# ABSTRACT

Social media is increasingly used by humans to express their feelings and opinions in the form of short text messages. Detecting sentiments in text has a wide range of applications including identifying anxiety or depression of individuals and measuring well-being or mood of a community. Sentiments can be expressed in many ways that can be seen such as facial expression and gestures, speech and by written text.

Sentiment Analysis in text documents is essentially a content – based classification problem involving concepts from the domains of Natural Language Processing as well as Machine Learning. In this paper, sentiment recognition based on textual data and the techniques used in sentiment analysis are discussed.

Sentiment analysis deals with identifying and classifying opinions or sentiments expressed in source text. Social media is generating a vast amount of sentiment rich data in the form of tweets, status updates, blog posts etc. Sentiment analysis of this user generated data is very useful in knowing the opinion of the crowd. Twitter sentiment analysis is difficult compared to general sentiment analysis due to the presence of slang words and misspellings. The maximum limit of characters that are allowed in Twitter is 140.

Knowledge base approach and Machine learning approach are the two strategies used for analyzing sentiments from the text. In this paper, we try to analyze the twitter posts about electronic products like mobiles, laptops etc using Machine Learning approach. By doing sentiment analysis in a specific domain, it is possible to identify the effect of domain information in sentiment classification. We present a new feature vector for classifying the tweets as positive, negative and extract peoples' opinion about products.

# TABLE OF CONTENT

**CHAPTER NO. TITLE PAGE NO**

**ABSTRACT**

**LIST OF ABBREVATIONS**

## LIST OF FIGURES 1 INTRODUCTION 1

1.1 OVERVIEW 1

1.1.1 Problem statement and applicability 1

1.1.2 Proposed methodology 2

1.2 DEFINITION AND MOTIVATION 2

## 2 LITERATURE REVIEW 5

2.1 SENTIMENT ANALYTICS 5

2.2SENTIMENT ANALYSIS USERS 7

2.2.1 Sentiment analytic uses 8

2.2.2 Understanding sentiment 9

2.3 SENTIMENT ANALYSIS CHALLENGES 10

2.3.1 Subjective and tones 12

2.3.2 Context and polarity 12

2.3.3 Irony and sarcasm 13

2.3.4 Comparison 13

2.3.5 Emojis 14 **3 SYSTEM SPECIFICATION 16**

3.1PROCESSING METHODS **16**

3.2 DATASET **16**

3.3 PRE-PROCESSING  **19**

3.3.1 Basic feature extraction **19**

3.4 ADVANCED TEXT PROCESSING **21**

## 4 DESIGN METHODOLOGY 24

4.1 FEATURE EXTRACTION **24**

4.2 SPLITTING DATASET **27**

## 5 IMPLEMENTATION 30

5.1 CONFUSION MATRIX **30**

5.2 ROC CURVE **30**

5.3 COMPARISON(UNIGRAM ONLY) **31**

5.4 ACCURACY(N-grams) **32**

## 6 CONCLUSION 33

6.1 CONCLUSION **33**

6.2 FUTURE ENHANCEMENT **34**

## 7 REFERENCES 34 APPENDIX 1 – SAMPLE CODE 35 APPENDIX 2 – SCREENSHOTS 45

.

**LIST OF ABBREVIATIONS**

**ACL -** Anterior Cruciate Ligament

**AI -** Artificial Intelligence

**ANN -** Artificial Neural Network

**API -** Application Programming Interface

**CNN -** Convolutional Neural Network

**IDF -** Inverse Document Frequency **NLP -** Natural Language Processing

**NLTK -** Natural Language Toolkit

**URL -** Uniform Resource Locator

**ROC -** Receiver Operating Characteristics

**SVM -** Support Vector Machine

**OPENCV -** Open-Source Computer Vision

**TF -** Term Frequency

## LIST OF FIGURES

**FIGURE NO FIGURE NAME PAGE NO.**

**1.1** Sentiment Analysis Flowchart 13

**2.1** Types of Emotions 15

**3.1** Emoticons 20

**3.2** Most used Words 23

**4.1** Term Frequency Graph25

**4.2** Most Used Words 26

**4.3** Pie Chart 27

**4.4** Naive Bayes Model 29

**5.1** Confusion Matrix 30

**5.2** ROC Graph 31

**A.2.1** Loading Dataset 45

**A.2.2** Extraction Process 45

**A.2.3** Most used Positive words 46

**A.2.4** Confusion matrix 46

**A.2.5** Classification 47

# CHAPTER 1

## INTRODUCTION

**1.1 OVERVIEW**

Sentiment Analysis is a technique widely used in text mining. Twitter Sentiment Analysis, therefore means, using advanced text mining techniques to analyze the sentiment of the text (here, tweet) in the form of positive, negative and neutral. It is also known as Opinion Mining, is primarily for analyzing conversations, opinions, and sharing of views (all in the form of tweets) for deciding business strategy, political analysis, and also for assessing public actions.

Enginuity, Revealed Context, Steamcrab, MeaningCloud, and SocialMention are some of the well-known tools used for the analysis of Twitter sentiment. R and Python are widely used for sentiment analysis dataset twitter. Sentiment Analysis of Twitter data is now much more than a college project or a certification program. A good number of Tutorials related to Twitter sentiment are available for educating students on the Twitter sentiment analysis project report and its usage with R and Python. You may also enroll for a python tutorial for the same program to get a promising career in sentiment analysis dataset twitter.

Our discussion will include, Twitter Sentiment Analysis in R and Python, and also throw light on its techniques and teach you how to generate the Twitter Sentiment Analysis project report, and the advantages of enrolling for its Tutorial.

**1.1.1 PROBLEM STATEMENT & APPLICABILITY**

Since the Advent of the Internet, humans have used it as a communication tool in the form of mostly text messages and nowadays video and audio streams and as we increase our dependence on technology it becomes increasingly important to better gauge human sentiments expressed with the help of technology.

However, in this textual communication data, we lose the access to sentiments or the emotions conveyed behind a sentence, as we often use our hands and facial expressions to express our intent behind the statement.

From this textual data, we can gain insights into the individual. Insights which can be used for multiple different uses such as content recommendation based on current mood, market segmentation analysis and psychological analysis in humans. In this project, we have attempted to classify human sentiment into two categories namely positive and negative.

Which helps us better understand human thinking and gives us an insight which can be used in a variety of ways as stated above.

**1.1.2 PROPOSED METHODOLOGY**

In this paper we classify sentiments with the help of machine learning and natural language processing (NLP) algorithms, we use the datasets from Kaggle which was crawled from the internet and labeled positive/negative.

The data provided comes with emoticons (emoji), usernames and hashtags which are required to be processed (so as to be readable) and converted into a standard form. We also need to extract useful features from the text such unigrams and bigrams which is a form of representation of the “tweet”.

We use various machine learning algorithms based on NLP (Natural Language Processing) to conduct sentiment analysis using the extracted features. Finally, we report our experimental results and findings at the end.

## 1.2DEFINITION AND MOTIVATION

Sentiment analysis is a strategy for checking assessments of people or groups; for example, a portion of a brand’s followers for an individual customer in correspondence with a customer support representative. With regard to a scoring mechanism, (IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 10, No. 2, 2019362 | P a g ewww.ijacsa.thesai.orgsentiment analysis monitors discussions and assesses dialogue and voice affectations to evaluate moods and feelings,especially those associated with a business, product or service,or theme.

Sentiment analysis is a means of assessing written orspoken languages to decide whether articulation is positive,negative or neutral and to what degree.

The current analysis tools in the market are able to deal with tremendous volumes of customer criticism reliably and precisely. In conjunction with contents investigation, sentiment analysis discovers customer’s opinions on various topics, including the purchase of items,provision of services, or presentation of promotions.Immense quantities of client-created web-based social networking communications are being persistently delivered in the forms of surveys, online journals, comments, discourses,pictures, and recordings.

These correspondences offer significant opportunities to obtain and comprehend the point of view of clients on themes such as intrigue and provide data equipped for clarifying and anticipating business and social news, such as product offers, stock returns, and the results of political decisions.

Integral to these examinations is the assessment of the notions communicated between clients in their content interchanges.“Notion examination” is a dynamic area of research designed to enhance computerized understanding of feelings communicated in content, with increases in implementation prompting more powerful utilization of the inferred data.Among the different web-based social networking platforms,Twitter has incited particularly far-reaching client appropriation and rapid development in terms of correspondence volume.

Twitter is a small-scale blogging stage where clients generate 'tweets' that are communicated to their devotees or to another client. At 2016, Twitter has more than 313 milliondynamic clients inside a given month, including 100 millionclients daily. Client origins are widespread, with 77%situated outside of the US, producing more than 500 milliontweets every day.

The Twitter site positioned twelfth universally for activity in 2017 and reacted to more than 15billion API calls every day. Twitter content likewise shows up in more than one million outsider sites. In Accordance with this enormous development, Twitter has oflate been the subject of much scrutiny, as Tweets frequently express client's sentiment on controversial issues.

In the social media context, sentiment analysis and mining opinions are highly challenging tasks, and this is due to the enormous information generated by humans and machines.

The data given from the dataset is in the form of commaseparated values files with

“tweets” and their corresponding sentiments., tweet where the tweet\_id is a unique integer identifying the tweet, sentiment is either 1 (positive) or 0 (negative), and tweet is the tweet enclosed in "". Similarly, the test dataset is a csv file of type tweet\_id, tweet respectively. The dataset is a mixture of words, emoticons, symbols, URLs and references to people as seen usually on twitter. Words and emoticons contribute to predicting the sentiment, but URLs and references to people don’t. incorrect, extra punctuations, and words with many repeated letters.

The “tweets”, therefore, must be preprocessed to standardize the dataset.

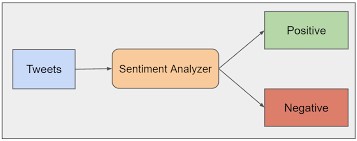


Figure 1.1 Sentiment Analysis Flowchart

# CHAPTER 2

## LITERATURE REVIEW

### 2.1 SENTIMENT ANALYTICS

Natural Language Processing (NLP) deals with actual text element processing. The text element is transformed into machine format by NLP. Artificial Intelligence (AI) uses information provided by the NLP and applies a lot ofmaths to determine whether something is positive or negative. Several methods exist to determine an author’s view on a topic from natural language textual information.

Some form of machine learning approach is employed and which has varying degree of effectiveness. One of the types of natural language processing is opinion mining which deals with tracking the mood of the people regarding a particular product or topic. This software provides automatic extraction of opinions, emotions and sentiments in text and also tracks attitudes and feelings on the web. People express their views by writing blog posts, comments, reviews and tweets about all sorts of different topics. Tracking products and brands and then determining whether they are viewed positively or negatively can be done using web.

The opinion mining has slightly different tasks and many names, e.g. sentiment analysis, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining, etc. Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages.

As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation. Modern NLP algorithms are based on machine learning, especially statistical machine learning. The paradigm of machine learning different from that of most prior attempts at language processing. Prior implementations of language-processing tasks typically involved the direct hand coding of large sets of rules.

The machine-learning paradigm calls instead for using general learning algorithms — often, although not always, grounded in statistical inference — to automatically learn such rules through the analysis of large corpora of typical real-world examples.

A corpus (plural, "corpora") is a set of documents (or sometimes, individual sentences) that have been hand-annotated with the correct values to be learned. Many different classes of machine learning algorithms have been applied to NLP tasks. These algorithms take as input a large set of "features" that are generated from the input data. Some of the earliest-used algorithms, such as decision trees, produced systems of hard if-then rules similar to the systems of hand-written rules that were then common.

Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each input feature. Such models have the advantage that they can express the relative certainty of many different possible answers rather than only one, producing more reliable results when such a model is included as a component of a larger system.

Systems based on machine-learning algorithms have many advantages over handproduced rules: The learning procedures used during machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often not at all obvious where the effort should be directed.

Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input (e.g. containing words or structures that have not been seen before) and to erroneous input (e.g. with misspelled words or words accidentally omitted). Generally, handling such input gracefully with hand-written rules — or more generally, creating systems of hand-written rules that make soft decisions — is extremely difficult, errorprone and time-consuming.

Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, systems based on hand-written rules can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task.

In particular, there is a limit to the complexity of systems based on hand-crafted rules, beyond which the systems become more and more unmanageable. However, creating more data to input to machine-learning systems simply requires a corresponding increase in the number of man-hours worked, generally without significant increases in the complexity of the annotation process.

The subfield of NLP devoted to learning approaches is known as Natural Language Learning (NLL) and its conference CoNLL and peak body SIGNLL are sponsored by ACL, recognizing also their links with Computational Linguistics and Language Acquisition. When the aim of computational language learning research is to understand more about human language acquisition, psycholinguistics, NLL overlaps into the related field of Computational Psycholinguistics.

Opinions are fundamental to every single human action since they are key influencers of our practices. At whatever point we have to settle on a choice, we need to know others thoughts.

In reality, organizations and associations dependably need to discover users’ popular sentiments about their items and services. Clients use different types of online platforms for social engagement including web-based social networking sites; for example, Facebook and Twitter.

Through these web based social networks, buyer engagement happens progressively. This kind of connection offers a remarkable open door for advertising knowledge. Individuals of every nationality, sexual orientation, race and class utilize the web to share encounters and impressions about virtually every feature of their lives. Other than composing messages, blogging or leaving remarks on corporate sites, a great many individuals utilize informal organization destinations to log opinions,express feelings and uncover insights about their everyday lives.

Individuals compose correspondence on nearly anything,including films, brands, or social exercises. These logs circulate throughout online groups and are virtual gatherings where shoppers illuminate and impact others.

### 2.2 SENTIMENT ANALYTICS

Sentiment analysis (a.k.a opinion mining) is the automated process of identifying and extracting the subjective information that underlies a text. This can be either an opinion, a judgment, or a feeling about a particular topic or subject. The most common type of sentiment analysis is called ‘polarity detection’ and consists in classifying a statement as ‘positive’, ‘negative’ or ‘neutral’. For example, let’s take this sentence: “I don’t find the app useful: it’s really slow and constantly crashing”. A sentiment analysis model would automatically tag this as Negative.

A sub-field of Natural Language Processing (NLP), sentiment analysis has been getting a

lot of attention in recent years due to its many exciting applications in a variety of fields, ranging from business to political studies. Thanks to sentiment analysis, companies can understand the reputation of their brand. By analyzing social media posts, product reviews, customer feedback, or NPS responses (among other sources of unstructured business data), they can be aware of how their customers feel about their product.

They can also track specific topics and get relevant insights on how people are talking about those topics. Sentiment analysis is particularly useful for social media monitoring because it goes beyond metrics that focus on the number of likes or retweets, and provides a qualitative point of view.

Let’s say a company has just launched a new product feature and you notice a sharp increase in mentions on Twitter. However, receiving tons of mentions does not necessarily mean a good thing. Are customers tweeting more because they are expressing good things about this new product feature? Or, are customers actually complaining about the feature having lots of bugs? Performing Twitter sentiment analysis can be an excellent way to understand the tone of those mentions and obtain real-time insights on how users are perceiving your new product.

**2.2.1 Sentiment analysis uses:**

Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics. Social media monitoring tools like Brandwatch Analytics make that process quicker and easier than ever before, thanks to real-time monitoring capabilities. The applications of sentiment analysis are broad and powerful.

The ability to extract insights from social data is a practice that is being widely adopted by organisations across the world. Shifts in sentiment on social media have been shown to correlate with shifts in the stock market.

The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election.

Being able to quickly see the sentiment behind everything from forum posts to news articles means being better able to strategise and plan for the future. It can also be an essential part of your market research and customer service approach.

Not only can you see what people think of your own products or services, you can see what they think about your competitors too.

The overall customer experience of your users can be revealed quickly with sentiment analysis, but it can get far more granular too. The ability to quickly understand consumer attitudes and react accordingly is something that Expedia Canada took advantage of when they noticed that there was a steady increase in negative feedback to the music used in one of their television adverts.

**2.2.2 Understanding sentiment:**

Tweet Sentiment analysis conducted by the brand revealed that the music played on the commercial had become incredibly irritating after multiple airings, and consumers were flocking to social media to vent their frustrations.

A couple of weeks after the advert first aired, over half of online conversation about the campaign was negative.

Rather than chalking up the advert as a failure, Expedia was able to address the negative sentiment in a playful and self-knowing way by airing a new version of the advert which featured the offending violin being smashed.

Sentiment analysis is the automated process of analyzing text data and classifying opinions as negative, positive or neutral. Usually, besides identifying the opinion, these systems extract attributes of the expression

e.g.:

* Polarity: if the speaker express a positive or negative opinion,
* Subject: the thing that is being talked about,

**2.3 SENTIMENT ANALYSIS CHALLENGES:**

Natural Language Processing (NLP) deals with actual text element processing. The text element is transformed into machine format by NLP.

Artificial Intelligence (AI) uses information provided by the NLP and applies a lot of maths to determine whether something is positive or negative. Several methods exist to determine an author’s view on a topic from natural language textual information.

Some form of machine learning approach is employed and which has varying degree of effectiveness. One of the types of natural language processing is opinion mining which deals with tracking the mood of the people regarding a particular product or topic. This software provides automatic extraction of opinions, emotions and sentiments in text and also tracks attitudes and feelings on the web.

People express their views by writing blog posts, comments, reviews and tweets about all sorts of different topics. Tracking products and brands and then determining whether they are viewed positively or negatively can be done using web.

The opinion mining has slightly different tasks and many names, e.g. sentiment analysis, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, review mining, etc.

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction.

Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation. Modern NLP algorithms are based on machine learning, especially statistical machine learning.

The paradigm of machine learning different from that of most prior attempts at language processing. Prior implementations of language-processing tasks typically involved the direct hand coding of large sets of rules. The machine-learning paradigm calls instead for using general learning algorithms — often, although not always, grounded in statistical inference — to automatically learn such rules through the analysis of large corpora of typical real-world examples. A corpus (plural, "corpora") is a set of documents (or sometimes, individual sentences) that have been hand-annotated with the correct values to be learned.

Many different classes of machine learning algorithms have been applied to NLP tasks.

These algorithms take as input a large set of "features" that are generated from the input data. Some of the earliest-used algorithms, such as decision trees, produced systems of hard if-then rules similar to the systems of hand-written rules that were then common. Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each input feature.

Such models have the advantage that they can express the relative certainty of many different possible answers rather than only one, producing more reliable results when such a model is included as a component of a larger system. Systems based on machine-learning algorithms have many advantages over hand-produced rules:

The learning procedures used during machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often not at all obvious where the effort should be directed. Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input (e.g. containing words or structures that have not been seen before) and to erroneous input (e.g. with misspelled words or words accidentally omitted). Generally, handling such input gracefully with hand-written rules — or more generally, creating systems of hand-written rules that make soft decisions — is extremely difficult, error-prone and time-consuming.

Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, systems based on hand-written rules can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task. In particular, there is a limit to the complexity of systems based on hand-crafted rules, beyond which the systems become more and more unmanageable. However, creating more data to input to machine-learning systems simply requires a corresponding increase in the number of manhours worked, generally without significant increases in the complexity of the annotation process.

The subfield of NLP devoted to learning approaches is known as Natural Language Learning (NLL) and its conference CoNLL and peak body SIGNLL are sponsored by ACL, recognizing also their links with Computational Linguistics and Language Acquisition.

**2.3.1 Subjectivity and Tone :**

The detection of subjective and objective texts is just as important as analyzing their tone. In fact, so-called objective texts do not contain explicit sentiments. Say, for example, we intend to analyze the sentiment of the following two texts:

* The package is nice.
* The package is red.

Most people would say that sentiment is positive for the first one and neutral for the second one, right? All predicates (adjectives, verbs, and some nouns) should not be treated the same with respect to how they create sentiment. In the examples above, nice is more subjective than red.

**2.3.2 Context and Polarity:**

All utterances are uttered at some point in time, in some place, by and to some people. All utterances are uttered in context. Analyzing sentiment without context gets pretty difficult. However, machines cannot learn about contexts if they are not mentioned explicitly. One of the problems that arise from context is changes in polarity. Look at the following responses to a survey:

* Everything of it.
* Absolutely nothing!

Imagine the responses above come from answers to the question What did you like about the event? The first response would be positive and the second one would be negative, right? Now, imagine the responses come from answers to the question What did you Dislike about the event? The negative in the question will make sentiment analysis change altogether.

A good deal of preprocessing or postprocessing will be needed if we are to take into account at least part of the context in which texts were produced. However, how to preprocess or postprocess data in order to capture the bits of context that will help analyze sentiment is not straightforward.

**2.3.3 Irony and Sarcasm**

Differences between literal and intended meaning (i.e. irony) and the more insulting or ridiculizing version of irony (i.e. sarcasm) usually change positive sentiment into negative whereas negative or neutral sentiment might be changed to positive. However, detecting irony or sarcasm takes a good deal of analysis of the context in which the texts are produced and, therefore, are really difficult to detect automatically.

For example, look at some possible answers to the question Have you had a nice customer experience with us? below.

* Yeah. Sure.
* Not one, but many!

What sentiment would we assign to the responses above? Probably, we have listened to the first response so many times, we would have said negative, right? The problem is there is no textual cue that will make a machine learn that negative sentiment since most often, yeah and sure belong to positive or neutral texts.

How about the second response? In this context, sentiment is positive, but we’re sure we can come up with many different contexts in which the same response can express negative sentiment.

**2.3.4 Comparisons:**

How to treat comparisons in sentiment analysis is another challenge worth tackling. Look at the texts below:

* This product is second to none.
* This is better than the old tools.
* This is better than nothing.

There are some comparisons like the first one above that do not need any contextual clues in order to be classified correctly. The second and third texts are a little more difficult to classify, though. Would we classify them as neutral or positive?

Probably, we are more likely to choose positive for the second one and neutral for the third, right? Once again, context can make a difference.

For example, if the old tools the second text talks about were considered useless in context, then the second text turns out to be pretty similar to the third text. However, if no context is provided, these texts feel different.

**2.3.5 Emojis:**

There are two types of emojis according to Guibon et al.. Western emojis (e.g. :D) are encoded in only one character or in a combination of a couple of them whereas Eastern emojis are a longer combination of characters of a vertical nature. Particularly in tweets, emojis play a role in the sentiment of texts.

Sentiment analysis performed over tweets requires special attention to character-level as well as word-level. However, no matter how much attention we pay to each of them, a lot of preprocessing might be needed.

For example, we might want to preprocess social media content and transform both Western and Eastern emojis into tokens and whitelist them (i.e. always take them as a feature for classification purposes) in order to help improve sentiment analysis performance.

Sentiments can be expressed in many ways that can be seen such as facial expression and gestures, speech and by written text.

Sentiment Analysis in text documents is essentially a content – based classification problem involving concepts from the domains of Natural Language Processing as well as Machine Learning.



Figure 2.1.Types of Emotions

# CHAPTER 3

## SYSTEM SPECIFICATION

**3.1 PROCESSING METHODS:**

**Section A: Preparing The Test Set**

* Step A.1: Getting the authentication credentials
* Step A.2: Authenticating our Python script
* Step A.3: Creating the function to build the Test set

**Section B: Preparing The Training Set**

**Section C: Pre-processing Tweets in The Data Sets**

**Section D: Naive Bayes Classifier**

* Step D.1: Building the vocabulary
* Step D.2: Matching tweets against our vocabulary
* Step D.3: Building our feature vector
* Step D.4: Training the classifier

**Section E: Testing The Model**

### 3.2 DATASET

The data given is in the form of a comma-separated values files with tweets and their corresponding sentiments. The training dataset is a csv file of type tweet\_id, sentiment\_tweet where the tweet\_id is a unique integer identifying the tweet, sentiment is either 1 (positive) or 0 (negative), and tweet is the tweet enclosed in "". We will use this single dataset to cross validate our model.

The dataset is a mixture of words, emoticons, symbols, URLs, hashtags and references to people. Words and emoticons contribute to predicting the sentiment, but URLs and references to people don’t. Therefore, URLs and references can be ignored.

The words are also a mixture of misspelled words, extra punctuation, and words with many repeated letters. The tweets, therefore, have to be preprocessed to standardize the dataset.

Range Index: 99989 entries, 0 to 99988

**Why Twitter Data?**

Twitter is an online micro blogging tool that disseminates more than 400 million messages per day, including vast amounts of information about almost all industries from entertainment to sports, health to business etc. One of the best things about Twitter — indeed, perhaps its greatest appeal — is in its accessibility. It’s easy to use both for sharing information and for collecting it. Twitter provides unprecedented access to our lawmakers and our celebrities, as well as to news as it’s happening. Twitter represents an important data source for the business models of huge companies as well.

All the above characteristics make twitter a best place to collect real time and latest data to analyse and do any sought of research for real life situations.

If you need other datasets, you can download pre-existing datasets of various use cases like cancer detection to Q&A dataset to sports comments to chatbots. Here are various text classification datasets.

Or if you have your own unique use case, you can create your very own dataset for it. You can download images from the web and to make a big dataset in no time, use an annotation tool like Dataturks, where you upload the raw data and tag manually in a ziffy.

**Why is Twitter Sentiment Analysis important?**

Nearly 80% of the world’s digital data is unstructured, and data obtained from social media sources is no exception to that. Since the information is not organized in any predefined way, it’s difficult to sort and analyze. Fortunately, thanks to the developments in Machine Learning and NLP, it is now possible to create models that learn from examples and can be used to process and organize text data.

Twitter sentiment analysis systems allow you to sort large sets of tweets and detect the polarity of each statement automatically.

And the best part, it’s fast and simple, saving teams valuable hours and allowing them to focus on tasks where they can make a bigger impact.

These are some of the main advantages of Twitter sentiment analysis:

* **Scalability:** let’s say you need to analyze hundreds of tweets mentioning a brand. While you could do that manually, it would take hours and hours of manual processing and would end up being inconsistent and impossible to scale. By performing Twitter sentiment analysis you can automate this task and obtain cost-effective results in a very short time.
* **Real-Time Analysis**: Twitter sentiment analysis is critical to notice sudden shifts in customer moods, detect if critics and complaints are increasing and take action before the problem escalates. You can be monitoring your brand in real-time and get valuable insights that allow you to make changes or improvements when needed.
* **Consistent Criteria:** analyzing sentiment in a text is a subjective task. When done manually, the same tweet may be perceived differently by two members of the same team, and the results will probably be biased.

By training a machine learning model to perform sentiment analysis on Twitter, you can set the parameters to analyze all your data and obtain more consistent and accurate results.

How to Use Sentiment Analysis with Twitter Data?

So far we have learned what sentiment analysis is and the benefits it delivers when using it to analyze Twitter data. Now it’s time to learn how to actually do Twitter sentiment analysis.

We can identify four main steps in this process:

* Data gathering
* Data preparing
* The creation of the sentiment analysis model
* Visualization of the results

In this section, we’ll explain each of these stages and provide tools for both coders and noncoders so you can get started with sentiment analysis of tweets right away.

**Data Gathering: How to Get Twitter Data?**

If you want to perform Twitter sentiment analysis, the first step is to gather the data. This is the data that you will use for:

* 1. training the machine learning model, and
  2. running the actual sentiment analysis on Twitter data.

At this point, there’s something important to consider regarding Twitter. There are two main types of tweets you can extract from Twitter:

**Current Tweets:** this is useful to track keywords or hashtags in real-time.

**Historical Tweets:** you can use this to search past tweets during a predefined time frame.

### 3.3 PRE-PROCESSING

**3.3.1 Basic feature extraction**

Pre-processing Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people’s usage of social media. Tweets have certain special characteristics such as retweets, hashtags, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We have applied an extensive number of pre-processing steps to standardize the dataset and reduce its size.

We first do some general pre-processing on tweets which is as follows.

* Convert the tweet to lower case.
* Replace 2 or more dots (.) with space.
* Strip spaces and quotes (" and ’) from the ends of tweet.

We handle special twitter features as follows.

User names and mentions:

Every twitter user has a handle associated with them. Users often mention other users in their tweets by @handle. We delete all user mentions as they are unique and doesn’t contribute anything to sentiment.

**URL:**

Users often share hyperlinks to other webpages in their tweets. Any particular URL is not important for text classification as it would lead to very sparse features. Therefore, we delete all the URLs in tweets. Thus calculation becomes less resource hungry.

**Emoticon:**

Users often use a number of different emoticons in their tweet to convey different emotions. It is impossible to exhaustively match all the different emoticons used on social media as the number is ever increasing.We replace the matched emoticons with either ‘positiveemoji’ or

‘negetiveemoji’ depending on whether it is conveying a positive or a negative emotion. A list of all emoticons matched by our method is given below.

|  |  |  |
| --- | --- | --- |
| **Emoticons** | **Type** | **Replacement** |
| :) , : ) , :-) , (: , ( : , (-: , :’) , :O | Smile | positiveemoji |
| :D , : D , :-D , xD , x-D , XD , X-D | Laugh | positiveemoji |
| ;-) , ;) , ;-D , ;D , (; , (-; , @-) | Wink | positiveemoji |
| <3 , :\* | Love | positiveemoji |
| :-( , : ( , :( , ): , )-: , :-/ , :-| | Sad | negetiveemoji |
| :,( , :’( , :"( | Cry | negetiveemoji |

Figure 3.1 Emoticons

**Hashtag:**

Hashtags are un-spaced phrases prefixed by the hash symbol (#) which is frequently used by users to mention a trending topic on twitter. We replace all the hashtags with the words with the hash symbol. For example, #hello is replaced by hello.

After applying tweet level pre-processing, we processed individual words of tweets as follows.

* Strip any punctuation [’"?!,.():;] from the word.
* Convert 2 or more letter repetitions to 2 letters. Some people send tweets like ‘I am sooooohapppppy’ adding multiple characters to emphasize certain words. This is done to handle such tweets by converting them to ‘I am soo happy’.
* Removes any single characters.
* Remove - and ’. This is done to handle words like t-shirt and their’s by converting them to the more general form tshirt and theirs. Here we use contractions. We have created a separate dictionary file to handle contractions. Like ‘can’t’ will be converted into ‘cannot’.
* Replace 2 or more spaces with a single space.

### 3.4 ADVANCED TEXT PROCESSING

**Normalization:**

Stemming and Lemmatization are Text Normalization (or sometimes called Word

Normalization) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing.

**Tokenization:**

Tokenization refers to dividing the text into a sequence of words or sentences. We could use NLTK library, but we used string.split() which is faster.

**Stemming:**

Stemming refers to the removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach. Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. This indiscriminate cutting can be successful in some occasions, but not always, and that is why we affirm that this approach presents some limitations. For this purpose, we will use PorterStemmer from the NLTK library.

|  |  |  |
| --- | --- | --- |
| **From** | **Suffix** | **Stem** |
| **studies** | **-es** | **studi** |
| **studying** | **-ing** | **study** |

**Lemmatization:**

Lemmatization is a more effective option than stemming because it converts the word into its root word, rather than just stripping the suffices. It makes use of the vocabulary and does a morphological analysis to obtain the root word. Lemmatization takes into consideration the morphological analysis of the words. To do so, it is necessary to have detailed dictionaries which the algorithm can look through to link the form back to its lemma. Therefore, we usually prefer using lemmatization over stemming.

|  |  |  |
| --- | --- | --- |
| **From** | Morphological information | **Lemma** |
| **studies** | Third person, singular number, present tense of verb study | **study** |
| **studying** | Gerund of the verb study | **study** |

**Stop words:**

Not all words determine sentiments. Some of them (mainly prepositions and articles) don’t contribute anything to the sentiment. So we can remove these commonly occurring words from our text data. But NLTK library contains stop words like ‘no’, ‘not’, ‘nor’ etc., which determines positive or negative sentiments. So we built custom set of stop words, which doesn’t contain any words that can affect sentiment.

|  |  |
| --- | --- |
| **NLTK**  **stop words** | "i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for",  "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now" |
| **Custom stop words** | "i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your",  "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for",  "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there",  "when", "where", "why", "how", "all", "any", "both", "each", "few", "more",  "most", "other", "some", "such", "only", "own", "same", "so", "than", "too",  "very", "can", "will", "just", "should", "now" |

Figure 3.2 Most used Words

**CHAPTER 4**

# DESIGN METHODOLOGY

## 4.1 FEATURE EXTRACTION

We extract two types of features from our dataset, namely unigrams and bigrams. We create a frequency distribution of the unigrams and bigrams present in the dataset and choose top N unigrams and bigrams for our analysis.

**Unigrams:**

Probably the simplest and the most commonly used features for text classification is the presence of single words or tokens in the text. We extract single words from the training dataset and create a frequency distribution of these words.

**Bigrams:**

Bigrams are word pairs in the dataset which occur in succession in the corpus. These features are a good way to model negation in natural language like in the phrase – This is not good. A total of 1954953 unique bigrams were extracted from the dataset. Out of these, most of the bigrams at end of frequency spectrum are noise and occur very few times to influence classification. We therefore use only top 10000 bigrams from these to create our vocabulary.

**N-Grams:**

We extract three types of features from our dataset, namely unigrams bigrams and trigrams. N-grams are the combination of multiple words used together. Ngrams with N=1 are called unigrams. Similarly, bigrams (N=2), trigrams (N=3) and so on can also be used. Unigrams do not usually contain as much information as compared to bigrams and trigrams.

The basic principle behind n-grams is that they capture the language structure, like what letter or word is likely to follow the given one.

The longer the n-gram (the higher the n), the more context we have to work with. Optimum length really depends on the application – if our n-grams are too short, we may fail to capture important differences.

**Bag of Words:**

Bag of Words (BoW) refers to the representation of text which describes the presence of words within the text data. The intuition behind this is that two similar text fields will contain similar kind of words, and will therefore have a similar bag of words. Further, that from the text alone we can learn something about the meaning of the document. We use CountVectorizer from sklearn for this purpose.

**Term Frequency – Inverse Document Frequency:**

Term frequency is simply the ratio of the count of a word present in a sentence, to the length of the sentence. Therefore, we can generalize term frequency as:TF = (Number of times term T appears in the particular row) / (number of terms in that row)

The intuition behind inverse document frequency (IDF) is that a word is not of much use to us if it’s appearing in all the documents. Therefore, the IDF of each word is the log of the ratio of the total number of rows to the number of rows in which that word is present.

IDF = log (N/n), where, N is the total number of rows and n is the number of rows in which the word was present.TF-IDF is the multiplication of the TF and IDF which we calculated above. TF-IDF has penalized words like ‘don’t’, ‘can’t’, and ‘use’ because they are commonly occurring words. However, it has given a high weight to “disappointed” since that will be very useful in determining the sentiment of the tweet.

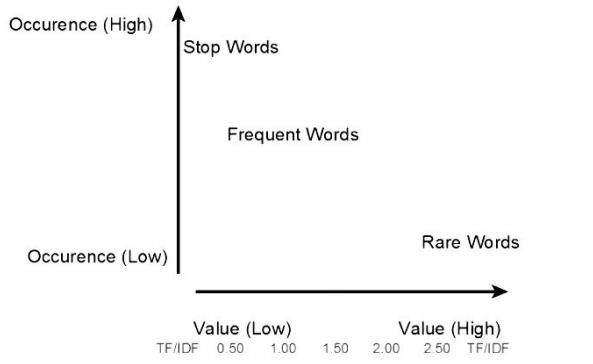


Figure 4.1 Term Frequency

**Most Used Words**



Figure 4.2 Most Used Words

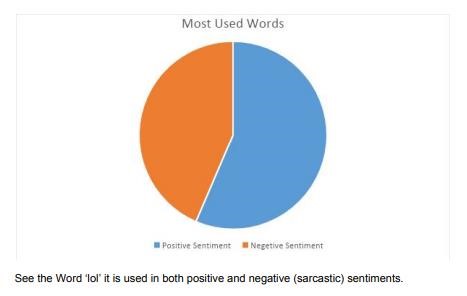


Figure 4.3 Piechart

## 4.2 SPLITTING DATASET

Spitting the dataset into train and test data for cross validation. We have split the dataset into 8:2 ratio of (train : test) data randomly. X\_train\_shape : (79991, 1195309) X\_test\_shape : (19998, 1195309) y\_train\_shape : (79991,) y\_test\_shape : (19998,)

**Naive Bayes Classifier**

It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes Theorem for events A and B, provided that P (B ) ≠ 0,

P (A | B) = P(B) P(B | A) P(A)

* P (A|B) is the posterior probability of class (c, target) given predictor (x, attributes).
* P (A) is the prior probability of class.
* P (B|A) is the likelihood which is the probability of predictor given class.
* P (B) is the prior probability of predictor.

**Pros:**

* It is easy and fast to predict class of test data set. It also perform well in multi class prediction
* When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and we need less training data.

**Cons:**

* If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”.To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
* On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict\_proba are not to be taken too seriously.
* Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

**Naive Bayes models:**

* **Gaussian:**

It is used in classification and it assumes that features follow a normal distribution.

* **Multinomial:**

It is used for discrete counts. For example, let’s say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of “word occurring in the document”, we have “count how often word occurs in the document”, we can think of it as “number of times outcome number x\_i is observed over the n trials”.

* **Bernoulli:**

The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with ‘bag of words’ model where the 1s& 0s are “word occurs in the document” and “word does not occur in the document” respectively.

Here we will be using Multinomial Classification.

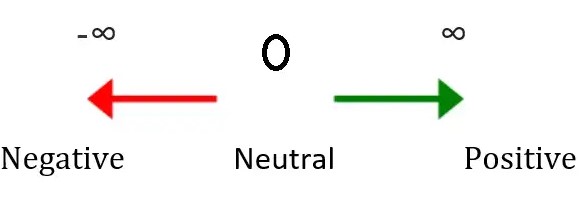


Figure 4.4 Naive Bayes Model

**CHAPTER 5**

# IMPLEMENTATION

## 5.1 PRECISION RECALL AND ACCURACY

* Precision measures how many texts were predicted correctly as belonging to a given category out of all of the texts that were predicted (correctly and incorrectly) as belonging to the category.
* Recall measures how many texts were predicted correctly as belonging to a given category out of all the texts that should have been predicted as belonging to the category. We also know that the more data we feed our classifiers with, the better recall will be.
* Accuracy measures how many texts were predicted correctly (both as belonging to a category and not belonging to the category) out of all of the texts in the corpus. Most frequently, precision and recall are used to measure performance since accuracy alone does not say much about how good or bad a classifier is.

## 5.2 CONFUSION MATRIX

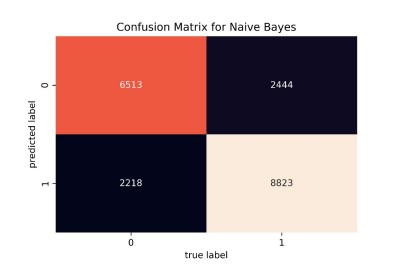
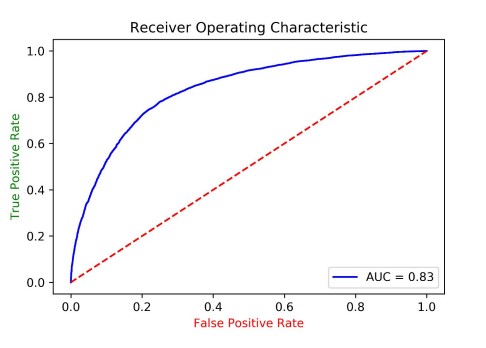


Figure 5.1 Confusion Matrix

**Classification Report**

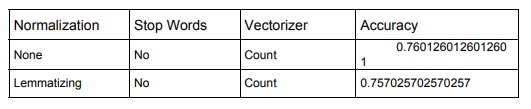
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | **0.73** | **0.75** | **0.74** | **8731** |
| **1** | **0.80** | **0.78** | **0.79** | **11267** |
| **Accuracy** |  |  | **0.77** | **19998** |
| **Micro avg** | **0.76** | **0.76** | **0.76** | **19998** |
| **Weighted avg** | **0.77** | **0.77** | **0.77** | **19998** |

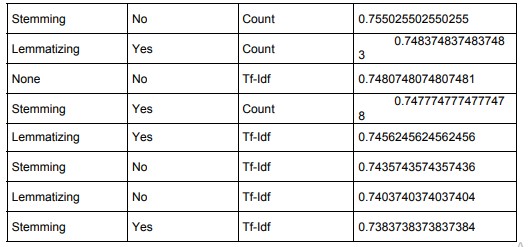
## 5.3 ROC CURVE



Area Under the Curve of anything greater than 0.80 with real life data is not bad. We achieved 0.83. Figure 5.2ROC Graph

## 5.4 COMPARISON (UNIGRAM ONLY)

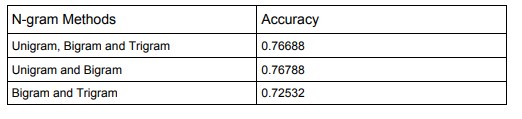




So here we see the best combination for sentiment analysis with this dataset is using

* Without normalization
* Without stop words
* Count Vectorizer

For document analysis the best combination will include lemmatization, stop words, but in sentiment analysis, they aren't providing edge over the raw datas. **Accuracy (N-grams)**



# CHAPTER 6

**CONCLUSION AND FUTURE ENHANCEMENT**

## 6.1 CONCLUSION

**Achievements**

The provided tweets were a mixture of words, emoticons, URLs, hashtags, user mentions, and symbols. Before training the dataset we pre-processed the tweets to make it suitable for feeding into models. We implemented machine learning algorithms Naive Bayes Classifier to classify the polarity of the tweet. We used three types of features namely unigrams, bigrams and trigrams for classification and observed that augmenting the feature vector with bigrams improved the accuracy and trigrams degraded the accuracy. Once the feature has been extracted it was represented as a sparse vector. It has been observed that presence in the sparse vector representation recorded a better performance than frequency. We finally achieved an accuracy of 76.78%

With unigram and bigram, without any normalization, stop words or tf-idf

Our model accuracy : 0.76788

Leaderboard highest : 0.79249

**6.2 FUTURE ENHANCEMENT**

* Handling emotion ranges: We can improve and train our models to handle a range of sentiments. Tweets don’t always have positive or negative sentiment. At times they may have no sentiment i.e. neutral. Sentiment can also have gradations like the sentence, ‘This is good’, is positive but the sentence, ‘This is extraordinary’, is somewhat more positive than the first. We can therefore classify the sentiment in ranges, say from -2 to +2.
* Using hashtags and other symbols like ‘?’ and ‘!’: During our pre-processing, we discard most of the symbols like hashtags, question marks and exclamation marks. These symbols may be helpful in assigning sentiment to a sentence.

Text Processing and Sentiment analysis emerges as a challenging field with lots of obstacles as it involves natural language processing.

**CHAPTER 7**

## REFERENCE

1. Bibi, M.; Abbasi, W.A.; Aziz, W.; Khalil, S.; Uddin, M.; Iwendi, C.; Gadekallu, T.R. A novel unsupervised ensemble framework using concept-based linguistic methods and machine learning for twitter sentiment analysis. Pattern Recognit. Lett. 2022, 158, 80–86. [CrossRef]
2. A Step-By-Step Guide to Getting Started on Twitter. Available online: http://img.constantcontact.com/docs/pdf/getting-startedon-twitter.pdf (accessed on 16 March 2021).
3. S. Rosenthal, N. Farra, and P. Nakov, “SemEval-2017 task 4: Sentiment analysis in twitter,” in Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 502–518.

1. Bose, D.; Aithal, P.S.; Roy, S. Survey of Twitter Viewpoint on Application of Drugs by

VADER Sentiment Analysis among Distinct Countries. Int. J. Manag. Technol. Soc. Sci. 2021, 6, 110–127. [CrossRef]

1. Ginossar, T.; Cruickshank, I.J.; Zheleva, E.; Sulskis, J.; Berger-Wolf, T. Cross-platform spread: Vaccine-related content, sources, and conspiracy theories in YouTube videos shared in early Twitter COVID-19 conversations. Hum. Vaccines Immunother. 2022, 18, 1–13. [CrossRef]
2. Poria, S., Cambria, E., & Gelbukh, A. (2015). Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In Proceed ings of the 2015 Conference on Empirical Methods in Natural Language Processing (pp. 2539- 2544).
3. TextBlob, 2017, https://textblob.readthedocs.io/en/dev/
4. Statista,, https://www.statista.com/statistics/282087/number-ofmonthly-active-twitter-users/
5. M. Cliche, “BB twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs,” in Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval2017). Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 573–580.
6. D. Mottl and J. Henriquel, “Getoldtweets3,” 2019. [Online]. Available:

<https://pypi.org/project/GetOldTweets3/>

# APPENDIX – 1

**SAMPLE CODE :**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

train = pd.read\_csv('drive/My Drive/Projects/Twitter Sentiment/train\_tweet.csv')

test = pd.read\_csv('drive/My Drive/Projects/Twitter Sentiment/test\_tweets.csv')

print(train.shape)

print(test.shape)

train.head()

test.head()

train.isnull().any()

test.isnull().any()

# checking out the negative comments from the train set

train[train['label'] == 0].head(10)

# checking out the postive comments from the train set

train[train['label'] == 1].head(10)

train['label'].value\_counts().plot.bar(color = 'pink', figsize = (6, 4))

# checking the distribution of tweets in the data

length\_train = train['tweet'].str.len().plot.hist(color = 'pink', figsize = (6, 4))

length\_test = test['tweet'].str.len().plot.hist(color = 'orange', figsize = (6, 4))

# adding a column to represent the length of the tweet

train['len'] = train['tweet'].str.len()

test['len'] = test['tweet'].str.len()

train.head(10)

train.groupby('label').describe()

train.groupby('len').mean()['label'].plot.hist(color = 'black', figsize = (6, 4),)

plt.title('variation of length')

plt.xlabel('Length')

plt.show()

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(stop\_words = 'english')

words = cv.fit\_transform(train.tweet)

sum\_words = words.sum(axis=0)

words\_freq = [(word, sum\_words[0, i]) for word, i in cv.vocabulary\_.items()]

words\_freq = sorted(words\_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words\_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')

plt.title("Most Frequently Occuring Words - Top 30")

from wordcloud import WordCloud

wordcloud = WordCloud(background\_color = 'white', width = 1000, height = 1000).generate\_from\_frequencies(dict(words\_freq))

plt.figure(figsize=(10,8))

plt.imshow(wordcloud)

plt.title("WordCloud - Vocabulary from Reviews", fontsize = 22)

normal\_words =' '.join([text for text in train['tweet'][train['label'] == 0]])

wordcloud = WordCloud(width=800, height=500, random\_state = 0, max\_font\_size = 110).generate(normal\_words)

plt.figure(figsize=(10, 7))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis('off')

plt.title('The Neutral Words')

plt.show()

negative\_words =' '.join([text for text in train['tweet'][train['label'] == 1]])

wordcloud = WordCloud(background\_color = 'cyan', width=800, height=500, random\_state = 0, max\_font\_size = 110).generate(negative\_words)

plt.figure(figsize=(10, 7))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis('off')

plt.title('The Negative Words')

plt.show()

# collecting the hashtags

def hashtag\_extract(x):

hashtags = []

for i in x:

ht = re.findall(r"#(\w+)", i)

hashtags.append(ht)

return hashtags

# extracting hashtags from non racist/sexist tweets

HT\_regular = hashtag\_extract(train['tweet'][train['label'] == 0])

# extracting hashtags from racist/sexist tweets

HT\_negative = hashtag\_extract(train['tweet'][train['label'] == 1])

# unnesting list

HT\_regular = sum(HT\_regular,[])

HT\_negative = sum(HT\_negative,[])

a = nltk.FreqDist(HT\_regular)

d = pd.DataFrame({'Hashtag': list(a.keys()),

'Count': list(a.values())})

# selecting top 20 most frequent hashtags

d = d.nlargest(columns="Count", n = 20)

plt.figure(figsize=(16,5))

ax = sns.barplot(data=d, x= "Hashtag", y = "Count")

ax.set(ylabel = 'Count')

plt.show()

a = nltk.FreqDist(HT\_negative)

d = pd.DataFrame({'Hashtag': list(a.keys()),

'Count': list(a.values())})

# selecting top 20 most frequent hashtags

d = d.nlargest(columns="Count", n = 20)

plt.figure(figsize=(16,5))

ax = sns.barplot(data=d, x= "Hashtag", y = "Count")

ax.set(ylabel = 'Count')

plt.show()

# tokenizing the words present in the training set

tokenized\_tweet = train['tweet'].apply(lambda x: x.split())

# importing gensim

import gensim

# creating a word to vector model

model\_w2v = gensim.models.Word2Vec(

tokenized\_tweet,

vector\_size=200, # desired no. of features/independent variables

window=5, # context window size

min\_count=2,

sg = 1, # 1 for skip-gram model

hs = 0,

negative = 10, # for negative sampling

workers= 2, # no.of cores

seed = 34)

model\_w2v.train(tokenized\_tweet, total\_examples= len(train['tweet']), epochs=20)

model\_w2v.wv.most\_similar(positive = "dinner")

model\_w2v.wv.most\_similar(positive = "cancer")

model\_w2v.wv.most\_similar(positive = "apple")

model\_w2v.wv.most\_similar(negative = "hate")

# removing unwanted patterns from the data

import re

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

train\_corpus = []

for i in range(0, 31962):

review = re.sub('[^a-zA-Z]', ' ', train['tweet'][i])

review = review.lower()

review = review.split()

ps = PorterStemmer()

# stemming

review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]

# joining them back with space

review = ' '.join(review)

train\_corpus.append(review)

test\_corpus = []

for i in range(0, 17197):

review = re.sub('[^a-zA-Z]', ' ', test['tweet'][i])

review = review.lower()

review = review.split()

ps = PorterStemmer()

# stemming

review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]

# joining them back with space

review = ' '.join(review)

test\_corpus.append(review)

# creating bag of words

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features = 2500)

x = cv.fit\_transform(train\_corpus).toarray()

y = train.iloc[:, 1]

print(x.shape)

print(y.shape)

# creating bag of words

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features = 2500)

x\_test = cv.fit\_transform(test\_corpus).toarray()

print(x\_test.shape)

# splitting the training data into train and valid sets

from sklearn.model\_selection import train\_test\_split

x\_train, x\_valid, y\_train, y\_valid = train\_test\_split(x, y, test\_size = 0.25, random\_state = 42)

print(x\_train.shape)

print(x\_valid.shape)

print(y\_train.shape)

print(y\_valid.shape)

# standardization

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_valid = sc.transform(x\_valid)

x\_test = sc.transform(x\_test)

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import f1\_score

model = RandomForestClassifier()

model.fit(x\_train, y\_train)

y\_pred = model.predict(x\_valid)

print("Training Accuracy :", model.score(x\_train, y\_train))

print("Validation Accuracy :", model.score(x\_valid, y\_valid))

# calculating the f1 score for the validation set

print("F1 score :", f1\_score(y\_valid, y\_pred))

# confusion matrix

cm = confusion\_matrix(y\_valid, y\_pred)

print(cm)

from sklearn.svm import SVC

model = SVC()

model.fit(x\_train, y\_train)

y\_pred = model.predict(x\_valid)

print("Training Accuracy :", model.score(x\_train, y\_train))

print("Validation Accuracy :", model.score(x\_valid, y\_valid))

# calculating the f1 score for the validation set

print("f1 score :", f1\_score(y\_valid, y\_pred))

# confusion matrix

cm = confusion\_matrix(y\_valid, y\_pred)

print(cm)

from xgboost import XGBClassifier

model = XGBClassifier()

model.fit(x\_train, y\_train)

y\_pred = model.predict(x\_valid)

print("Training Accuracy :", model.score(x\_train, y\_train))

print("Validation Accuracy :", model.score(x\_valid, y\_valid))

# calculating the f1 score for the validation set

print("f1 score :", f1\_score(y\_valid, y\_pred))

# confusion matrix

cm = confusion\_matrix(y\_valid, y\_pred)

print(cm)

**APPENDIX – 2**

**SCREENSHOTS**

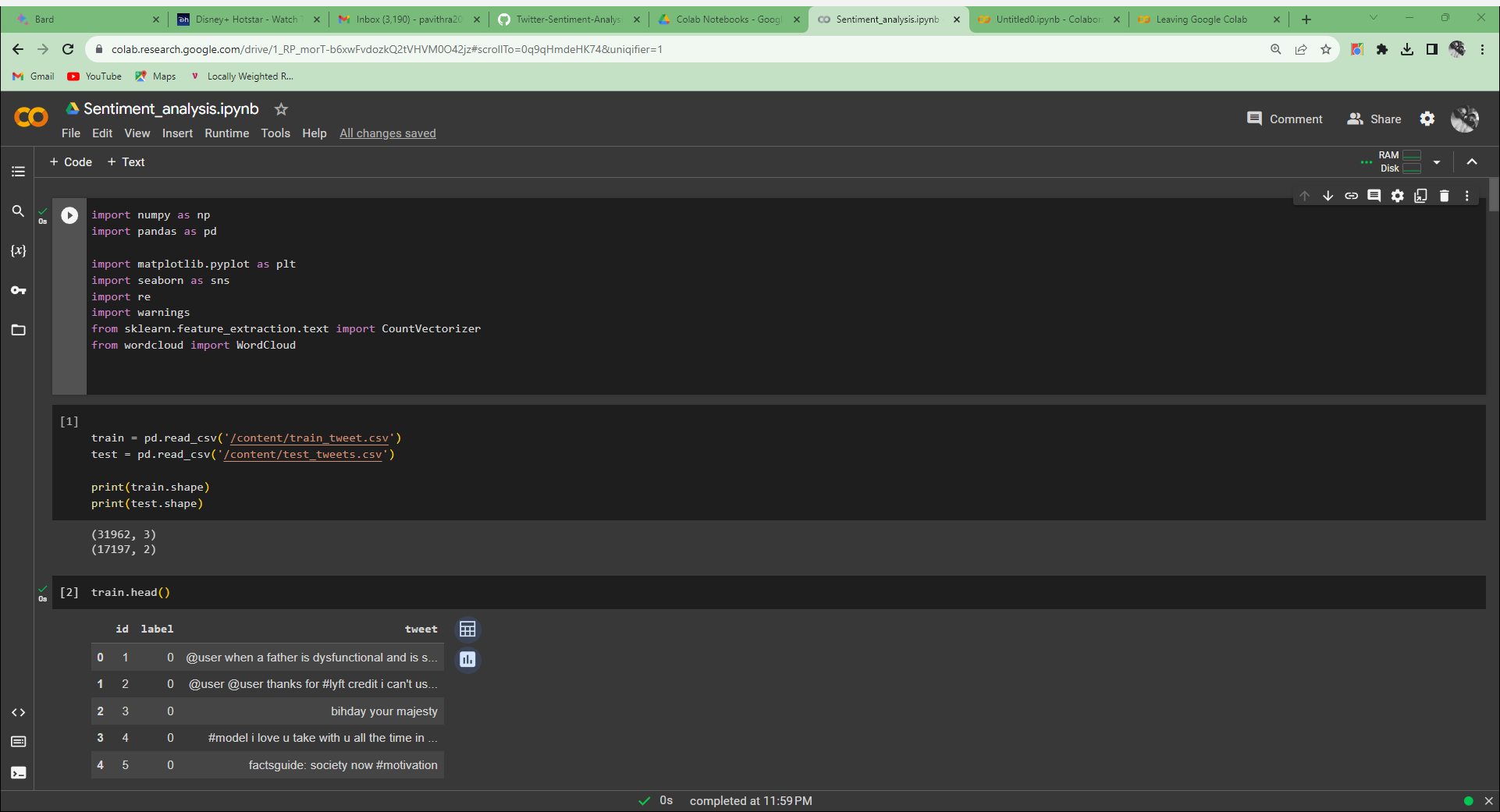


Figure A.2.1 Loading Dataset

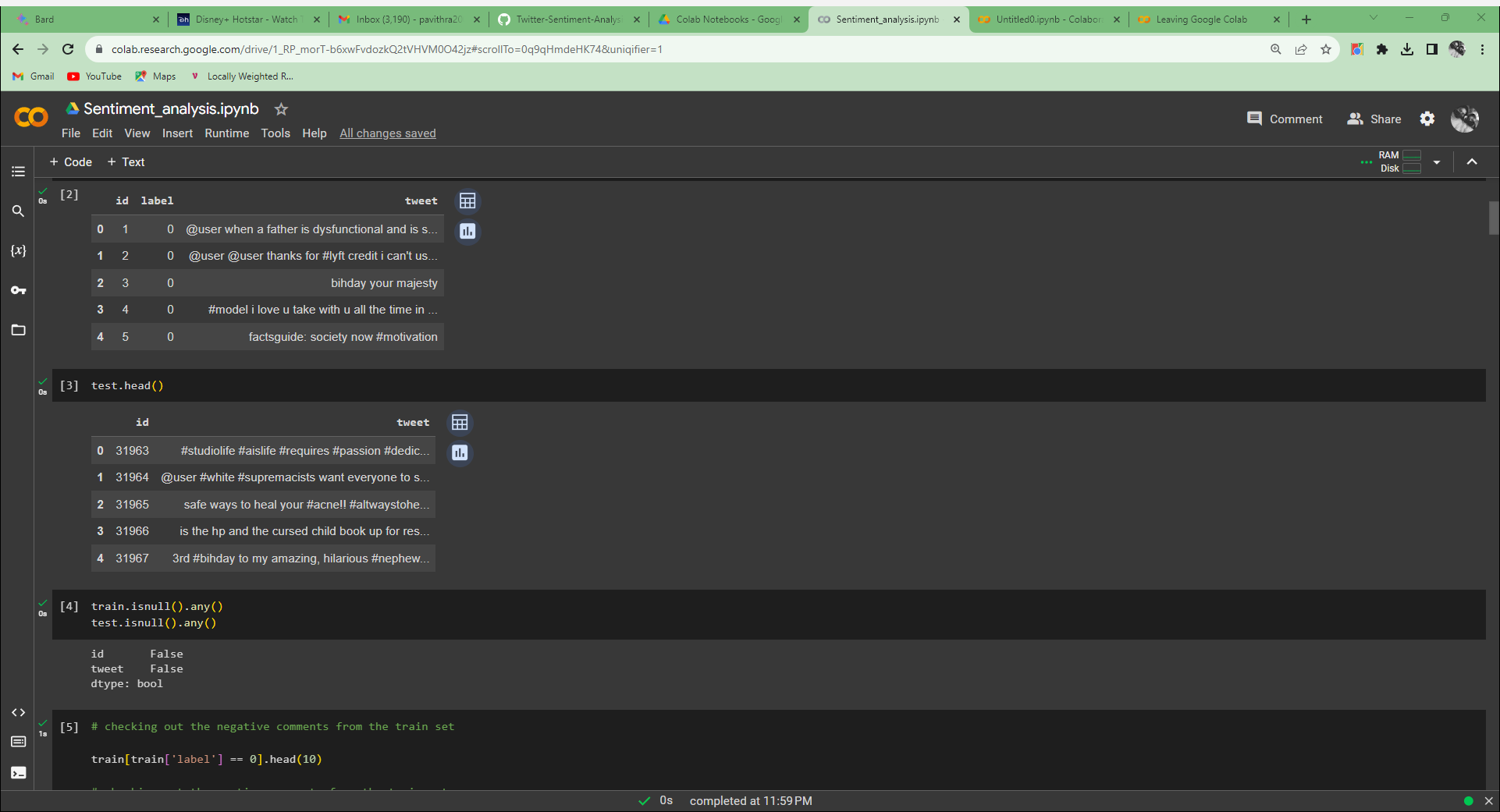


Figure A.2.2 Extraction Process

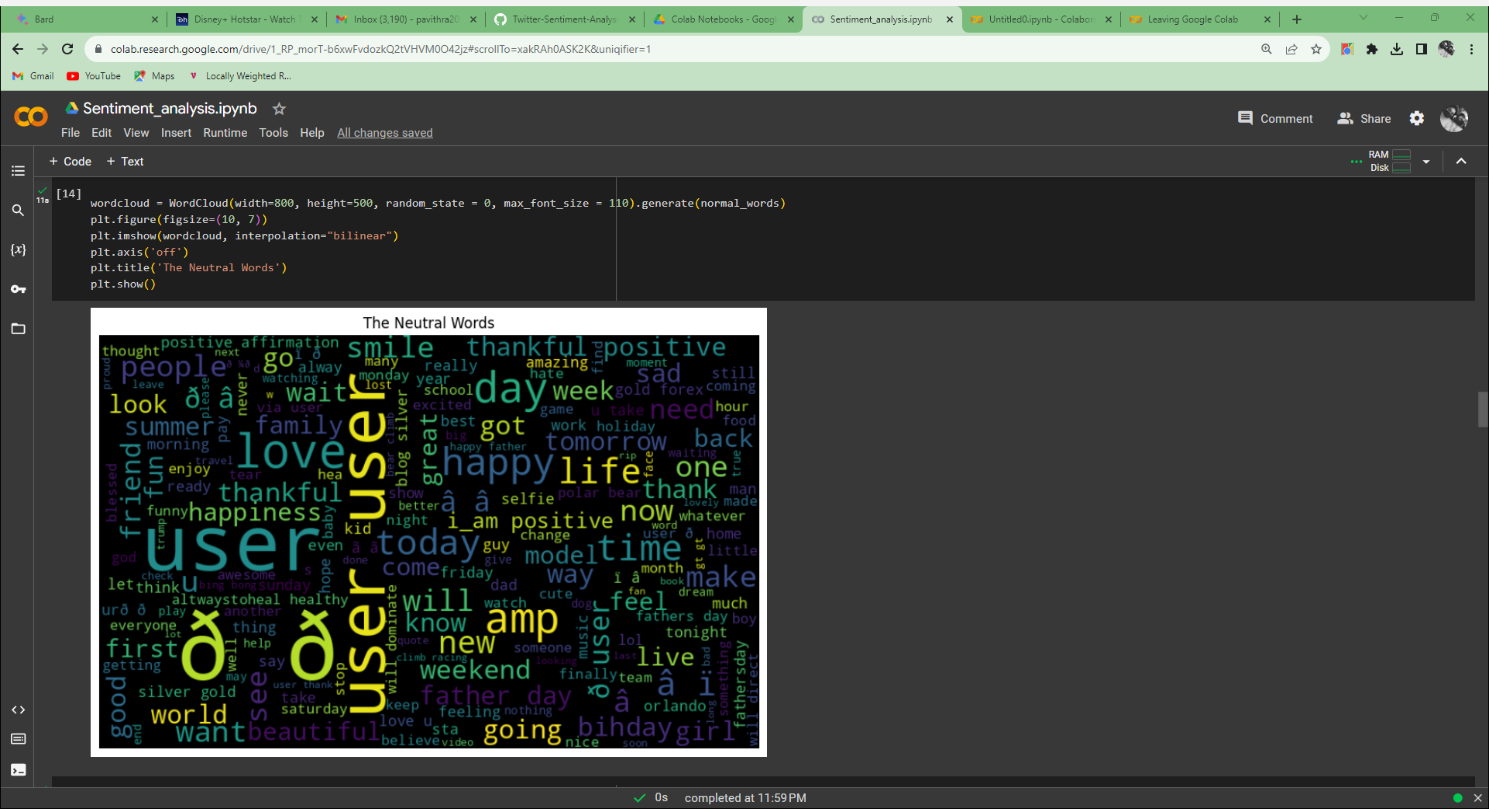


Figure A.2.3 Most used positive words

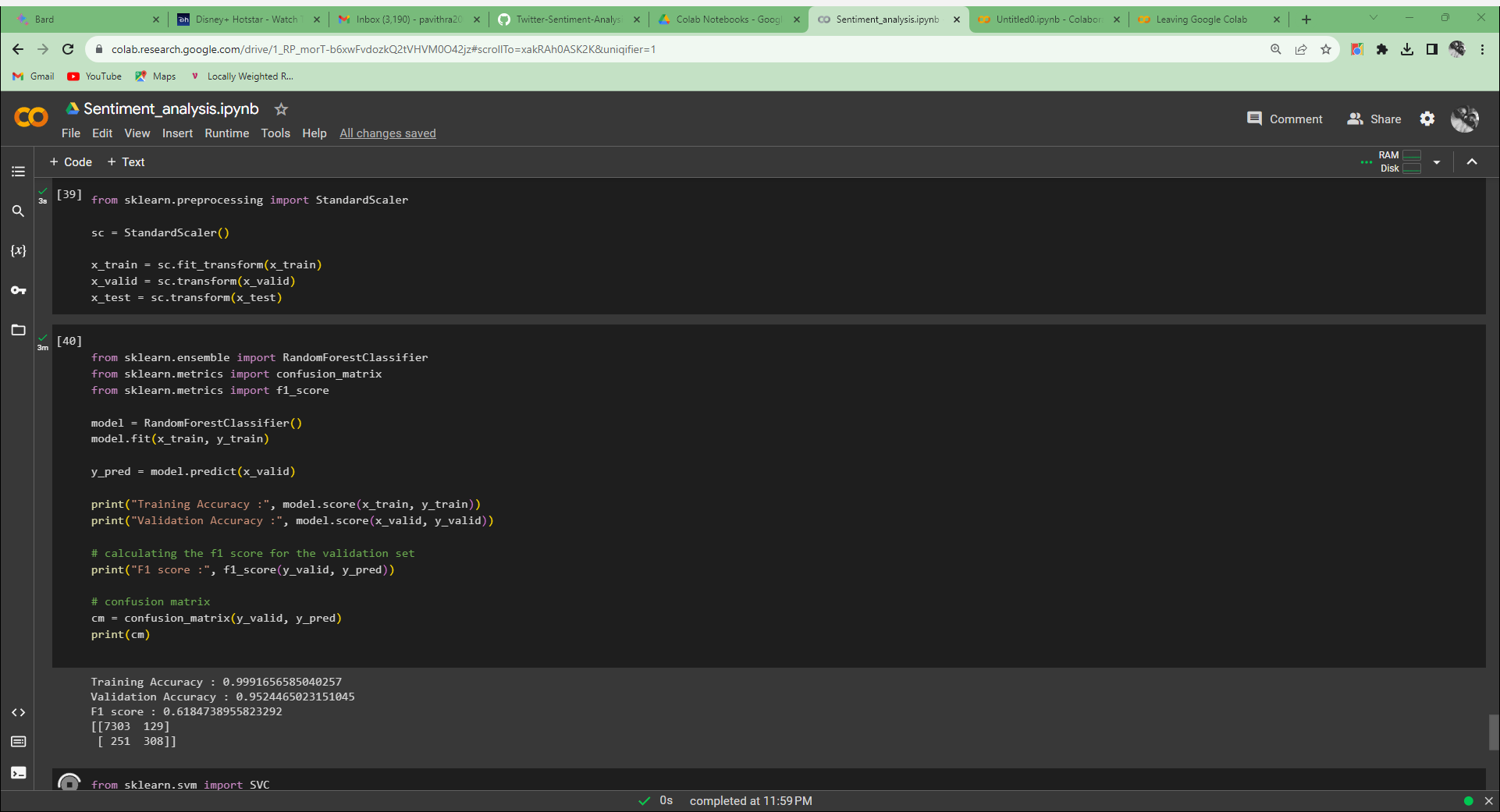


Figure A.2.4 Confusion Matrix

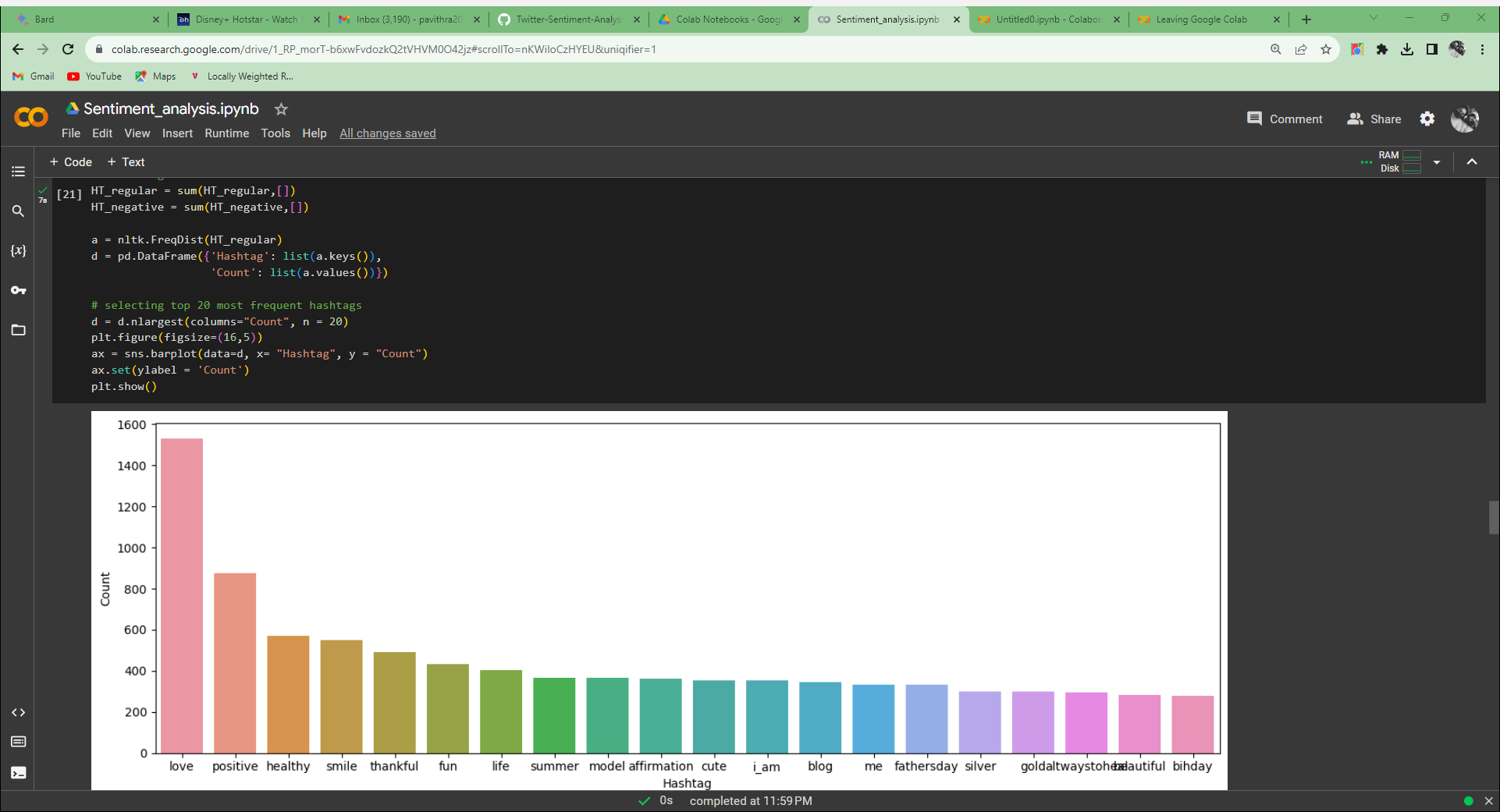


Figure A.2.5 Classification