Movie Review Sentiment Analysis with Naive Bayes in Python

**Artificial Intelligence INT-404**

Semester 4 Group Project – **K18HK**

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll no.** | **Regt no.** |
| Rakesh Yadav | A13 | 11802799 |
| Bhavik Jain | A29 | 11805618 |
| Ankit Sharma | B41 | 11802700 |

Submitted to: Dr. Rahul



# Abstract

Textual data dominates our world from the tweets you read to the timeless writings of Seneca. And while we’re consuming images (looking at you Instagram) and videos at increasing rates, you still read Google search results multiple times per daily.

One frequently recurring problem with text data is Sentiment Analysis (classification). Imagine you’re a big sugar + water beverage company. You want to know what people think of your products. You’ll search for texts with some tags, logos or names. You can then use Sentiment analysis to figure out if the opinions are positive or negative. Of course, you’ll send the negative ones to your highly underpaid support center in India to sort things out.

# Introduction:

With the advent of Web 2.0 various platforms like Facebook,

Twitter, LinkedIn, Instagram allows citizens to share their

comments, views, feelings, judgements on the myriad of

topics ranging from education to entertainment. These

platforms contain the huge amount of the data in the form of

tweets, blogs, and updates on the status, posts, etc. Sentiment

Analysis aims to determine the polarity of emotions like

happiness, sorrow, grief, hatred, anger and affection and

opinions from the text, reviews, posts which are available

online on these platforms. Opinion Mining finds the sentiment

of the text with respect to a given source of content. Sentiment

analysis is complicated because of the slang words,

misspellings, short forms, repeated characters, use of regional

language and new upcoming emoticons. So it is a significant

task to identify appropriate sentiment of each word. Sentiment

Analysis is one of the most active research areas and is also

widely studied in data mining. Sentiment analysis is applied in

almost every business and social domain because opinions are

central to most human activities & behaviours.

Sentiment analysis is very popular because of its efficiency.

Thousands of documents can be processed for sentiment

analysis. Since it is an efficient process which provides good

accuracy, therefore it has various applications:

• Purchasing Merchandise or Service: While purchasing a

merchandise or service we must take a right decision

which is not a difficult task anymore. By sentiment

analysis, people can easily evaluate reviews and opinions

of any commodity or service and can effortlessly

compare the competing brands.

• Quality Improvement in Product or Service: By Opinion

mining, the producers can collect the user’s opinion

Quality Improvement in Product or Service: By Opinion

mining, the producers can collect the user’s opinion

Quality Improvement in Product or Service: By Opinion

mining, the producers can collect the user’s opinion

* Quality Improvement in product or service: By Opinion mining, producers can collect users opinion whether favourable or not about their product or service and then they can enhance and upgrade their quality.
* Recommendation Systems: By analysing peoples opinion according to their preferences and interests, the system can predict which item to recommend.
* Decision Making: Peoples sentiments, ideas, feelings are very important factor to make a decision. while buying anything users first view on the reviews has a great impact.
* Marketing Research : The result of sentiment analysing technique can be implemented in marketing research.By this technique the attitude of consumers about some products can be analysed
* Detection of Flame :The monitoring of news groups, blogs and social media is easily possible by sentiment analysis.

There are following phases of Sentiment Analysis:

**Pre-Processing Phase**: The data is first cleaned to reduce

noise.

**Feature Extraction**: A token is given to the keywords and

this token is now put under analysis.

**Classification Phase**: Based on different algorithms these

keywords are put under certain category.

**MACHINE LEARNING METHODS**

**Naïve Bayes**

It is a technique based on Bayes’ Theorem. Naive Bayes

classifier assumes that the presence of a particular feature in a

class is unrelated to the presence of any other feature. This

model is easy to build and particularly useful for very large

datasets. Along with simplicity, Naive Bayes is known

to outperform even highly sophisticated classification

methods.

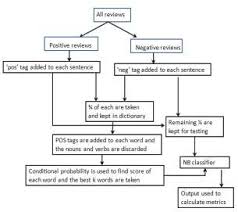
PC|X= P(X|C) P(C)/P(X) (1)

P(C|X) is posterior probability of class C

P(C) is prior probability of class C

P(X|C) is probability of predictor given the class.

P(X) is prior probability of predictor



**K- Nearest Neighbour**

K-NN is the simplest of all machine learning algorithms. The

principle behind this method is to find a predefined number of

training samples closest in distance to the new point and

predict the label from these. The number of samples can be a

user-defined constant or vary based on the local density of

points. The distance can be any metric measure. Standard

Euclidean distance is the most common choice for calculating

the distance between two points. The Nearest Neighbours

have been successful in a large number of classification and

regression problems, including handwritten digits or satellite

image processing and so on.

**Random Forest**

Random Forests are the learning method for classification and

regression. It construct a number of decision trees at training

time. To classify new case it sends the new case to each of the

trees. Each tree perform classification and output a class. The

output class is chosen based on majority voting that is the

maximum number of similar class generated by various trees

is considered as the output of the Random Forest.

Random Forests are easy to learn and use for both

professionals and laypeople with little research and

programming required. It can easily be used by persons that

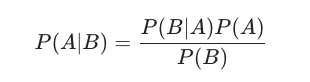
don’t have a strong statistical background.

**TECHNIQUE USED:**

**Naive Bayes** models are probabilistic classifiers that use the Bayes Theorem and make a strong assumption that the features of the data are independent. For our case, this means that each word is independent of others.

Intuitively, this might sound like a dumb idea. You know that (even from reading) the prev word(s) influence the current and next ones. However, the assumption simplifies the math and works really well in practice

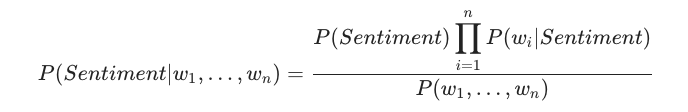
The Bayes theorem is defined as:



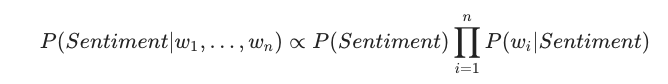
where **A** and **B** are some events and **P(.)** is a probability.

This equation gives us the conditional probability of event **A** occurring given **B** has happened. In order to find this, we need to calculate the probability of **B** happening given **A** has happened and multiply that by the probability of **A** (known as Prior) happening. All of this is divided by the probability of **B** happening on its own.

The naive assumption allows us to reformulate the Bayes theorem for our example as:



We don’t really care about probabilities. We only want to know whether a given text has a positive or negative sentiment. We can skip the denominator entirely since it just scales the numerator:



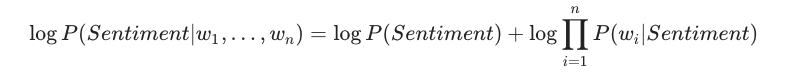
# **Implementing Multinomial Naive Bayes**

As you might’ve guessed by now, we’re classifying text into one of two groups/categories — positive and negative sentiment.

Multinomial Naive Bayes allows us to represent the features of the model as frequencies of their occurrences (how often some word is present in our review). In other words, it tells us that the probability distributions we’re using are multinomial.

## ****Note on numerical stability****

Our model relies on multiplying many probabilities. Those might become increasingly small and our computer might round them down to zero. To combat this, we’re going to use log probability by taking log of each side in our equation:



Note that we can still use the highest score of our model to predict the sentiment since log is monotonic.

# Proposed Methodology

# **Dealing with Text**

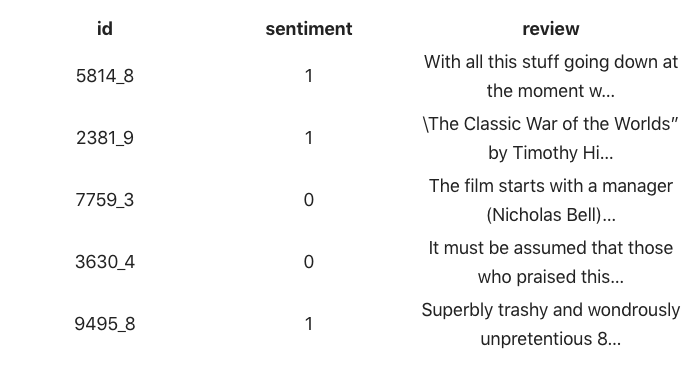
Computers don’t understand text data, though they do well with numbers. Natural Language Processing(NLP) offers a set of approaches to solve text-related problems and represent text as numbers. While NLP is a vast field, we’ll use some simple pre-processing techniques and Bag Of Words model.

## ****The Data****

We have 25,000 movie reviews from IMDB labelled as positive or negative. You might know that IMDB ratings are in the 0–10 range. An additional pre-processing step, done by the dataset authors, converts the rating to binary sentiment (<5 — negative ). Of course, a single movie can have multiple reviews, but no more than 30.

## ****Exploration****

Let’s get a feel for our data. Here are the first 5 rows of the training data:

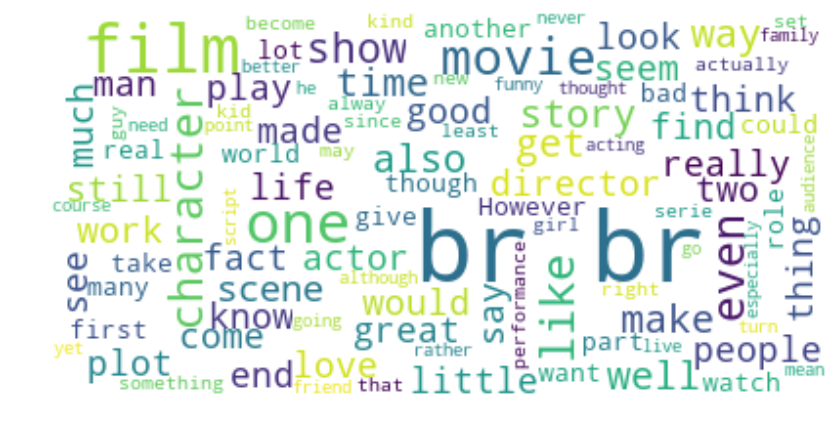


We’re going to focus on the sentiment and review columns. The id column is combining the movie id with a unique number of a review. This might be a piece of important information in real-world scenarios, but we’re going to keep it simple.



Both positive and negative sentiments have an equal presence. No need for additional gimmicks to fix that!

Here are the most common words in the training dataset reviews:



## ****Cleaning****

Real-world text data is really messy. It can contain excessive punctuation, HTML tags (including that **br**), multiple spaces, etc. We’ll try to remove/normalize most of it.

Most of the cleaning we’ll do using [Regular Expressions](https://en.wikipedia.org/wiki/Regular\_expression), but we’ll use two libraries to handle HTML tags (surprisingly hard to remove) and removing common (stop) words:

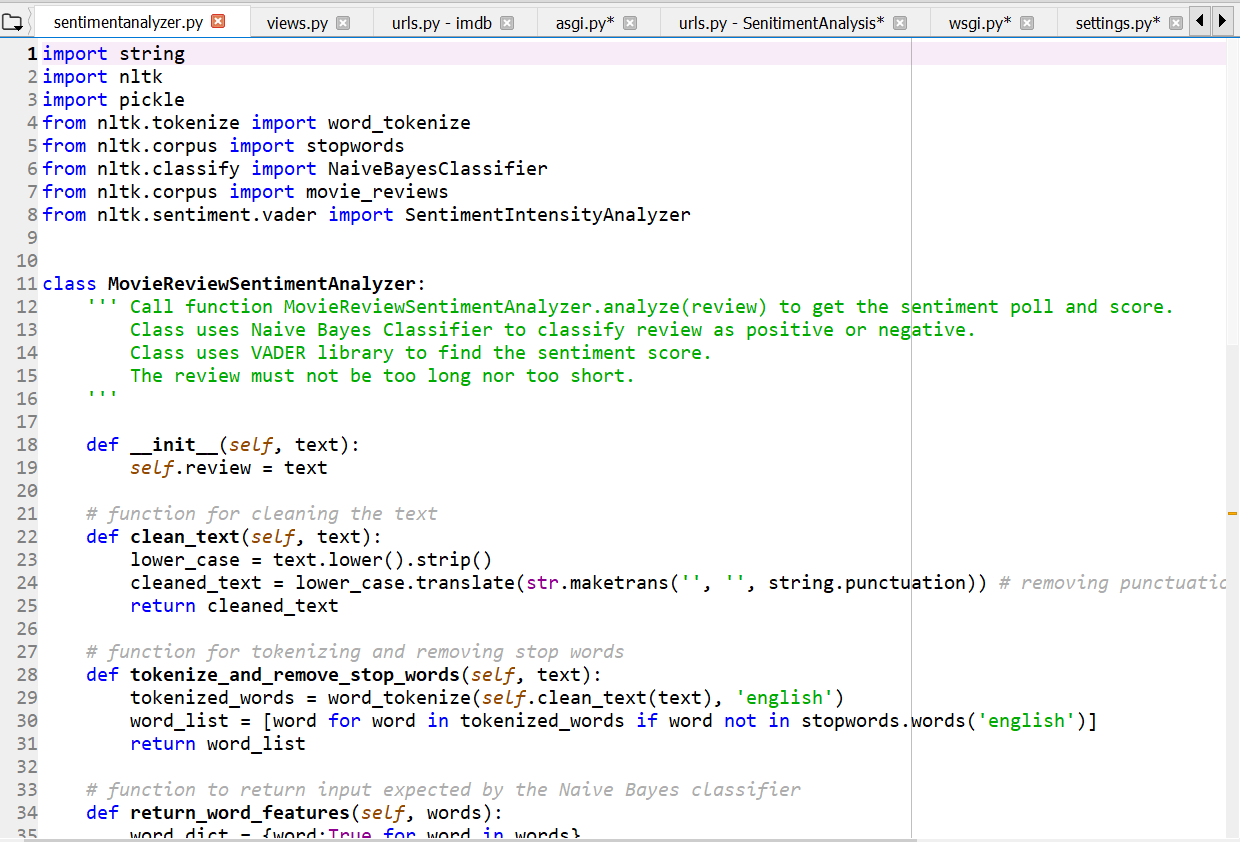
## ****Tokenization****

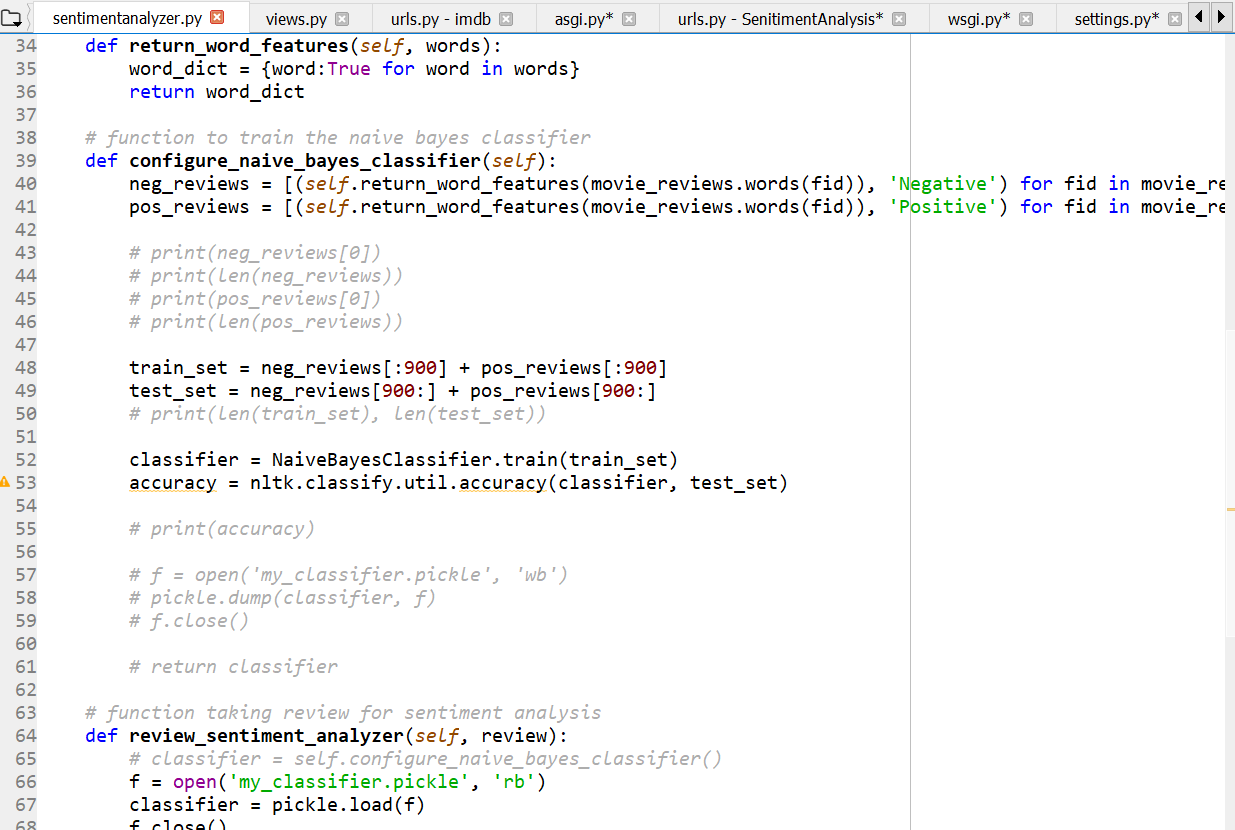
Now that our reviews are “clean”, we can further prepare them for our Bag of Words model. Let’s them to lowercase letters and split them into individual words. This process is known as tokenization.

The last step of our pre-processing is to remove stop words using those defined in the NLTK library. Stop words are usually frequently occurring words that might not significantly affect the meaning of the text. Some examples in English are: “is”, “the”, “and”.

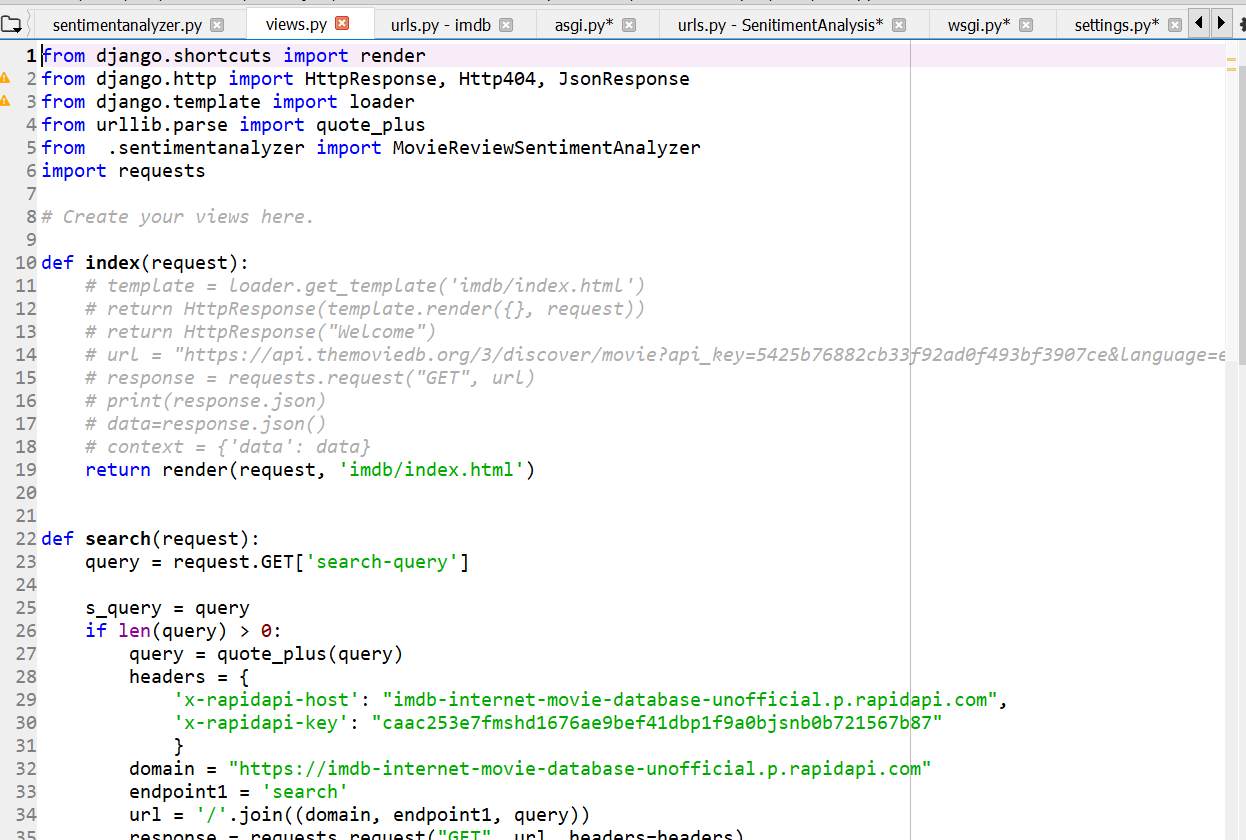
An additional benefit of removing stop words is speeding up our models since we’re removing the amount of train/test data. However, in real-world scenarios, you should think about whether removing stop words can be justified.

## Code : the steps involving of cleaning tokenization etc…

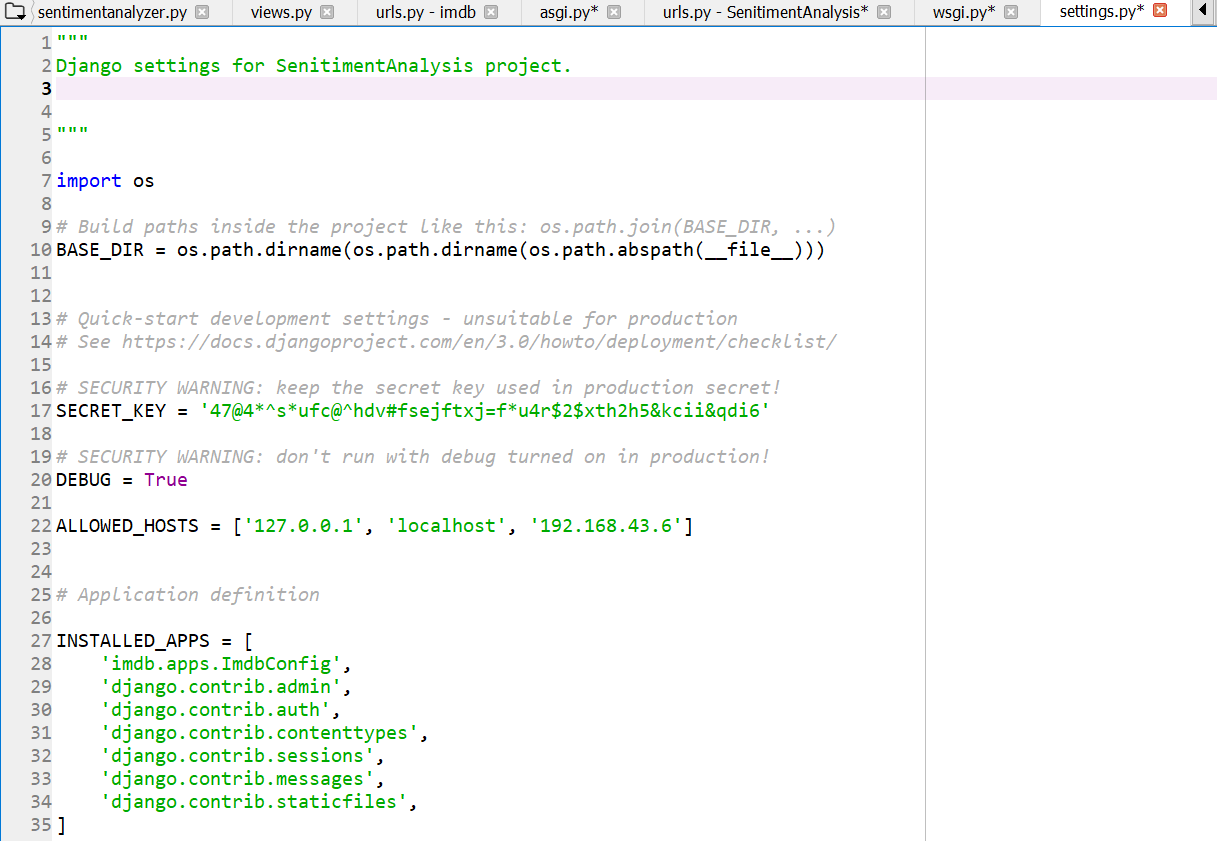




The filtering of reviews and the analysis template of the code :



Django Settings for the API’s used and the domains involved:



Conclusion

Sentiment analyzer systems open new opportunities of retrieving personalized information on the Internet. It also helps to alleviate the problem of information overload which is a very common phenomenon with [information retrieval systems](https://www.sciencedirect.com/topics/computer-science/information-retrieval-systems) and enables users to have access to products and services which are not readily available to users on the system. Various learning algorithms used in generating analysis models and [evaluation metrics](https://www.sciencedirect.com/topics/computer-science/evaluation-metric) used in measuring the quality and performance of [analysis algorithms](https://www.sciencedirect.com/topics/computer-science/recommendation-algorithm) were discussed. This knowledge will empower researchers and serve as a road map to improve the state of the art sentiment analysis techniques.

# References

=> Wikipedia.com

=> Towardsdatascience.com

=> Sciencedirect.com

=> Youtube.com

=> Github.com