# Sentiment Analysis

***Report***

Panagiotis Kyriakidis

**Programming for Data Science**

MSc in Data Science

UNIVERSITY CENTER OF INTERNATIONAL PROGRAMMES OF STUDIES
SCHOOL OF SCIENCE AND TECHNOLOGY

For this assignment, we will be analyzing an IMDB Review Dataset, found on Kaggle.com [1]

The steps taken will be briefly described below:

## Importing Libraries

First, we import the necessary libraries required for our analysis. Libraries like sklearn, pandas, nltk, matplot, seaborn and wordcloud are among the many used for this assignment.

## The IMDB dataset

This simple dataset consists of two attributes:

- **Review attribute**
  Contains a user submitted review of a random movie
- **Sentiment**
  Characterizes the review as 'positive' or 'negative'.

## Data Cleaning

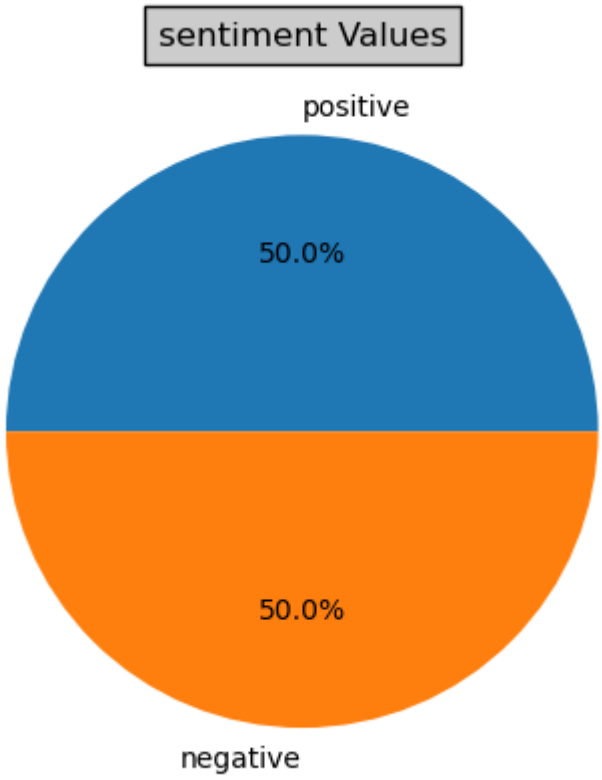
This dataset has low dimensionality and contains 50,000 entries, making it an ideal dataset for testing Machine Learning techniques, such as Sentiment Analysis.

There are no empty rows to remove. For better processing of the data, we have assigned numbered labels to each review value: 1 is assigned to "positive" and 0 is assigned to "negative".

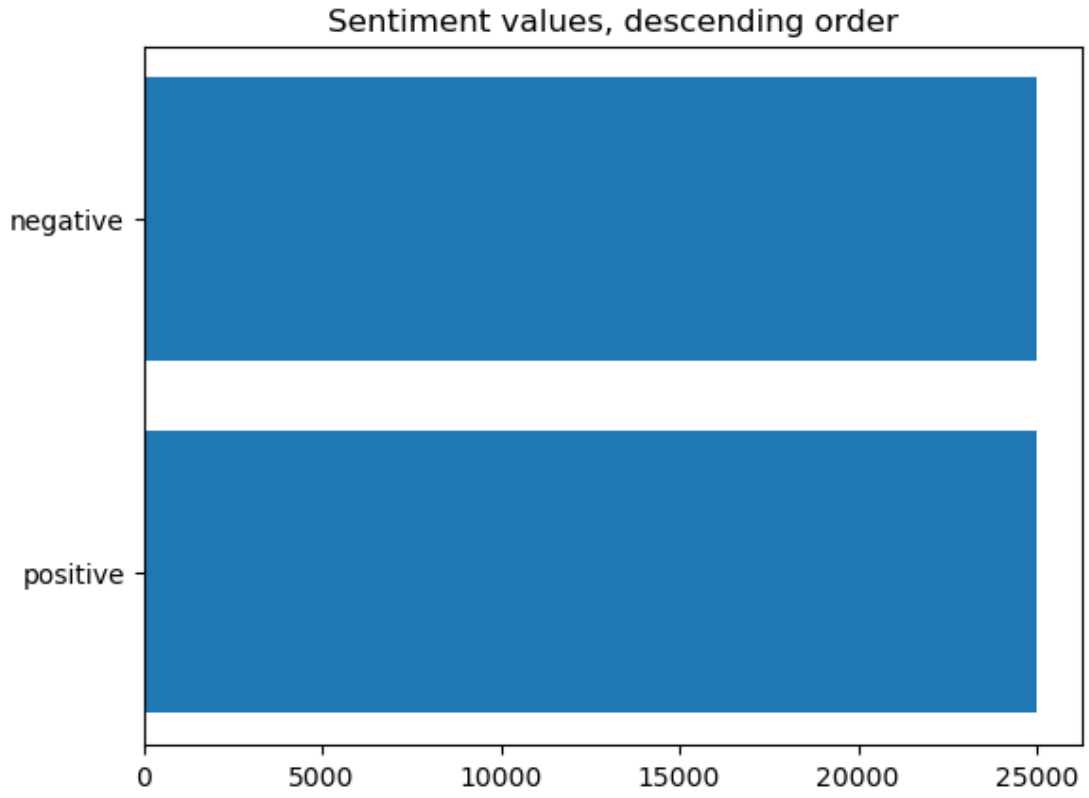
## Data Visualization

Visualizing data aids in understanding the distribution of our problem and the underlying structure of our dataset. It will enhance the process of analysis.

- Distribution of Sentiment values in a Pie Chart ( Positive – Negative)

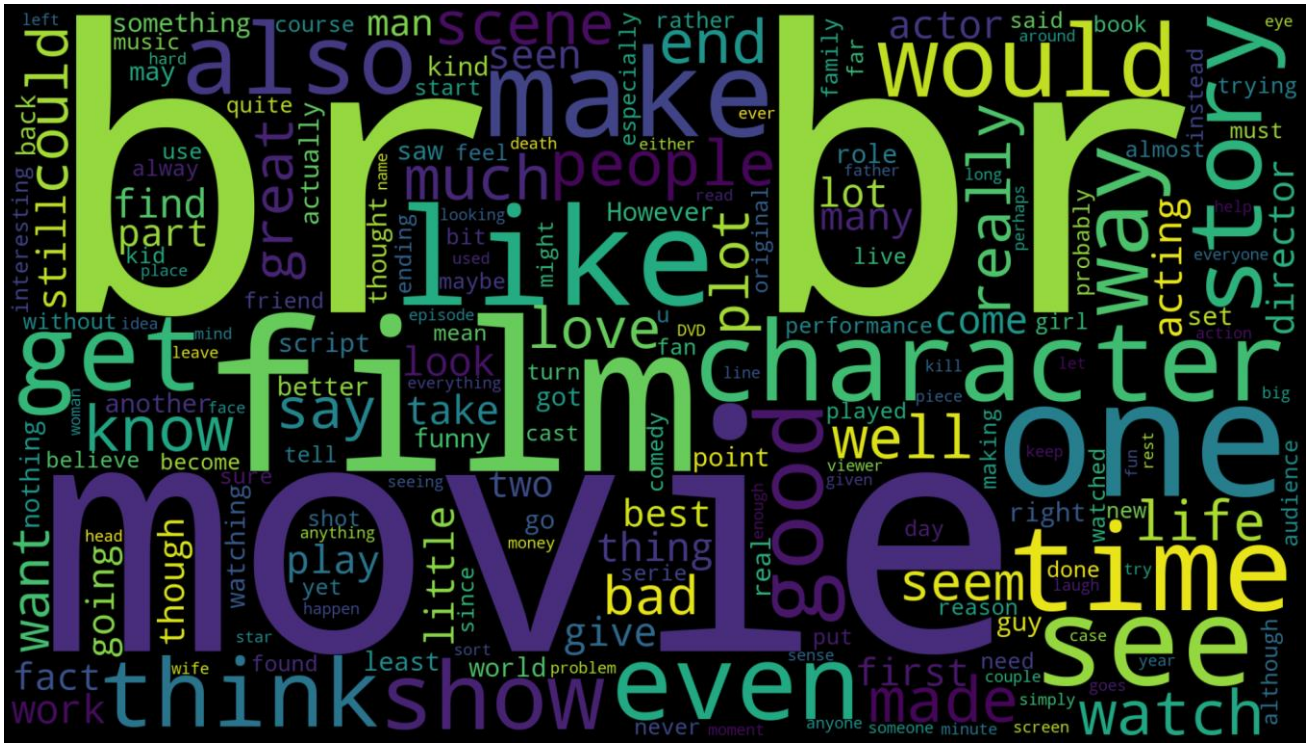


- Distibution in a Bar Chart




It is clear the data is evenly distributed 50-50. This is a perfectly balanced dataset.

- Distribution of most used words present in the reviews with WordCloud



Through WordCloud we showcase the most frequently used words in the reviews in total.

It is clear that the popular words 'br' offer no meaning to the sentiment. In reality, <(br)> is used for html formatting, which was passed along with the text while the data was retrieved.

## Data Preprocessing

A very crucial part of sentiment analysis. The text that will be used for our sentiment models must be preprocessed and converted into a proper form, devoid of any noise and transformed into a form readable to the machine.

The steps taken are as follows:

- **Unicode to ascii**
  Converting any Unicode characters to ascii for machine readability.

- **Contractions handling**
  Contractions are words or combinations of words that are shortened by dropping letters and replacing them by an apostrophe, such as "you are" -> "you're" .

- **Removing HTML tags**
  Removing noise such as <br> tags, as it was indicated through the WordCloud visualization.

- **Converting text to lowercase**

- **Removing URLs and Special Characters**

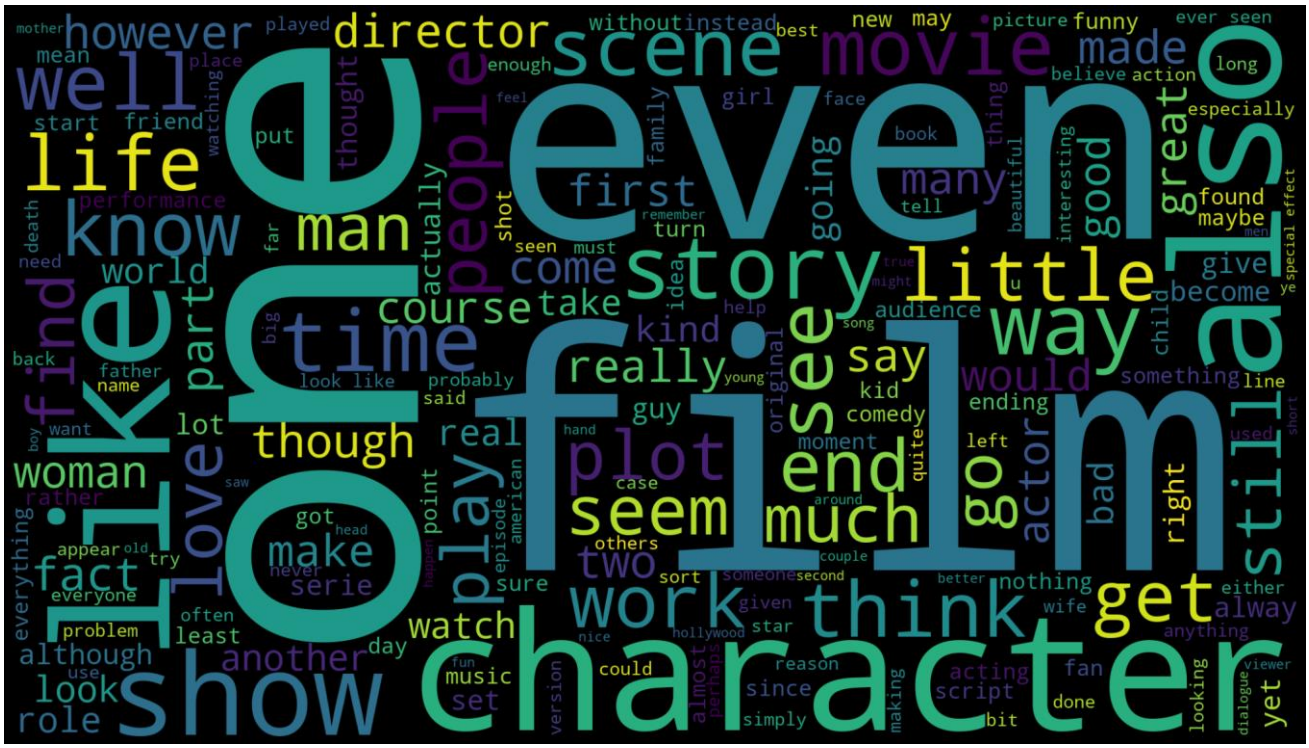
- **Deleting Numbers**

- **Punctuation Handling**

- **Removing Stopwords**
  Stopwords are frequent words that contain little to no meaning for a sentence. They can be removed.

- **Lemmatization-stemming**
  This is the process of converting any word into their root form, simplifying text analysis.

After the preprocessing, the text is ready to be analyzed by algorithms. If we distribute reviews with WordCloud again, we can see the actual popularity of words in a clean dataset.




## Sentiment Analysis with TextBlob

TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, and more.

By extracting the polarity score for each separate review, we assign a label to each one:

- **Positive** if polarity >0
- **Negative** if polarity < 0
- **Neutral** if polarity = 0

These are the counts of the values:

```
positive    36806
negative    13146
neutral        48
Name: TextBlob Sentiment, dtype: int64
```

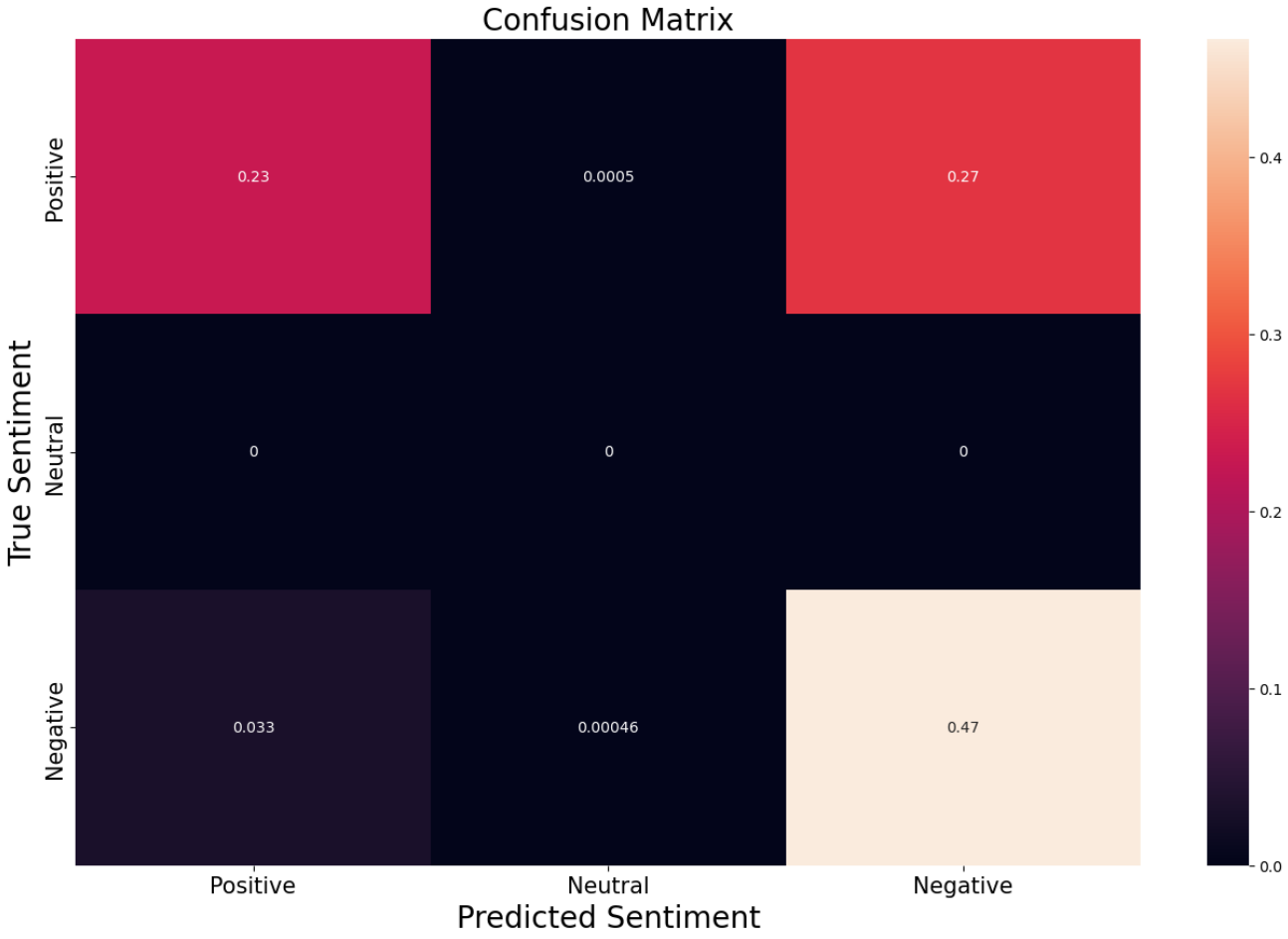
## Evaluation metrics for TextBlob

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.88 | 0.46 | 0.60 | 25000 |
| neutral | 0.00 | 0.00 | 0.00 | 0 |
| positive | 0.63 | 0.93 | 0.76 | 25000 |
| accuracy | | | 0.70 | 50000 |
| macro avg | 0.50 | 0.46 | 0.45 | 50000 |
| weighted avg | 0.76 | 0.70 | 0.68 | 50000 |

```
Accuracy on sentiments: 0.69726
Precision on sentiments: 0.46484
Recall on sentiments: 0.5034446120908618
F1-Score sentiments: 0.45309281669590734
```

Note that there are no “neutral” values on the actual review sentiments.

## Confusion Matrix



# Vader Sentiment Analysis

Polarity scores give a dictionary containing pos, neg and neu values, as well as compound scores. The compound scores will be used similarly to the previous analysis.

By extracting the compound score for each separate review, we assign a label to each one:

- **Positive** if score >= 0.05
- **Negative** if score < 0.05
- **Neutral** if score = 0

These are the counts of the values:

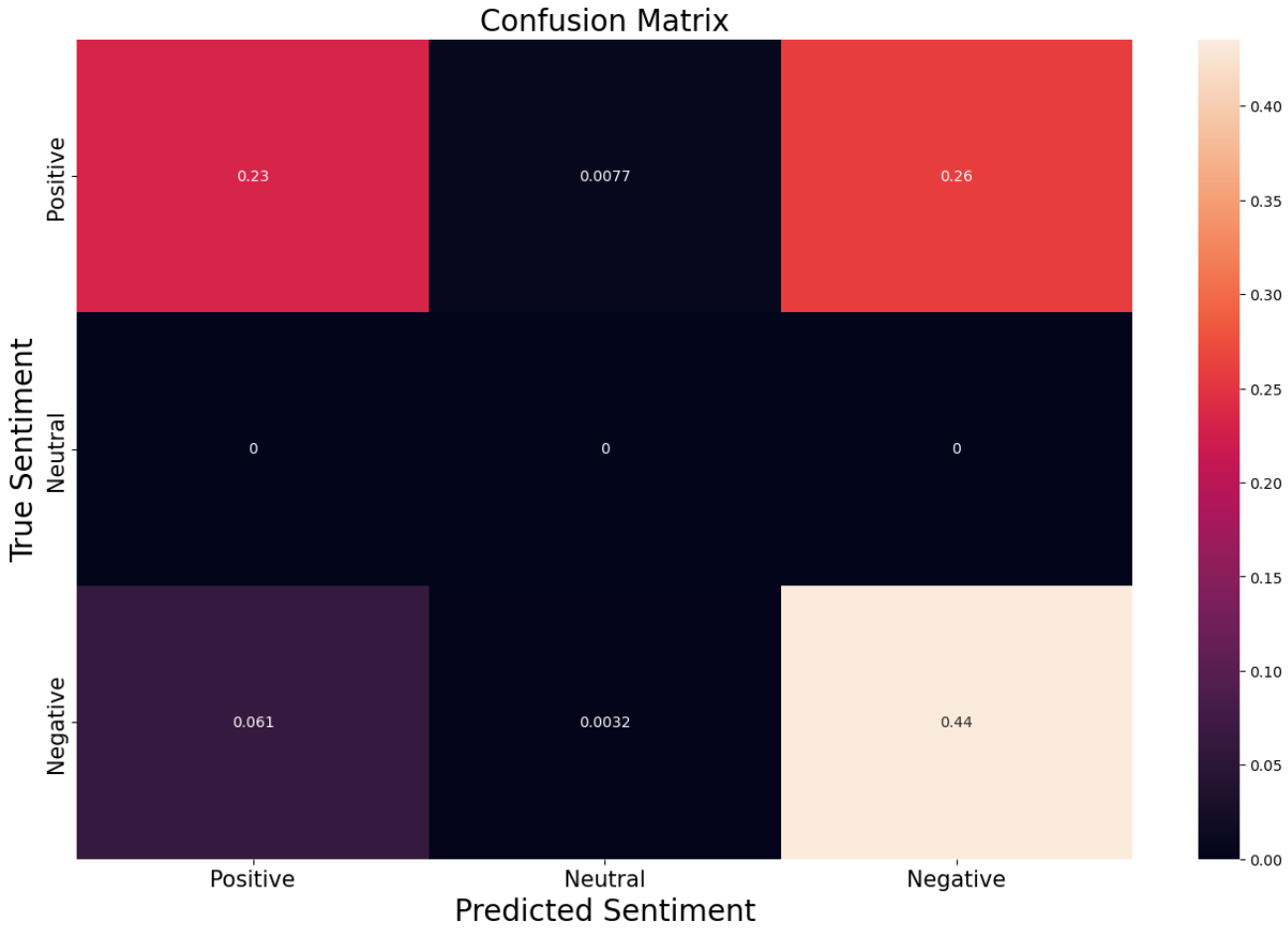
```
positive    34739
negative    14717
neutral       544
Name: Vader Sentiment, dtype: int64
```

## Evaluation metrics for Vader

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negative | 0.79 | 0.47 | 0.59 | 25000 |
| neutral | 0.00 | 0.00 | 0.00 | 0 |
| positive | 0.63 | 0.87 | 0.73 | 25000 |
| accuracy | | | 0.67 | 50000 |
| macro avg | 0.47 | 0.45 | 0.44 | 50000 |
| weighted avg | 0.71 | 0.67 | 0.66 | 50000 |

```
Accuracy on sentiments: 0.66836
Precision on sentiments: 0.4455733333333334
Recall on sentiments: 0.47271294782699114
F1-Score sentiments: 0.43846231606856173
```

## Confusion Matrix




# Conclusion

The IMDB review dataset is a very good dataset for testing sentiment analysis, as it is easily cleaned, preprocessed and analyzed. After performing analysis with the libraries TextBlob and Vader, the former scored a slightly higher score for all accuracy metrics performed on the resulting sentiment predictions, compared to the actual sentiments provided by the dataset.