Text Mining Hand-in Assignment 2

Elin Benja Dijkstra (s2696096)
December 1st 2019

1 Introduction

Sentiment analysis is the practive of recognizing sentiment from text. Sentiment analysis can be used to, among others, get an overview of the opinion on a certain topic of larger groups of people. This can be adopted in for example social media, reviews or forums. For this assignment, the goal is to correctly classify reviews from the IMBd database as positive or negative by recognizing the sentiment used in the review. This will be done by implementing the Doc2Vec package.

2 Data Exploration

The dataset consists of 50000 movie reviews posted on the Internet Movie Database (IMDb). It is split into two equally large sets for training testing. The sets are disjoint to avoid having the model achieve high scores due to memorization instead of generalization. All movie reviews are highly polarizing, meaning they are either very negative with a rating of $\leq 4/10$ or highly positive with a rating $\geq 7/10$. The dataset is well balanced so that there are equally as many negative as there are positive reviews. Since rating for the same movie tend to be correlation, no more than 30 reviews per movie are to be included.

Looking at the pre-processed data, we see that the number of unique words in the learned vocabulary is 262179. The most frequent words are, not surprisingly, stopwords such as 'the', 'and', 'a'. The most unique words include 'sadlers', 'sord', 'galiena' and 'horserace'. The least frequent words could be typing mistake, for example where 'sord' perhaps should have been 'sword' or 'sort'.

To get an idea of the length of the documents, we include a histogram of the number of words in the reviews in Figure 1. We see that most reviews are short with the most frequent length being around 125 words. It has an interesting little spike at the beginning at around 50 which might point towards a multi-modal distribution. The distribution has a long tail, cut of at 1200 for visual purposes.

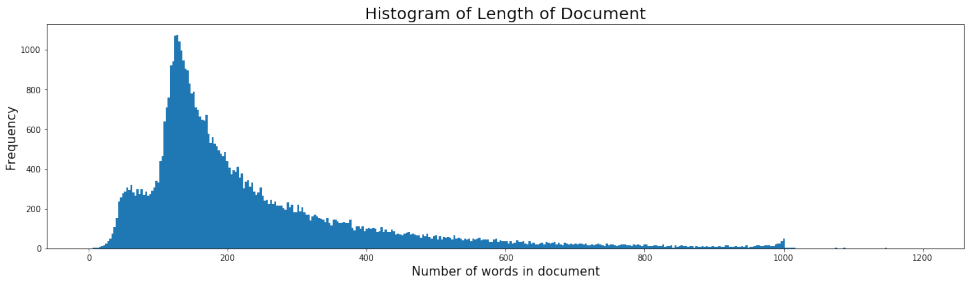

Figure 1: Histogram of number of words per document

3 Methods

The process of classifying our documents consists of three phases: preprocessing, training and testing.

3.1 Pre-processing

Before we start training the model, the first thing we ought to do is make the data suitable for our model. Doc2Vec asks for an input where every line represents a document, without any punctuation, capital letters of special characters. In the IMDb dataset, this is still included. We also see some *HTML* formatting where *</br>* is used to indicate a new line. We eliminate all these characters and combine all separate review files into one. When reading and writing them to new *.txt* files, as done in the assignment, we set the encoding to *UTF-8* to avoid encoding errors.

When removing punctuation and *HTML* coding, it results in some words being merged. Therefore, in an attempt to increase performance, replacing these with a white space is also tested. Additionally, we train the model on the text where numbers are removed.

3.2 Doc2Vec

The Doc2Vec package in Python is used. Doc2Vec is build on Word2Vec which provides a numeric representation for each word. Therefore, it will be able to capture the relationship between words. Doc2Vec is an extension to Word2Vec which is able to create a numeric representation of a document of any length. Here a document is represented as a vector with a unique label. If these vectors represent similar concepts, their vector should also be close in distance. Doc2Vec has the nice functionality that is takes the word-order into account whereas Count Vectorizer and TF-IDF do not.

3.3 Features

The settings we explore include 'window' and 'size' in the Doc2Vec model. Size is the dimension of the feature vector in the output. We try increasing and decreasing it. We do the same for the window size, which determines that maximum distance between the current and predicted word.

3.4 Classifiers

Logistic Regression is a regression model that estimates the probability of membership of a class. It becomes a classifier by combining it with a decision rule assigning it to a class.

The Linear Support Vector Classifier determines the best decision boundary between categories. It does do by maximizing the margin between points within certain categories and the maximum margin line or hyperplane.

The SGDClassifier uses Linear Support Vector Machines with stochastic gradient descent learning. The classifier is updated with a step size of the learning rate in the direction of the negative gradient of the loss function.

4 Results

Feature Set	Classifier	Labels	Weighted Precision	Weighted Recall	F1-score	Accuracy
Dim = 100, Window = 10	Logistic Regression	Pos	0.86	0.86	0.86	0.86284
		Neg	0.86	0.87	0.86	
Dim = 100, Window = 10	SGDClassifier	Pos	0.83	0.88	0.86	0.85064
		Neg	0.88	0.82	0.85	
Dim = 100, Windows = 10	LinearSVC	Pos	0.86	0.86	0.86	0.86328
		Neg	0.86	0.87	0.86	
Dim = 250, Windows = 10	LinearSVC	Pos	0.87	0.86	0.86	0.8658
		Neg	0.86	0.87	0.87	
Dim = 100, Window = 5	LinearSVC	Pos	0.87	0.87	0.87	0.87028
		Neg	0.87	0.87	0.87	
Dim = 250, Window = 5	LinearSVC	Pos	0.88	0.86	0.87	0.87168
		Neg	0.87	0.88	0.87	
Dim = 250, Window = 5,	LinearSVC	Pos	0.88	0.87	0.87	0.87252
With white space		Neg	0.87	0.88	0.87	
Dim = 250, Window = 5,	LinearSVC	Pos	0.87	0.87	0.87	0.87192
With white space, no digits		Neg	0.87	0.87	0.87	

Table 1: Result table

From the results depicted in Table 1. we conclude that out of the tested classifiers, the LinearSVC classifier works best. When experimenting with the dimension size, increasing the dimension results in slightly better performance. For the window size, where a smaller value seems to have a positive impact. Replacing punctuation with white space improve the model slightly whereas removing digits does not.

5 Discussion and Conclusion

In conclusion, or model achieves the best result with the LinearSVC with higher number of dimension and smaller window size. When pre-processing, replacing special characters by white space instead of removing it has a slight positive impact. In order to further improve our model we could look into running more epoch for training or doing a grid-search on the parameters such as dimension and window size. When doing the data exploration, we realize spelling mistakes can be made. It would be possible that training a model correcting these mistakes would enhance performance.