Birla Institute of Technology

Project

On

Sentiment Analysis using NLP

Supervised by :-

Pf. Rashmi Rathi Upadhyay

(professor, Department of Computer Applications, Birla Institute of Technology,Mesra,Ranchi)

submitted by:-

Manisha kiran

(MCA/10034/19)

Sem-IV

Abstract

This technical report collects the final report of the Sentiment analysis using NLP from MANISHA KIRAN student of BIT MESRA RANCHI . These projects have spanned the students' entire semester, during which they have worked closely with their faculty advisors to plan and carry out their projects. This work counts as of academic credit of semester. In addition to doing the research, the students presented a brief midterm progress report of semester, presented an oral summary in the year-end campuswide Meeting of the Minds and submitted a written project in April.

Contents

Acknowledgements

I wish to offer my gratitude to the advisors of this work, Prof. RASHMI RATHI UPADHYAY for letting me use their experience in building this system. It is to their credit that this research turned out to be state of the art. I would also like to give due credit to Dr. vandana who initiated this research as part of project and encouraged to pursue this project beyond a course. I would like to thank my friends who guided me selflessly throughout this project. My weekly meetings with them ensured steady progress of this work and helped me achieve the high system performance on time. Also, I am grateful to my friends for motivating me throughout. This work wouldn’t have been possible without the support of my parents and friends who helped me a lot.

Literature Review

One of the first popular work in the field of Twitter sentiment analysis was done by Go et al 2009. . The work was primarily conducted as a project work that turned into research publication. The research uses distant supervision technique to overcome the problem of manual annotation of large set of tweets. Tweets with ":)" emoticon were considered positive while tweets with ":(" in the message were considered negative. This resulted in two defects. First, emoticons, which were considered important by other works in the area (see below) couldn't be used to learn sentiments. Secondly, the authors are unsure if all tweets with ":)" are truly positive or can contain negative or sarcastic sentiments too. Therefore, the dataset used in the experiment was labelled noisy. The task of classification was limited to positive and negative though the need for neutral class was highlighted in the end. Go et al. (2009) used SVM, MaxEnt and Naive Bayes and reported that SVM and Naive Bayes were equally good and beat MaxEnt. The research also found POS (parts of speech) tags were not helpful for their purpose. However, they only tried general purpose POS 4 tagger which is shown to have inaccuracies when ran on microblogging text. Another popular work in this topic was conducted by Pak and Paroubek (2010). The same distant supervision procedure of tagging tweets positive or negative based on the emoticons it mentions was used. However, highlighting the significance of neutral class, the team also collected neutral tweets. These tweets were strictly objective and were collected from newspapers and magazines. Unlike Go et al. (2009), POS features were deemed useful. However, TreeTagger (Schmid, 1995) was used which wasn't designed to work with microblogging text like tweets. The research didn't use auto annotated tweets in the test set. 5 Instead, a small size test set was hand annotated (216 tweets). Like Go et al. (2009), Pak and Paroubek (2010) also used linear kernel SVM to run the experiment. The results were not reported as accuracy metric reported by Go was different and hence the results were not comparable to previous research work. Major boost in the discussed field came as a result of annual SemEval workshop (Nakov et al., 2013). The 6 workshop had more than 30 entries in 2013 when the organizers first introduced twitter sentiment analysis as one of their exercises. Not only this provided opportunity for more frequent research in the area, it provided large dataset of hand annotated tweets with all three sentiment classes positive, negative and neutral . The consistent scoring metric was also available for the community to gauge their research with other teams in the field.

Introduction

As a human, we speak and write in English, Spanish or Chinese. But a computer's native language – known as machine code or machine language – is largely incomprehensible to most people. At our device's lowest levels, communication occurs not with words but through millions of zeros and ones that produce logical actions.

By combining the power of artificial intelligence, computational linguistics and computer science, Natural Language Processing (NLP) helps machines "read" text by simulating the human ability to understand language. NLP is everywhere even if we don't realize it. NLP is a way of computers to analyse, understand and derive meaning from a human languages such as English, Spanish, Hindi, etc.

Natural Language Processing (NLP) is a part of computer science and artificial intelligence which deals with human language. Natural language processing (NLP) is a major area of artificial intelligence research, which in its turn serves as a field of application and interaction of a number of other traditional AI areas. It is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyse large amounts of natural language data.

Natural language processing helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important. NLP helps developers to organize and structure knowledge to perform tasks like translation, summarization, named entity recognition, relationship extraction, speech recognition, topic segmentation, etc.

Today, a huge amount of information is available in online documents such as web pages, newsgroup postings, and on-line news databases. Among the myriad types of information available, one useful type is the *sentiment*, or *opinions* people express towards a *subject*.

Digital media represents a huge opportunity for businesses of any type to capture the opinions, needs and intent that users share on social media. In fact, the number of Google searches, WhatsApp messages and emails sent in 60 seconds is truly impressive (2,315,000 Google searches, 44,000,000 WhatsApp messages, more than 150,000,000 emails). Truly listening to a customer’s voice requires deeply understanding what they have expressed in natural language: Natural Language Processing (NLP) is the best way to understand the language used and uncover the sentiment behind it.

People often consider sentiment (in terms of positive or negative) as the most significant value of the opinions users express via social media. However, in reality emotions provide a richer set of information that address consumer choices and , in many cases, even determines their decisions. Because of this, Natural Language Processing for sentimental analysis focused emotions is extremely useful.

1. Natural Language Processing (NLP)

The field of study that focuses on the interactions between human language and computers is called Natural Language Processing or NLP for short. It sits at the intersection of computer science, artificial intelligence, and computational linguistics.

“Natural Language Processing is a field that covers computer understanding and manipulation of human language, and it’s ripe with possibilities for news gathering”

NLP is a way for computers to analyse, understand and derive meaning from human language in a smarter and useful way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentimental analysis, speech recognition, and topic segmentation.

“Apart from common word processor operations that treat text like a mere sequence of symbols, NLP considers the hierarchical structure of language: several words make a phrase, several phrases make a sentence and ultimately sentences convey ideas,” John Rehling , an NLP expert at Meltwater Group, said in ***How Natural Language Processing Helps Uncover Social Media Sentiment.*** “By analysing language for its meaning, NLP systems have long filled useful roles, such as correcting grammar, converting speech to text and automatically translating between languages.”

2. History of NLP

.Here, are important events in the history of Natural Language Processing:

1950- NLP started when Alan Turing published an article called "Machine and Intelligence."

The history of natural language processing (NLP) generally started in the 1950s, although work can be found from earlier periods. In 1950, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence

1950- Attempts to automate translation between Russian and English

The Georgetown experiment in 1954 involved fully automatic translation of more than sixty Russian sentences into English

1960- The work of Chomsky and others on formal language theory and generative syntax

1990- Probabilistic and data-driven models had become quite standard

2000- A Large amount of spoken and textual data become available

3. Why NLP is important

Large volumes of textual data

Natural language processing helps computers communicate with humans in their own language and scales other language-related tasks. For example, NLP makes it possible for computers to read text, hear speech, interpret it, measure sentiment and determine which parts are important.

Today’s machines can analyse more language-based data than humans, without fatigue and in a consistent, unbiased way. Considering the staggering amount of unstructured data that’s generated every day, from medical records to social media, automation will be critical to fully analyse text and speech data efficiently.

Structuring a highly unstructured data source

Human language is astoundingly complex and diverse. We express ourselves in infinite ways, both verbally and in writing. Not only are there hundreds of languages and dialects, but within each language is a unique set of grammar and syntax rules, terms and slang. When we write, we often misspell or abbreviate words, or omit punctuation. When we speak, we have regional accents, and we mumble, stutter and borrow terms from other languages. ‘

While supervised and unsupervised learning, and specifically deep learning, are now widely used for modelling human language, there’s also a need for syntactic and semantic understanding and domain expertise that are not necessarily present in these machine learning approaches. NLP is important because it helps resolve ambiguity in language and adds useful numeric structure to the data for many downstream applications, such as speech recognition or text analytics.

4. NLU and NLG

Systems are based on Natural Language Processing (NLP) and help humans as well as machines for communicating in natural language. Natural Language Understanding (NLU) and Natural Language Generation (NLG) are subsets of NLP.

NLU attempts to understand the meaning behind the written text. After having the speech recognition software convert speech into text, NLU software comes into the picture to decipher its meaning. It is quite possible that the same text has various meanings, or different words have the same meaning, or that the meaning changes with the context. Knowing the rules and structure of the language, understanding the text without ambiguity are some of the challenges faced by NLU systems. Popular applications include sentiment detection and profanity filtering among others. Google acquired API.ai provides tools for speech recognition and NLU.

NLG does exactly the opposite. Given the data, it analyses it and generates narratives in conversational language. It goes way beyond template-based systems, having been configured with the domain knowledge and experience of a human expert to produce well-researched, accurate output within seconds. Narratives can be generated for people across all hierarchical levels in an organization, in multiple languages.

5. Ambiguity in NLP

Ambiguity is an intrinsic characteristic of human conversations and one that is particularly challenging in natural language understanding (NLU) scenarios by ambiguity
Technically defining ambiguity can, well, ambiguous. However, there are different forms of ambiguity that are relevant in natural language and, consequently, in artificial intelligence (AI) systems.

There are different types of ambiguities

6.1 Lexical Ambiguity: is the ambiguity of a single word. A word can be ambiguous with respect to its syntactic class. Eg: book, study. For eg: The word silver can be used as a noun, an adjective, or a verb.
She bagged two silver medals.
She made a silver speech.
His worries had silvered his hair.

Lexical ambiguity can be resolved by Lexical category disambiguation i.e, parts-of-speech tagging. As many words may belong to more than one lexical category part-of-speech tagging is the process of assigning a part-of-speech or lexical category such as a noun, verb, pronoun, preposition, adverb, adjective etc. to each word in a sentence.

6.1.1 Lexical Semantic Ambiguity: The type of lexical ambiguity, which occurs when a single word is associated with multiple senses.

For eg: The tank was full of water.

I saw a military tank.

The occurrence of tank in both sentences corresponds to the syntactic category noun, but their meanings are different. Lexical Semantic ambiguity resolved using word sense disambiguation (WSD) techniques, where WSD aims at automatically assigning the meaning of the word in the context in a computational manner.

6.2 Syntactic Ambiguity: The structural ambiguities were syntactic ambiguities. Structural ambiguity is of two kinds: Scope Ambiguity and Attachment Ambiguity.

6.2.1 Scope Ambiguity: Scope ambiguity involves operators and quantifiers.

Consider the example: Old men and women were taken to safe locations. The scope of the adjective (i.e., the amount of text it qualifies) is ambiguous. That is, whether the structure (old men and women) or ((old men) and women)? The scope of quantifiers is often not clear and creates ambiguity.

Every man loves a woman.

The interpretations can be, For every man there is a woman and also it can be there is one particular woman who is loved by every man.

6.2.2 Attachment Ambiguity : A sentence has attachment ambiguity if a constituent fits more than one position in a parse tree. Attachment ambiguity arises from uncertainty of attaching a phrase or clause to a part of a sentence.

Consider the example: The man saw the girl with the telescope.

It is ambiguous whether the man saw a girl carrying a telescope, or he saw her through his telescope. The meaning is dependent on whether the preposition 'with' is attached to the girl or the man.

Consider the example: Buy books for children Preposition Phrase 'for children' can be either adverbial and attach to the verb buy or adjectival and attach to the object noun books.

6.3 Semantic Ambiguity: This occurs when the meaning of the words themselves can be misinterpreted. Even after the syntax and the meanings of the individual words have been resolved, there are two ways of reading the sentence.

Consider the example, Seema loves her mother and Sriya does too.

The interpretations can be Sriya loves Seema’s mother or Sriya likes her own mother. Semantic ambiguities born from the fact that generally a computer is not in a position to distinguishing what is logical from what is not.

Consider the example: The car hit the pole while it was moving.

The interpretations can be The car, while moving, hit the pole and The car hit the pole while the pole was moving. The first interpretation is preferred to the second one because we have a model of the world that helps us to distinguish what is logical (or possible) from what is not. To supply to a computer a model of the world is not so easy.

Consider the example: We saw his duck.

Duck can refer to the person’s bird or to a motion he made.

Semantic ambiguity happens when a sentence contains an ambiguous word or phrase.

6.4 Discourse : Discourse level processing needs a shared world or shared knowledge and the interpretation is carried out using this context. Anaphoric ambiguity comes under discourse level.

6.4.1 Anaphoric Ambiguity : Anaphors are the entities that have been previously introduced into the discourse.

Consider the example, The horse ran up the hill. It was very steep. It soon got tired.

The anaphoric reference of ‘it’ in the two situations cause ambiguity. Steep applies to surface hence ‘it’ can be hill. Tired applies to animate object hence ‘it’ can be horse.

6.5 Pragmatic Ambiguity : Pragmatic ambiguity refers to a situation where the context of a phrase gives it multiple interpretation. One of the hardest tasks in NLP. The problem involves processing user intention, sentiment, belief world, modals etc.- all of which are highly complex tasks.

Consider the example, Tourist (checking out of the hotel): Waiter, go upstairs to my room and see if my sandals are there; do not be late; I have to catch the train in 15 minutes. Waiter (running upstairs and coming back panting): Yes sir, they are there.

Clearly, the waiter is falling short of the expectation of the tourist, since he does not understand the pragmatics of the situation.

Pragmatic ambiguity arises when the statement is not specific, and the context does not provide the information needed to clarify the statement. Information is missing, and must be inferred.

Consider the example, I love you too.

This can be interpreted as I love you (just like you love me)

I love you (just like someone else does)

I love you (and I love someone else)

I love you (as well as liking you)

6. Steps in Natural Language Processing(NLP)

Step 1: Sentence Segmentation

The first step in the pipeline is to break the text apart into separate sentences. That gives us this:

1. “London is the capital and most populous city of England and the United Kingdom.”
2. “Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia.”
3. “It was founded by the Romans, who named it Londinium.”

We can assume that each sentence in English is a separate thought or idea. It will be a lot easier to write a program to understand a single sentence than to understand a whole paragraph.
Coding a Sentence Segmentation model can be as simple as splitting apart sentences whenever you see a punctuation mark. But modern NLP pipelines often use more complex techniques that work even when a document isn’t formatted cleanly.

Step 2: Word Tokenization

Now that we’ve split our document into sentences, we can process them one at a time. Let’s start with the first sentence from our document:

“London is the capital and most populous city of England and the United Kingdom.”

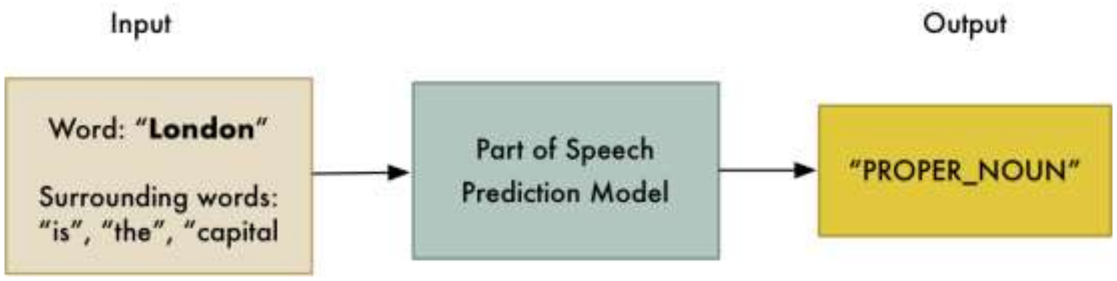
The next step in our pipeline is to break this sentence into separate words or *tokens*. This is called *tokenization*. This is the result:

“London”, “is”, “ the”, “capital”, “and”, “most”, “populous”, “city”, “of”, “England”, “and”, “the”, “United”, “Kingdom”, “.”

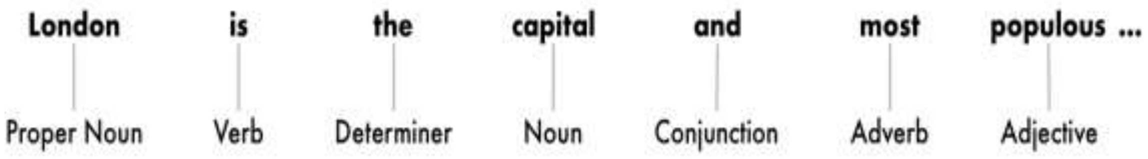
Tokenization is easy to do in English. We'll just split apart words whenever there's a space between them. And we'll also treat punctuation marks as separate tokens since punctuation also has meaning.

Step 3: Predicting Parts of Speech for Each Token

Next, we'll look at each token and try to guess its part of speech — whether it is a noun, a verb, an adjective and so on. Knowing the role of each word in the sentence will help us start to figure out what the sentence is talking about.
We can do this by feeding each word (and some extra words around it for context) into a pre-trained part-of-speech classification model:


Input
Output
Word: "London"
Surrounding words: "is", "the", "capital
Part of Speech Prediction Model
"PROPER_NOUN"

The part-of-speech model was originally trained by feeding it millions of English sentences with each word's part of speech already tagged and having it learn to replicate that behaviour.
Keep in mind that the model is completely based on statistics — it doesn't actually understand what the words mean in the same way that humans do. It just knows how to guess a part of speech based on similar sentences and words it has seen before.
After processing the whole sentence, we'll have a result like this:



With this information, we can already start to glean some very basic meaning. For example, we can see that the nouns in the sentence include "London" and "capital", so the sentence is probably talking about London.

Step 4: Text Lemmatization

In English (and most languages), words appear in different forms. Look at these two sentences:

I had a **pony**.

I had two **ponies**.

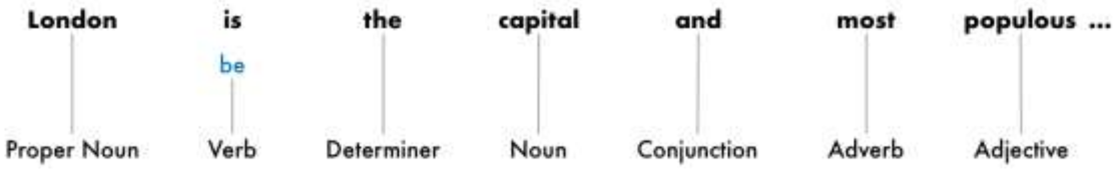
Both sentences talk about the noun **pony,** but they are using different inflections. When working with text in a computer, it is helpful to know the base form of each word so that you know that both sentences are talking about the same concept. Otherwise the strings "pony" and "ponies" look like two totally different words to a computer.

In NLP, we call finding this process *lemmatization* — figuring out the most basic form or *lemma* of each word in the sentence.

The same thing applies to verbs. We can also lemmatize verbs by finding their root, unconjugated form. So "**I had two ponies**" becomes "**I [have] two [pony].**"

Lemmatization is typically done by having a look-up table of the lemma forms of words based on their part of speech and possibly having some custom rules to handle words that you've never seen before.

Here's what our sentence looks like after lemmatization adds in the root form of our verb:

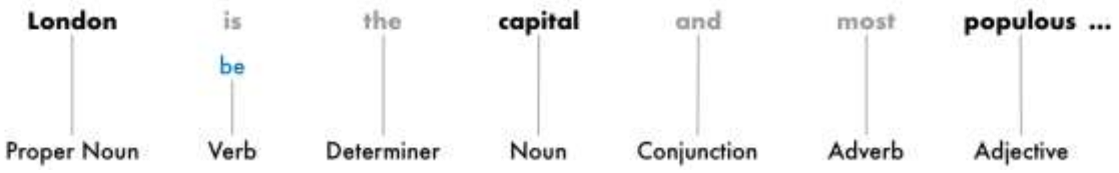


The only change we made was turning "is" into "be".

Step 5: Identifying Stop Words

Next, we want to consider the importance of a each word in the sentence. English has a lot of filler words that appear very frequently like "and", "the", and "a". When doing statistics on text, these words introduce a lot of noise since they appear way more frequently than other words. Some NLP pipelines will flag them as **stop words** —that is, words that you might want to filter out before doing any statistical analysis.

Here’s how our sentence looks with the stop words grayed out:



Stop words are usually identified by just by checking a hardcoded list of known stop words. But there’s no standard list of stop words that is appropriate for all applications. The list of words to ignore can vary depending on your application.
For example if you are building a rock band search engine, you want to make sure you don’t ignore the word “The”. Because not only does the word “The” appear in a lot of band names, there’s a famous 1980’s rock band called *The The*!

Step 6: Dependency Parsing

The next step is to figure out how all the words in our sentence relate to each other. This is called *dependency parsing*.
The goal is to build a tree that assigns a single **parent** word to each word in the sentence. The root of the tree will be the main verb in the sentence.

Step 7: Named Entity Recognition (NER)

The goal of *Named Entity Recognition*, or *NER*, is to detect and label these nouns with the real-world concepts that they represent.
Here are just some of the kinds of objects that a typical NER system can tag:

- People’s names
- Company names
- Geographic locations (Both physical and political)
- Product names

- Dates and times
- Amounts of money
- Names of events

Step 8: Coreference Resolution

At this point, we already have a useful representation of our sentence. We know the parts of speech for each word, how the words relate to each other and which words are talking about named entities.

However, we still have one big problem. English is full of pronouns — words like *he*, *she*, and *it*. These are shortcuts that we use instead of writing out names over and over in each sentence. Humans can keep track of what these words represent based on context. But our NLP model doesn't know what pronouns mean because it only examines one sentence at a time.

Let's look at the third sentence in our document:

"It was founded by the Romans, who named it Londinium."

If we parse this with our NLP pipeline, we'll know that "it" was founded by Romans. But it's a lot more useful to know that "London" was founded by Romans.

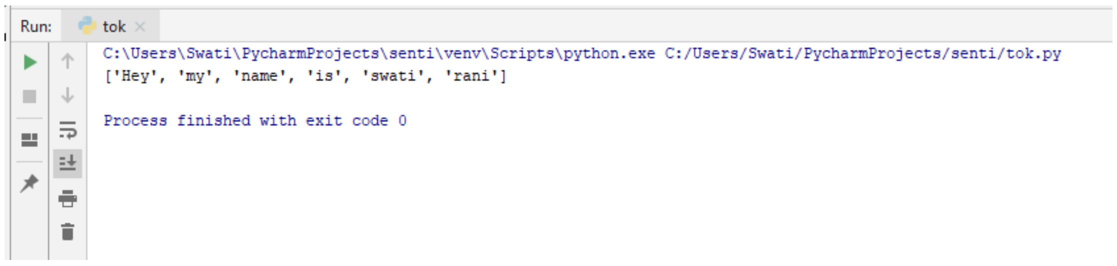
As a human reading this sentence, you can easily figure out that "*it"* means "*London"*. The goal of coreference resolution is to figure out this same mapping by tracking pronouns across sentences. We want to figure out all the words that are referring to the same entity.

7. Practical implementation

Tokenization

```
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
AI= ("Hey my name is swati rani ")
AI_tokens=word_tokenize(AI)
print(AI_tokens)
```

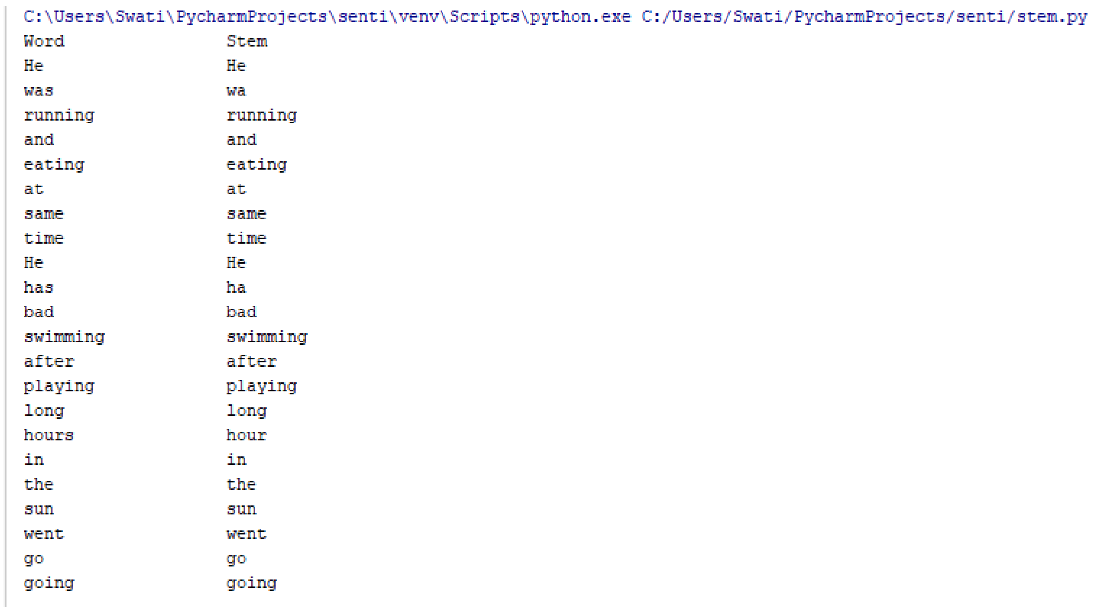
Output :



```
C:\Users\Swati\PycharmProjects\senti\venv\Scripts\python.exe C:/Users/Swati/PycharmProjects/senti/tok.py
['Hey', 'my', 'name', 'is', 'swati', 'rani']

Process finished with exit code 0
```

Stemming

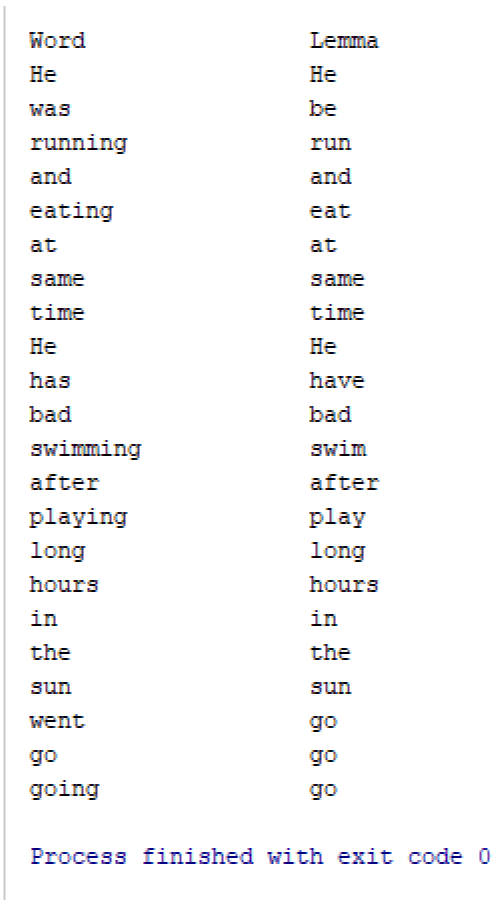
```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

sentence_1 = "He was running and eating at same time. He has bad
swimming after playing long hours in the sun went go going."
punctuations = "?:!.,;"

sentence_words = nltk.word_tokenize(sentence_1)

for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

sentence_words
print("{0:20}{1:20}".format("Word","Lemma"))
for word in sentence_words:
    print("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word)))
print()
```



```
C:\Users\Swati\PycharmProjects\senti\venv\Scripts\python.exe C:/Users/Swati/PycharmProjects/senti/stem.py
Word                Stem
He                  He
was                 wa
running             running
and                 and
eating              eating
at                  at
same                same
time                time
He                  He
has                 ha
bad                 bad
swimming            swimming
after               after
playing             playing
long                long
hours               hour
in                  in
the                 the
sun                 sun
went                went
go                  go
going               going
```

Lemmatization

```
import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

sentence_1 = "He was running and eating at same time. He has bad swimming after playing long hours in the sun went go going."
punctuations = "?:!.,;"

sentence_words = nltk.word_tokenize(sentence_1)

for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

sentence_words


print("{0:20}{1:20}".format("Word","Lemma"))
for word in sentence_words:


print("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word,pos="v")))
```

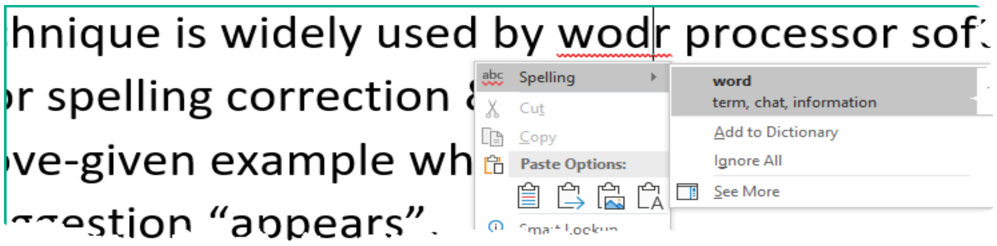


```
Word                Lemma
He                  He
was                 be
running             run
and                 and
eating              eat
at                  at
same                same
time                time
He                  He
has                 have
bad                 bad
swimming            swim
after               after
playing             play
long                long
hours               hours
in                  in
the                 the
sun                 sun
went                go
go                  go
going               go

Process finished with exit code 0
```

8. NLP Application

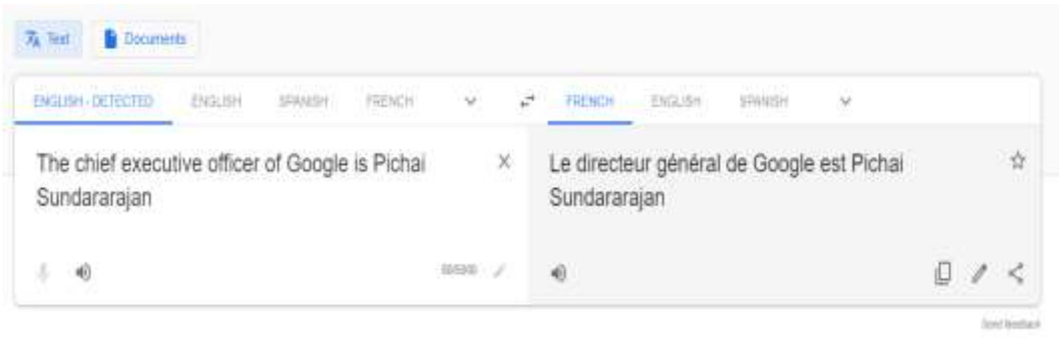
Here, are common Applications of NLP:

- **Information retrieval & Web Search -** Google, Yahoo, Bing, and other search engines base their machine translation technology on NLP deep learning models. It allows algorithms to read text on a webpage, interpret its meaning and translate it to another language.
- **Grammar Correction -** NLP technique is widely used by word processor software like MS-word for spelling correction & grammar check.



hnique is widely used by wodr processor sof
r spelling correction &
ve-given example wh
ggestion "appears".

- **Machine Translation -** Use of computer applications to translate text or speech from one natural language to another



The chief executive officer of Google is Pichai Sundararajan

Le directeur général de Google est Pichai Sundararajan

- **Sentiment analysis -** NLP helps companies to analyse a large number of reviews on a product. It also allows their customers to give a review of the particular product.

9. Sentimental Analysis

Today, a huge amount of information is available in online documents such as web pages, newsgroup postings, and on-line news databases. Among the myriad types of information available, one useful type is the *sentiment*, or *opinions* people express towards a *subject*. (A subject is either a topic of interest or a feature of the topic.) For example, knowing the reputation of their own or their competitors' products or brands is valuable for product development, marketing and consumer relationship management. Traditionally, companies conduct consumer surveys for this purpose. Though well-designed surveys can provide quality estimations, they can be costly especially if a large volume of survey data is gathered.

Sentiment Analysis is a technique used in text mining. It may, therefore, be described as a text mining technique for analysing the underlying sentiment of a text message, i.e., a tweet. Twitter sentiment or opinion expressed through it may be positive, negative or neutral.

Sentiment analysis of Twitter data may also depend upon sentence level and document level.

Methods like, positive and negative words to find on the sentence is however inappropriate, because the flavour of the text block depends a lot on the context. This may be done by looking at the POS (Part of Speech) Tagging.

10. Why Twitter Sentiment Analysis

Sentiment Analysis Dataset Twitter has a number of applications:

Business: Companies use Twitter Sentiment Analysis to develop their business strategies, to assess customers’ feelings towards products or brand, how people respond to their campaigns or product launches and also why consumers are not buying certain products.

Politics: In politics Sentiment Analysis Dataset Twitter is used to keep track of political views, to detect consistency and inconsistency between statements and actions at the government level. Sentiment Analysis Dataset Twitter is also used for analysing election results.

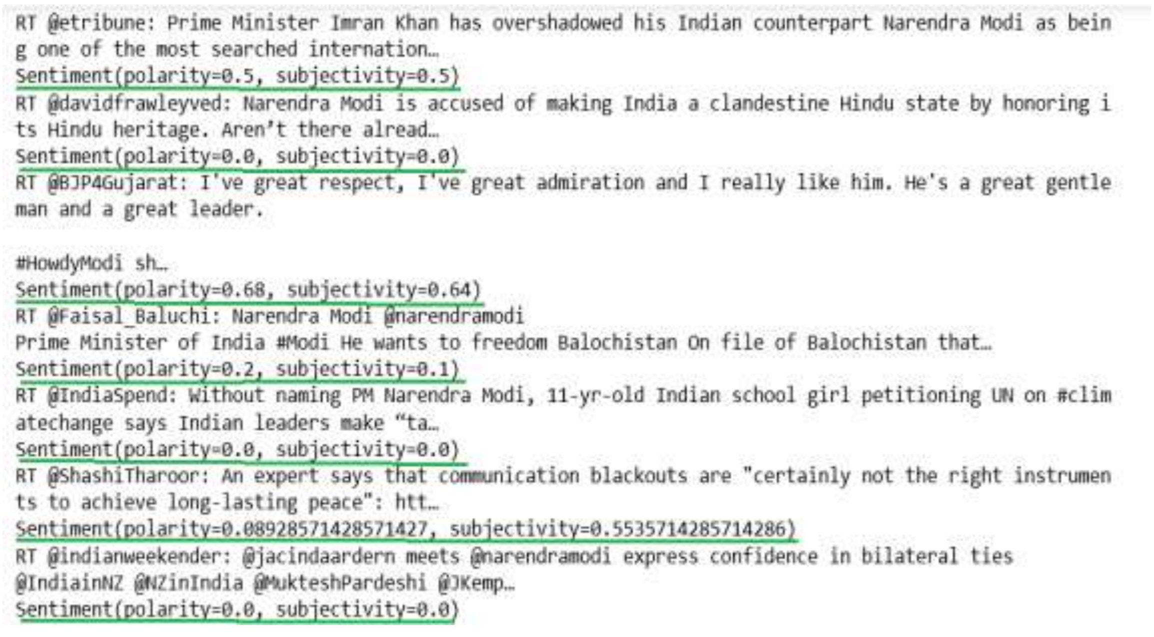
Public Actions: Twitter Sentiment Analysis also is used for monitoring and analysing social phenomena, for predicting potentially dangerous situations and determining the general mood of the blogosphere.

11. Program for twitter sentimental Analysis

Program 1

```
import tweepy
from textblob import TextBlob
consumer_key = "KXy7CJbu7lWOXzaoGd7S7anYn"
consumer_secret =
"JklFFiRYbzlPMMqJNs2JkIeB0IGONJa6lsu7Ze3FwCKGTVslgG"
access_token = "1067775738328289282-
3AoN30yB2VLnndTUCUYOQZtIJPRz1y"
access_token_secret =
"r6dXmM2kxrhe2REeFlKzUO04Rj7iHibyabmGp8vJADu5x"
auth = tweepy.OAuthHandler(consumer_key,consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

public_tweets = api.search('Narendra Modi')

for tweet in public_tweets:
    print(tweet.text)
    analysis = TextBlob(tweet.text)
    print(analysis.sentiment)
```

Output :

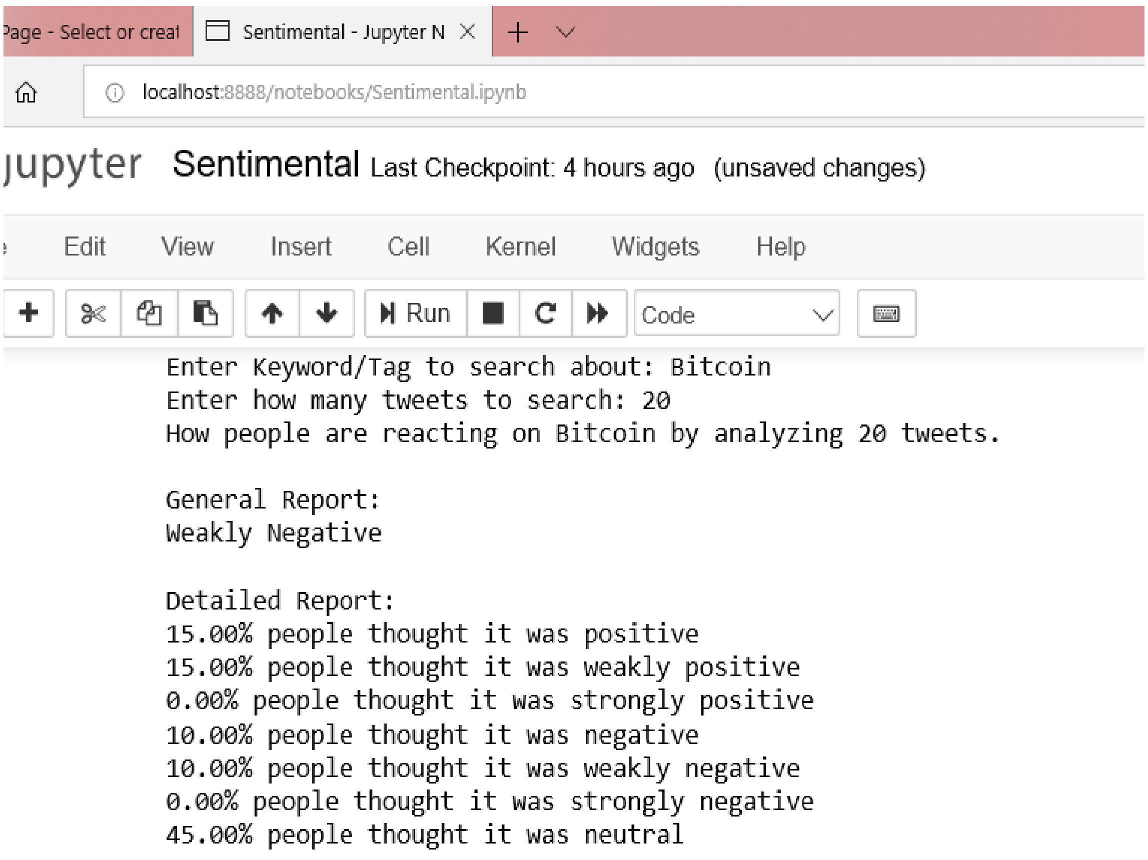


```
RT @etribune: Prime Minister Imran Khan has overshadowed his Indian counterpart Narendra Modi as bein
g one of the most searched internation…
Sentiment(polarity=0.5, subjectivity=0.5)
RT @davidfrawleyved: Narendra Modi is accused of making India a clandestine Hindu state by honoring i
ts Hindu heritage. Aren’t there alread…
Sentiment(polarity=0.0, subjectivity=0.0)
RT @BJP4Gujarat: I've great respect, I've great admiration and I really like him. He's a great gentle
man and a great leader.

#HowdyModi sh…
Sentiment(polarity=0.68, subjectivity=0.64)
RT @Faisal_Baluchi: Narendra Modi @narendramodi
Prime Minister of India #Modi He wants to freedom Balochistan On file of Balochistan that…
Sentiment(polarity=0.2, subjectivity=0.1)
RT @IndiaSpend: Without naming PM Narendra Modi, 11-yr-old Indian school girl petitioning UN on #clim
atechange says Indian leaders make “ta…
Sentiment(polarity=0.0, subjectivity=0.0)
RT @ShashiTharoor: An expert says that communication blackouts are "certainly not the right instrumen
ts to achieve long-lasting peace": htt…
Sentiment(polarity=0.08928571428571427, subjectivity=0.5535714285714286)
RT @indianweekender: @jacindaardern meets @narendramodi express confidence in bilateral ties
@IndiainNZ @NZinIndia @MukteshPardeshi @JKemp…
Sentiment(polarity=0.0, subjectivity=0.0)
```

Program 2

```
import sys, tweepy, csv, re
from textblob import TextBlob
import matplotlib.pyplot as plt


class SentimentAnalysis:

    def __init__(self):
        self.tweets = []
        self.tweetText = []

    def DownloadData(self):
        # authenticating
        consumerKey = 'L4pTFJjqhq966hbjoWrmw0cVL'
        consumerSecret =
'CCfnE1wVoAh8jvaQcTkHDVfSwVbnF9KsJK76xSxRBqSpeWvlqC'
        accessToken = '1067775738328289282-
5U2BTehLszRbv6Sr7WBZXA3f4bTuie'
        accessTokenSecret =
'1bXMCALhRjD1D2SVxpugq5xpt2uYZxOawdC75IeSQsf69'
        auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
        auth.set_access_token(accessToken, accessTokenSecret)
        api = tweepy.API(auth)

        # input for term to be searched and how many tweets to search
        searchTerm = input("Enter Keyword/Tag to search about: ")
        NoOfTerms = int(input("Enter how many tweets to search: "))

        # searching for tweets
        self.tweets = tweepy.Cursor(api.search, q=searchTerm,
lang="en").items(NoOfTerms)

        # Open/create a file to append data to
        csvFile = open('result.csv', 'a')

        # Use csv writer
        csvWriter = csv.writer(csvFile)

        # creating some variables to store info
        polarity = 0
        positive = 0
        wpositive = 0
        spositive = 0
        negative = 0
        wnegative = 0
        snegative = 0
        neutral = 0
        # iterating through tweets fetched
        for tweet in self.tweets:
            # Append to temp so that we can store in csv later. I use encode UTF-
8
            self.tweetText.append(self.cleanTweet(tweet.text).encode('utf-8'))
            # print (tweet.text.translate(non_bmp_map))    #print tweet's text
            analysis = TextBlob(tweet.text)
            # print(analysis.sentiment)  # print tweet's polarity
            polarity += analysis.sentiment.polarity  # adding up polarities to find
the average later

            if (analysis.sentiment.polarity == 0):  # adding reaction of how people
are reacting to find average later
                neutral += 1
            elif (analysis.sentiment.polarity > 0 and analysis.sentiment.polarity <=
0.3):
                wpositive += 1
            elif (analysis.sentiment.polarity > 0.3 and analysis.sentiment.polarity
<= 0.6):
                positive += 1
            elif (analysis.sentiment.polarity > 0.6 and analysis.sentiment.polarity
<= 1):
                spositive += 1
            elif (analysis.sentiment.polarity > -0.3 and analysis.sentiment.polarity
<= 0):
                wnegative += 1
            elif (analysis.sentiment.polarity > -0.6 and analysis.sentiment.polarity
<= -0.3):
                negative += 1
            elif (analysis.sentiment.polarity > -1 and analysis.sentiment.polarity
<= -0.6):
                snegative += 1

        # Write to csv and close csv file
        csvWriter.writerow(self.tweetText)
        csvFile.close()

        # finding average of how people are reacting
        positive = self.percentage(positive, NoOfTerms)
        wpositive = self.percentage(wpositive, NoOfTerms)
        spositive = self.percentage(spositive, NoOfTerms)
        negative = self.percentage(negative, NoOfTerms)
        wnegative = self.percentage(wnegative, NoOfTerms)
        snegative = self.percentage(snegative, NoOfTerms)
        neutral = self.percentage(neutral, NoOfTerms)

        # finding average reaction
        polarity = polarity / NoOfTerms

        # printing out data
        print("How people are reacting on " + searchTerm + " by analyzing "
+ str(NoOfTerms) + " tweets.")
        print()
        print("General Report: ")

        if (polarity == 0):
            print("Neutral")
        elif (polarity > 0 and polarity <= 0.3):
            print("Weakly Positive")
        elif (polarity > 0.3 and polarity <= 0.6):
            print("Positive")
        elif (polarity > 0.6 and polarity <= 1):
            print("Strongly Positive")
        elif (polarity > -0.3 and polarity <= 0):
            print("Weakly Negative")
        elif (polarity > -0.6 and polarity <= -0.3):
            print("Negative")
        elif (polarity > -1 and polarity <= -0.6):
            print("Strongly Negative")

        print()
        print("Detailed Report: ")
        print(str(positive) + "% people thought it was positive")
        print(str(wpositive) + "% people thought it was weakly positive")
        print(str(spositive) + "% people thought it was strongly positive")
        print(str(negative) + "% people thought it was negative")
        print(str(wnegative) + "% people thought it was weakly negative")
        print(str(snegative) + "% people thought it was strongly negative")
        print(str(neutral) + "% people thought it was neutral")

        self.plotPieChart(positive, wpositive, spositive, negative, wnegative,
snegative, neutral, searchTerm,
                          NoOfTerms)

    def cleanTweet(self, tweet):
        # Remove Links, Special Characters etc from tweet
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w +:\ / \ / \S
+)", " ", tweet).split())

    # function to calculate percentage
    def percentage(self, part, whole):
        temp = 100 * float(part) / float(whole)
        return format(temp, '.2f')

    def plotPieChart(self, positive, wpositive, spositive, negative, wnegative,
snegative, neutral, searchTerm,
                     noOfSearchTerms):
        labels = ['Positive [' + str(positive) + '%]', 'Weakly Positive [' +
str(wpositive) + '%]',
                  'Strongly Positive [' + str(spositive) + '%]', 'Neutral [' +
str(neutral) + '%]',
                  'Negative [' + str(negative) + '%]', 'Weakly Negative [' +
str(wnegative) + '%]',
                  'Strongly Negative [' + str(snegative) + '%]']
        sizes = [positive, wpositive, spositive, neutral, negative, wnegative,
snegative]
        colors = ['yellowgreen', 'lightgreen', 'darkgreen', 'gold', 'red',
'lightsalmon', 'darkred']
        patches, texts = plt.pie(sizes, colors=colors, startangle=90)
        plt.legend(patches, labels, loc="best")
        plt.title('How people are reacting on ' + searchTerm + ' by analyzing '
+ str(noOfSearchTerms) + ' Tweets.')
        plt.axis('equal')
        plt.tight_layout()
        plt.show()


if __name__ == "__main__":
    sa = SentimentAnalysis()
    sa.DownloadData()
```

Output :



Page - Select or creat
Sentimental - Jupyter N
localhost:8888/notebooks/Sentimental.ipynb
jupyter Sentimental Last Checkpoint: 4 hours ago (unsaved changes)
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
Enter Keyword/Tag to search about: Bitcoin
Enter how many tweets to search: 20
How people are reacting on Bitcoin by analyzing 20 tweets.

General Report:
Weakly Negative

Detailed Report:
15.00% people thought it was positive
15.00% people thought it was weakly positive
0.00% people thought it was strongly positive
10.00% people thought it was negative
10.00% people thought it was weakly negative
0.00% people thought it was strongly negative
45.00% people thought it was neutral
```

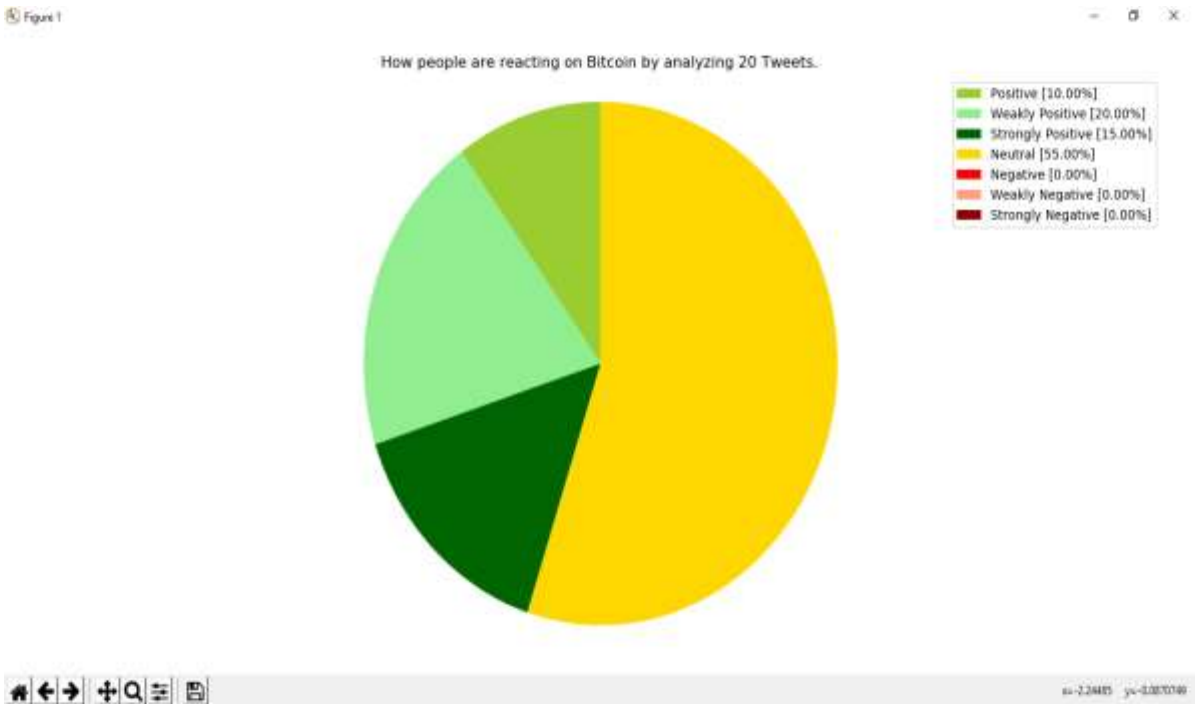


Figure 1
How people are reacting on Bitcoin by analyzing 20 Tweets.
Positive [10.00%]
Weakly Positive [20.00%]
Strongly Positive [15.00%]
Neutral [55.00%]
Negative [0.00%]
Weakly Negative [0.00%]
Strongly Negative [0.00%]
x=-2.24485 y=-0.0870749

Conclusion

One of the main aim of this project is to understand about Natural Language Processing and how it is used in Sentimental Analysis.

I have illustrated a sentiment analysis approach for extracting sentiments associated with polarity of positive or negative for specific subjects from a document, instead of classifying the whole document as positive or negative. The projected framework gathers data from the twitter and uses natural language processing techniques to extract features. Then natural language processing is applied to the data to classify the sentiments as Positive, Negative and Neutral. Polarity and partiality are also calculated by the dictionary, that consists of a semantic score of a tweet. It is observed that natural language processing is a better method for sentiment analysis as compared to traditional methods.

References

- https://medium.com/@narrativesci/natural-language-processing-vs-natural-language-generation-1b2d18dd0b67
- https://medium.com/@jrodthoughts/ambiguity-in-natural-language-processing-15f3e4706a9a
- https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17
- http://www.ijircce.com/upload/2014/sacaim/59_Paper%2027.pdf
- https://en.wikipedia.org/wiki/Natural_language_processing
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.7125&rep=rep1&type=pdf
- https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e
- https://expertsystem.com/natural-language-processing-sentiment-analysis/
- Farooqui N**.** **;** Mehra R**.** 2019 “Sentimental Analysis of Twitter Accounts using Natural Language Processing” International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8 Issue-3, February 2019.