets	Similarity Index
When I saw the news first posted by @TMZ I was better be fake because these reporters rushed to post it so fast! Before local and national news. #tmzbad #KobeBryant #sad	0.598
Kobe, thank you for all those memories. A big loss to the world, my condolence to your families, who are enduring greater pain. To a #legend #Kobe #sad	0.4784
@HammerNation19 Don't forget this was Kobe's first year to be eligible for HOF, which was going to be announced at the All-Star game. #sad	0.3201
My heart goes out to @kobebryant; his daughter. But let's not forget about the other 7 lost their families left behind grieving #PrayersToAll #Sad	0.3189

Table 6.19: Similar Tweets for key tweet 1 by Jaccard Similarity

Similar Tweets	Similarity Index
@espn @wojespn has confirmed the horrible news that @kobebryant has passed from a helicopter crash! #NoWay #Crazy #RIPKobe	0.4181
Live Updates: Helicopter With Kobe Bryant Got Special Approval to Fly - The New York Times https://t.co/IF9dTQ3Vrr	0.3083
@realDonaldTrump The impeachment isn't a hoax. It's a done deal. You are, and forever will be, impeached. #sad	0.1209
@cenkuygur @realDonaldTrump These nuthuggers don't get it until it's them he pisses on. Remember the dossier said he loved golden showers. It just left out the part where he loves giving them. #sad	0.1250

Key Tweet 2:

Trump believes everything he says. He thinks he can do whatever he likes regardless of laws. He is dangerous to the U.S. and citizens! #ImpeachTrumpNow https://t.co/uMIGmYty6a

Table 6.20: Similar Tweets for key tweet 2 by Cosine Similarity

Similar Tweets	Similarity Index
Sexual Assault At College Campus * A 19-year-old has been charged in connection to a sexual assault that happened at a #HudsonValley college campus. #DailyVoice. https://t.co/ugsnZxhwzM	0.7565
The right to vote in a free and democratic society is a sacred privilege many around the world don't have. To my fellow Canadians, please get out and vote on October 21! It's a beautiful thing! #cdnpoli #tcmv #halalsocks #fromtheOR https://t.co/yCstiuNQfQ	0.7565
Trump doubles down on his crime by publicly inviting China to look up dirt on Biden. He normalizes his actions by repeating them in broad daylight. Republicans will say requesting info on an opponent is not against the Constitution in 3, 2, 1. . . #Trumpcrimespree	0.3209
"Wisconsin out here badgering the opposition Every @BadgerMHockey game all season long live and uninterrupted on the home of College Hockey. #FloHockey"	0.1150

Table 6.21: Similar Tweets for key tweet 2 by Jaccard Similarity

Similar Tweets	Similarity Index
Sexual Assault At College Campus * A 19-year-old has been charged in connection to a sexual assault that happened at a #HudsonValley college campus. #DailyVoice. https://t.co/ugsnZxhwzM	0.8333
The right to vote in a free and democratic society is a sacred privilege many around the world don't have. To my fellow Canadians, please get out and vote on October 21! It's a beautiful thing! #cdnpoli #tcmv #halalsocks #fromtheOR https://t.co/yCstiuNQfQ	0.8333
It's just as bad as what happened in Ukraine? @SpeakerPelosi get this clown @realDonaldTrump out of #OURWHITEHOUSE	0.5320
@realdonaldTrump should be worried about what latest #economic news means for #election2020 #economy #recession #tradewars #tariffs @FinancialTimes @MarketWatch @WSJPolitics @ReutersBiz @Reuters @AP @BBC @washingtonpost @nytimes @CNBC @MSNBC @business https://t.co/OhcYAvW29q	0.2850

6.2.3.3 Similar Documents as Topic-Specific Dataset

For the Topic-specific dataset, shown in Table 6.16, we chose the keyword, "impeachment." We then selected a key tweet at random in which "impeachment" appears. The results of Cosine similarity are presented in Table 6.22, and the results of Jaccard similarity are presented in Table 6.23.

Key Tweet 3:

Prayer time. Just like a normal trial! #impeachment

The results depend on how the dataset is curated. If a dataset contains similar

Table 6.22: Similar Tweets for key tweet 3 by Cosine Similarity

Similar Tweets	Similarity Index
"Pledge time! Just like a normal trial! #ImpeachmentTrial #impeachment"	0.82
"#Impeachment #Bolton A Factor, Like it or Not"	0.37
"@SenSherrodBrown Just a heads up, the concept of a "Fair Trial" is in place to protect the DEFENDANT, not the prosecution. #impeachment"	0.33

Table 6.23: Similar Tweets for key tweet 3 by Jaccard Similarity

Similar Tweets	Similarity Index
"Pledge time! Just like a normal trial! #ImpeachmentTrial #impeachment"	0.7
"#Impeachment #Bolton A Factor, Like it or Not"	0.23
"@SenSherrodBrown Just a heads up, the concept of a "Fair Trial" is in place to protect the DEFENDANT, not the prosecution. #impeachment"	0.18

kinds of documents or, in our case, tweets, the results should be reasonably accurate. For example, for the Topic-Specific dataset, shown in Tables 6.21 and 6.22; the results were quite similar to the results for key tweet 3. However, for the Randomized dataset, shown in Tables 6.19 and 6.20, we concluded that the results were not very close to the results for key tweet 2. In this chapter we presented the results for comparison and evaluation of sentiment analysis tools and Tweet summarization techniques. In the next chapter we discuss the results and summarize them.

Chapter 7

Conclusion

This thesis focuses on computational tools for tweet analysis and concentrates on two different research themes. First, popular sentiment analysis tools are used to evaluate tweets. We presented the results of our comparative evaluation of sentiment analysis tools on Twitter data, as well as the results for the three summary techniques we developed. The purpose of this research was to understand how sentiment analysis tools perform on social media data, and we also wanted to study how the tools handle the challenges of sentiment analysis overall. Most of the tools had an average-low performance against the challenging datasets, while two tools, TextBlob and Lexicon-Based Model, performed relatively consistently against all of the datasets' benchmarks. For very straightforward documents, we conclude that it is safe to use any of the tools we evaluated. However, in cases of challenging datasets, we believe it would be ideal to choose TextBlob.

The motivation behind working on summarization techniques was to devise an

efficient way to analyze large documents, like Hurricane Harvey data, for example. Several factors need to be considered before building summarization models, such as data source, content, data size, reliability, and others. Our data source was Twitter, and it had presented a fair share of challenges, as discussed in the next section. Based on the results from the summarization models, we believe it is safe to conclude that it is easier to summarize documents that are topic-specific.

7.1 Challenges

We obtained all of our datasets from Twitter, and one of the notable disadvantages was the social media language. Social media language introduces vocabulary that is not traditional English. So, overcoming that challenge for sentiment analysis was a hassle. However, for the lexicon-based model, we were able to include those unique social media vocabularies in the model's dictionary. Another problem with the Twitter API is the 18,000-tweet-per-day limit on data collection; therefore, we had to perform multiple runs of the data collection.

Getting the data set up correctly for all of the tools was almost impossible. Since we were working with two different tasks, we had to curate five datasets that were specifically designed for the tools. We manually annotated three of the five benchmark datasets. Another challenge we faced with the sentiment analysis tools was scoring. Even though the scores were within the range of [-1,1], each tool used a different mechanism for scoring. Therefore, we devised a normalization technique that followed a 7-score method so that all the scores would be on the same

scale.

Our data was very limited for building the summarization models. We developed a Twitter-specific model, the Hashtag Approach for keyword extraction. Only 8% of tweets posted each day have hashtags, and since all of the datasets we created must have hashtags, we had to collect thousands of raw tweets.

7.2 Future Work

There is a lot of potential to improve the work presented in this thesis. Throughout this project, we explored NLP techniques for sentiment analysis and summarization models, but we only compared and evaluated five tools. We could extend this work to evaluating more tools. We developed tweet summarization methods, however we were not able to evaluate them due to the lack of benchmark datasets that contain the ground truth for the three different summarization tasks. Therefore, creating such a benchmark and to use those then to evaluate the three summarization methods is a promising direction for future work. Lastly, the data can be extended beyond Twitter: we can work to build reusable models for numerous other kinds of text data.

Bibliography

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of Twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, USA, 2011. Association for Computational Linguistics.

[2] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! In *International AAAI Conference on Web and Social Media*, 2011.

[3] J. Clement. Twitter: number of active users 2010-2019. *https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/*, Aug 2019.

[4] Analyze – Twitter developers. *https://developer.Twitter.com/en/use-cases/analyze*, May 2019.

[5] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

[6] E. Cambria, B. Schuller, Y. Xia, and C. Havasi. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, 28(2):15–21, 2013.

[7] Andrew Ortony, Gerald L. Clore, and Allan Collins. *The cognitive structure of emotions*. Cambridge University Press, 1990.

[8] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. Association for Computing Machinery.

[9] Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. Automatic detection of text genre. In *Proceedings of the 35th Annual Meeting of the Association*

for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, ACL '98/EACL '98, pages 32–38, USA, 1997. Association for Computational Linguistics.

[10] M. Kanakaraj and R. M. R. Guddeti. NLP based sentiment analysis on Twitter data using ensemble classifiers. In *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–5, 2015.

[11] Filipe Nunes Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Marcos André Gonçalves, and Fabrício Benevenuto. Sentibench - a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, 5:1–29, 2016.

[12] Ahmed Abbasi, Ammar Hassan, and Milan Dhar. Benchmarking Twitter sentiment analysis tools. In *LREC*, 2014.

[13] Wei Wu, Bin Zhang, and Mari Ostendorf. Automatic generation of personalized annotation tags for Twitter users. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 689–692, USA, 2010. Association for Computational Linguistics.

[14] Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achanauparp, Ee-Peng Lim, and Xiaoming Li. Topical keyphrase extraction from Twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 379–388, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[15] A. Bellaachia and M. Al-Dhelaan. Ne-rank: A novel graph-based keyphrase extraction in Twitter. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 372–379, 2012.

[16] Kazi Saidul Hasan and Vincent Ng. Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 365–373, USA, 2010. Association for Computational Linguistics.

[17] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[18] S. Phuvipadawat and T. Murata. Breaking news detection and tracking in Twitter. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 120–123, 2010.

[19] Luís F. S. Teixeira, José Gabriel Pereira Lopes, and Rita Almeida Ribeiro. Automatic extraction of document topics. In *DoCEIS*, 2011.

[20] Priyanka Tyagi and R. C. Tripathi. A review towards the sentiment analysis techniques for the analysis of Twitter data. In *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) 2019*, 2019.

[21] Ummara Chauhan, Muhammad Afzal, Abdul Shahid, Moloud Abdar, Ehsan Basiri, and Xujuan Zhou. A comprehensive analysis of adverb types for mining user sentiments on Amazon product reviews. *World Wide Web*, Feb 2020.

[22] Vasileios Hatzivassiloglou and Janyce M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *COLING 2000 Volume 1: The 18th International Conference on Computational Linguistics*, 2000.

[23] Natural language toolkit. In *https://www.nltk.org/book/ch00.html*.

[24] Contributing guidelines. In *Contributing guidelines - TextBlob 0.15.2 documentation*.

[25] Nicolas Pröllochs, Stefan Feuerriegel, and Dirk Neumann. Statistical inferences for polarity identification in natural language. *Plos One*, 13(12), 2018.

[26] E. Henry. Are investors influenced by how earnings press releases are written? *Journal of Business Communication*, 45(4):363–407, Jan 2008.

[27] Bharat Patil. Lmmasterdictionary, loughran and mcdonald sentiment master dictionary in edgar, platform for edgar filing management and textual analysis. *https://rdrr.io/cran/edgar/man/LMMasterDictionary.html*, 2019.

[28] Matthew Jockers. Cumulative sentiments. In *http://www.matthewjockers.net/*.

[29] Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1367–1373, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING.

[30] Richard Tong. An operational system for detecting and tracking opinions in on-line discussions. In *Working Notes of the SIGIR Workshop on Operational Text Classification*, pages 1–6, New Orleans, Louisianna, 2001.

[31] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Elsevier/Morgan Kaufmann, 2012.

[32] T. Chai and R. R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3):1247–1250, June 2014.

[33] Michael Friendly and Daniel Denis. The early origins and development of the scatterplot. *Journal of the history of the behavioral sciences*, 41:103–30, Feb 2005.

[34] *Spearman Rank Correlation Coefficient*, pages 502–505. Springer New York, New York, NY, 2008.

[35] *Kendall Rank Correlation Coefficient*, pages 278–281. Springer New York, New York, NY, 2008.