Applied Natural Language Processing

Introduction

The popularity of every film can be measured based on popular sentiment. Involved parties may articulate their views in various ways, whether indefinite, numerical or text type. This answer is often helpful or not general terms for the directors or producers to adapt their film. The movie reviews offer a glimpse of production, music, casting and action. Many topics cover machine learning, and one of them is opinion mining. The primary objective of the sentiment analysis is to obtain the views of the respondent on the subject. For sentiment analysis, related topics like data mining and natural language processing have also been needed (NLP). Conclude the respondent's view on the particular or generically defined subject in a sentimental review. In emotion analysis, the principal challenge is to assess the polarity of the reviewer while his observations are positive or negative. This main project activity is to carry out a sentiment analysis on the IMDB website with a film review dataset. Analyze people's reactions to films and understand the criticism over movies when they are critical or screening the movie—using terms that work in response to assess the total degree of polarity of the respondent to prevent the input from the citizens. The use of data is linked to public reaction in this project and is obtained by IMDB.

Problem

The fundamental purpose of this paper is to determine the basic feeling of a film review based on the text. In this experiment, we try to find out whether or not a person enjoyed the film. To assess its relative success by analysis and audience feedback is especially useful. It is also helpful. Based on preliminary ratings, the results of this evaluation were used to recommend films to audiences. Finding a segment of audiences for related movies would also be an application of this project. It's beautiful (likes or disgusting). We will study different strategies for extracting text mining functions as part of this research. For example, to find keywords, to detect lexical affinities and to foresee our problem's relevance. - This is our problem. We explore different classifications and remove functional methods and how well they perform with different types of feature representation. Finally, we discover that the mixture of classification definitions and approaches to the current forecasting task sometimes is more precise.

Literature review

Initial study of the dataset by the Stanford University team, unattended learning was used to fuse terms with close semantics and word vectors. To define the polarity of feedback, they have carried out different classification models for these vectors. This approach is constructive if the data contain rich emotions and are subjective in the summarized affinities and significance of the terms. To find polarity in the film evaluations and evaluations of the product, Bo Pang and Peter Turnkey have carried out a lot of testing. They have also sought to build a hierarchy of many assessments and to forecast film/product critic's assessments. In the following papers, the use of the RFC to differentiate reviews and the use of different mining approaches are addressed. Important to remember as neutral texts were not unreasonably classifiable alongside the binary classification boundaries. These Days, several free or qualified sentiment analyses

and applications are available. Sensational comments have been used extensively to analyze the rise of microblogging and to gain feedback from general public opinions.

Exploratory Data Analysis

EDA is used by computer scientists to analyze, examine and outline data sets, often using data visualization methods. Scientists use data visualization. It will also help you to determine if the methodology you evaluate is appropriate for data processing. One of the first steps in the review text is to calculate the average input size to get an insight into the review material. The graphs below illustrate clearly how each study differs from the word count. From this data, we have concluded that people generally tend to make comprehensive film analyses, making it a good topic for research. People also tend to write reviews when you're feeling intense about a film; you love or disdain it.

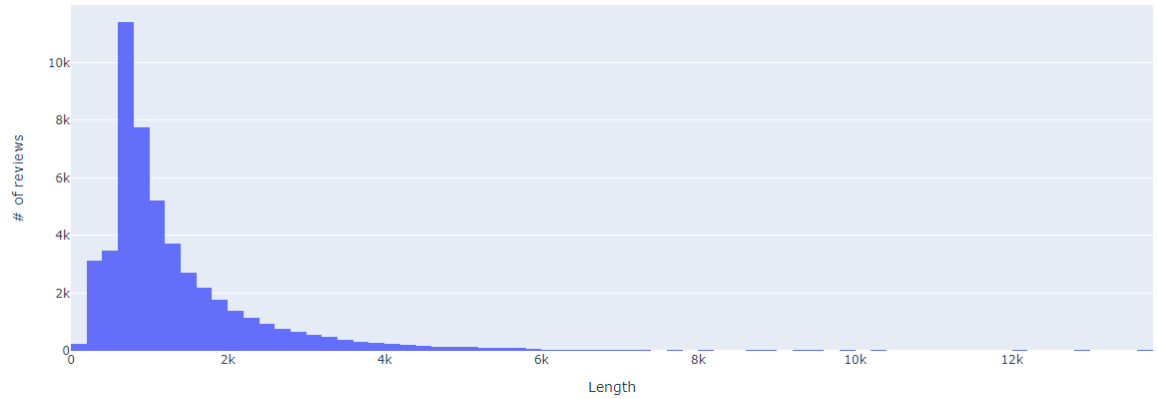


Figure 1 Overall length of Reviews

One of the first steps in the review text is to calculate the average input size to get an insight into the review material. The graphs below illustrate clearly how each study differs from the word count.

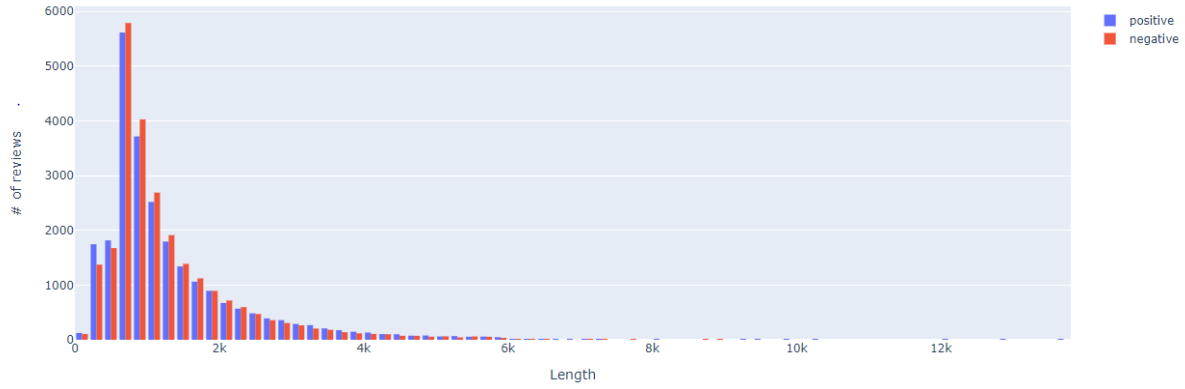


Figure 2 Per review length

From this data, we have concluded that people generally tend to make comprehensive film analyses, making it a good topic for study. People also tend to write reviews when you're feeling intense about a film; you love or disdain it.

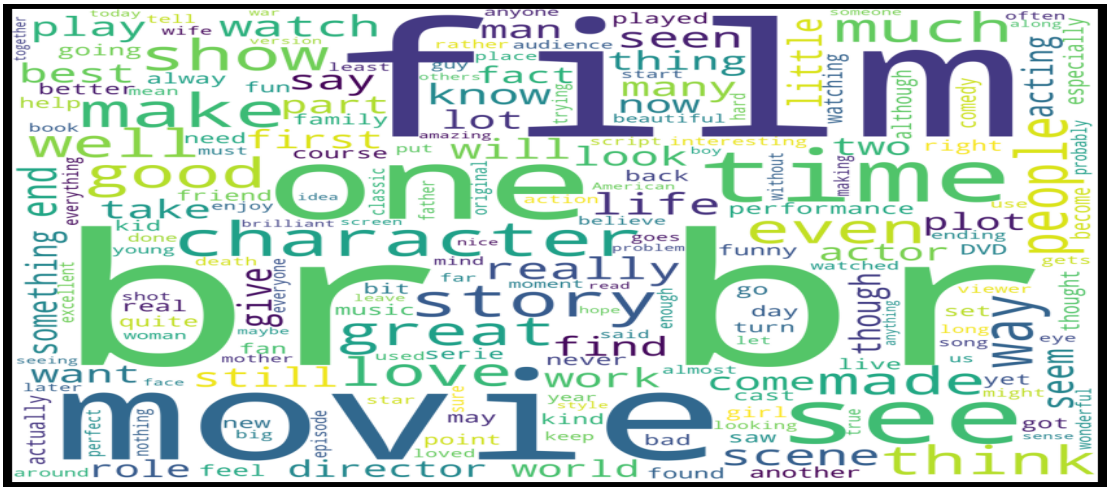


Figure 3 Positive Reviews Word Cloud

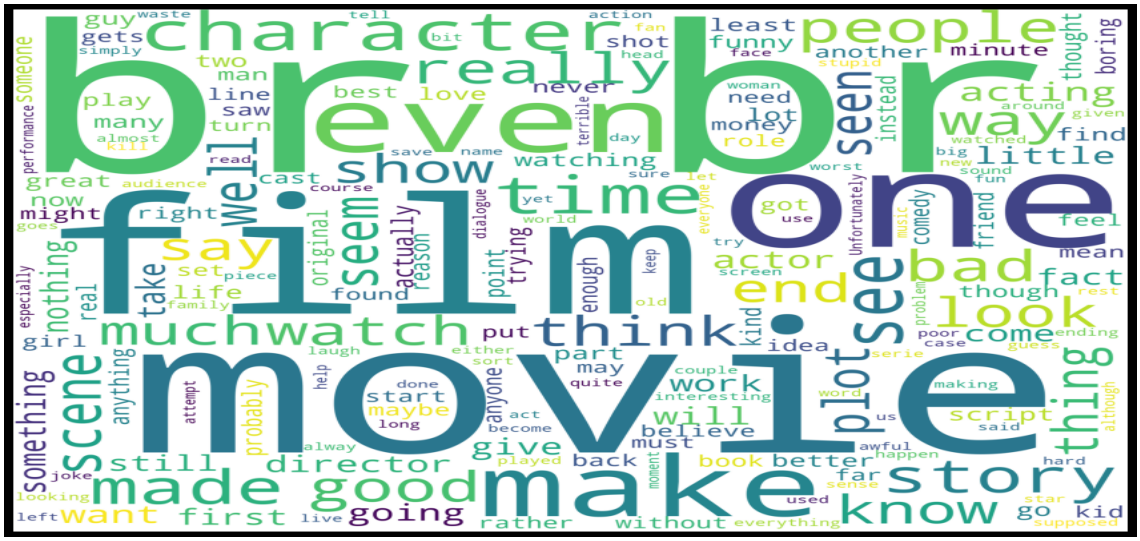


Figure 4 Negative Reviews Word Cloud

- **Data Set Balancing**

negative	25000
positive	25000

data preprocessing

The film reviews are then processed in advance. To do this, we are using the Sklearn Count Vectorizer API, which transforms text materials into a token counts matrix. In practice, the sparkling bag of words is produced with the admonishment that all interesting vocabulary elements, such as word order, are ignored. In some instances, it is essential to order the words. If you have a "not," the "not" applies to the object in front of you. However, we want to guess whether something, in this case, is excellent or negative. If "absurd" or "cryptic" always occurs, this is probably not very important. Therefore, the plan is to make it a text matrix in which we merely mention, instead of the order in which they are seen, the words in each document (i.e. any review).

- **Avoidable Items**

HTML tags, Stop words, numbers, punctuation,

- **Uppercase to Lowercase**
- **Stop words 'n punctuation removing.**

Text Normalization

Until text data are being used in NLP training models, it is preprocessed in a suitable form. Normalization of text is also a critical step of text preprocessing. The standardization of the text makes modelling more manageable and enhances model accuracy. In standardizing the language, no set sequence of exercises was used. Tasks are based on the application parameters. The standardization of text started with a text-to-speech system and subsequently was essential to the processing of social media text. Text normalization limits the distinction in terms of form to a related condition in which the variants are the same. america or America, for instance, in the United States.

Tokenization

Before we translate our text into a term-document matrix, we must tokenize it first. The text is translated to a word list with NLP tokenization. For example.

```
transform_text('there is a tree near <br/> the river 123! see')
```

```
'tree near river see'
```

Lemmatization and Stemming

Data would be organized for grammatical purposes under different names. The families of derived words, including socialism, democracy and democratization, have similar meanings. It appears in some instances that records have been returned to look for one of these words in a different world in the series. The aim is to reduce inflection forms and, at times, derivative conditions, both stemming and lemmatization, into one basic setup.

The two words differ in their taste. Stemming is usually a simple heuristic process that cuts out the ends of the terms in the expectation that this goal will be accomplished much of the time accurately and eliminates derivative appeals. A vocabulary and morphological word analysis to properly achieve this usually includes removing inflection endings and restoring a simple or dictionary word type, referred to as the lemma. Lemmatization requires If this token was seen, the only s could be returned as the token would attempt to see whether this token was used as a verb or as a substantive. It would not have been practicable to return the token.

```
lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("rocks", pos="v"))
print(lemmatizer.lemmatize("gone", pos="v"))
```

```
rock
go
```

See the word cloud again after the pre-emptive text and the normalization of the text.

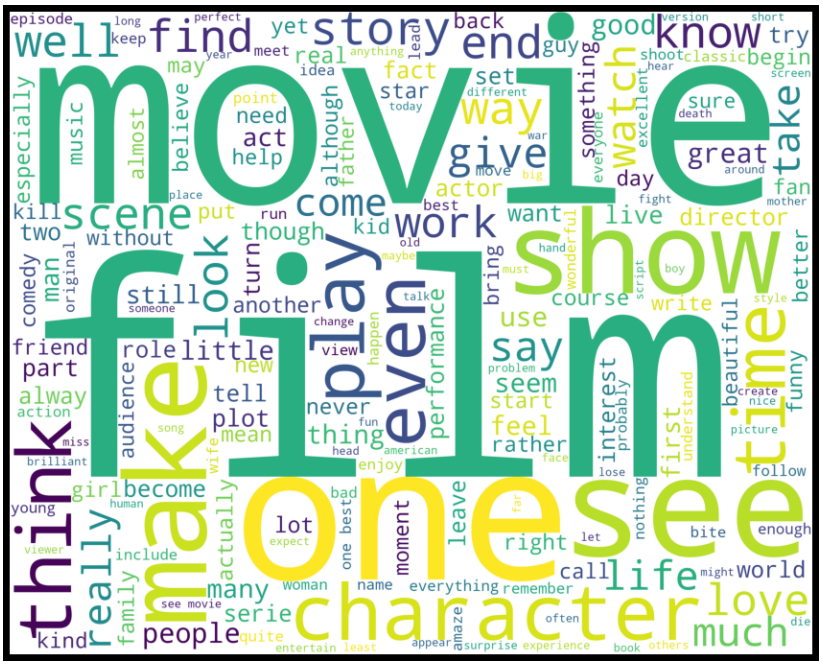


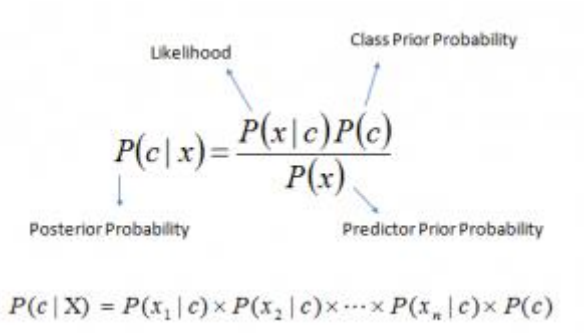
Figure 5 Positive reviews word cloud after performing text preprocessor.

Figure 6 Negative reviews word cloud after performing text preprocessor

Naive Bayes Algorithm

It is a classification method based on the Bayes theorem, which takes on the independence of predictors. In other words, there is no relation between the life of any function within a course and that of other features in the group Naive Bayes. The fruit may be called an apple, for example, because it is red, oval, and 3 inches in diameter. And if these features jointly determine or rely on the existence of other features, then all these features independently add to the chance that the fruit is an apple.

The Naive Bayes paradigm is easy to build and particularly valuable for big data sets. Naive Bayes also proposes a complex classification and simplicity approaches instead. The theorem for Bayes provides a means of determining the prior probability of P(c|x) of P(c), P(x) and P(x|c). See the calculation below



Let's look at a clue. Below I have a collection of training data and the relevant 'positive or negative' target variable (suggesting positive or negative possibilities). We now decide whether or not the sentence is positive or negative based on opinions.

Train and Test dataset

We will now divide the dataset is divided into two segments: train and evaluates

```
pos_train = df[df['sentiment']=='positive'][['review_lm', 'sentiment']].head(17500)
neg_train = df[df['sentiment']=='negative'][['review_lm', 'sentiment']].head(17500)


# Test dataset (last 7.500 rows)
pos_test = df[df['sentiment']=='positive'][['review_lm', 'sentiment']].tail(7500)
neg_test = df[df['sentiment']=='negative'][['review_lm', 'sentiment']].tail(7500)

# put all toghether again...
train_df = pd.concat([pos_train, neg_train]).sample(frac = 1).reset_index(drop=True)
test_df = pd.concat([pos_test, neg_test]).sample(frac = 1).reset_index(drop=True)
```

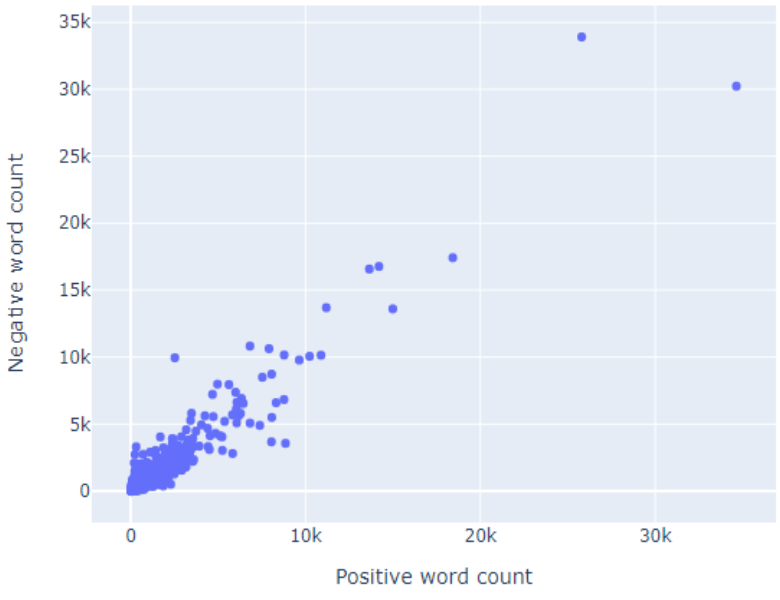
Model Fitting

The current moment rings out the "educated" way of the model.

```
word_count_df = pd.DataFrame(word_counts).fillna(0).sort_values(by='positive', ascending=False).reset_index()
word_count_df
```

	index	positive	negative
0	film	34640.0	30235.0
1	movie	25792.0	33910.0
2	one	18422.0	17424.0
3	see	14993.0	13607.0
4	make	14203.0	16773.0
...	...	...	...

Now visualize the word distribution.



Check now whether phrases are positive or negative.

```
result = predict(test_df['review_lm'], vocab, word_counts, num_messages, log_class_priors)
result[0:10]
```

```
['negative',
 'negative',
 'positive',
 'positive',
 'positive',
 'positive',
 'positive',
 'negative',
 'negative',
 'negative']
```

Confusion matrix

Predicted	negative	positive
Real		
negative	82.31	17.69
positive	11.17	88.83

Let us now finally test the consistency of the model. **Which is 0.8557**

The accuracy is the self-design model, so it's considerable.

Visualization of the Confusion Matrix

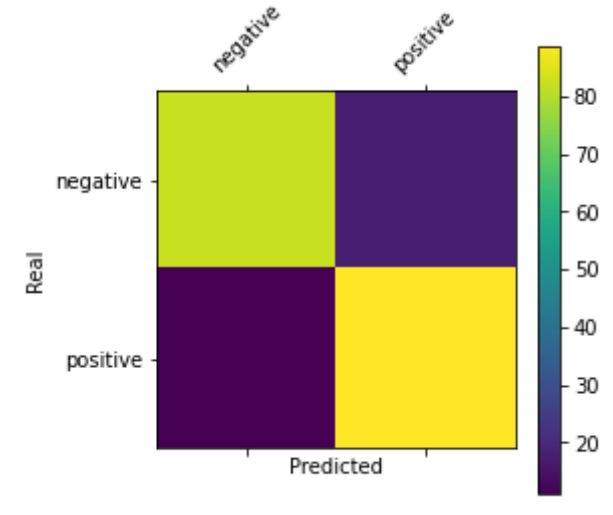


Figure 7 Picturing the Confusion Matrix