M.Sc. In Data Science
DSC 510: Introduction to Data Science and Analytics
Homework Project

Nikolas Petrou

University Of Cyprus
Department of Computer Science
petrou.p.nikolas@ucy.ac.cy

1 Overview and Motivation

1.1 Overview

The goal of this work is to go through a complete data science process, in order to answer questions about a chosen topic. The process involves the data acquirement, design of the visualization, data analysis and results.

The main part of the work focuses on the exploration and study of different techniques which are used for sentiment analysis. In addition, the work utilizes and compares different approaches for classification tasks in Natural Language Processing.

1.2 Motivation

Natural Language Processing has been extensively studied and applied for the last couple of years. Natural Language Processing (NLP) is the process of analysing text using computational methods and algorithms. More precisely, it is a set of theoretically based computer approaches for evaluating and modelling text, in order to achieve a human-like language processing behaviour for a variety of activities and applications. That human-like processing approach reveals the true field of NLP, which is Artificial Intelligence and Machine Learning [1].

There are a lot of applications in NLP which help businesses or individuals. Some examples are translation, automatic language grammatical error correction and image description which is an application that combines both computer vision and NLP [2].

Other fundamental NLP applications involve sentiment analysis of textual information. Sentiment analysis (SA) can be characterized as the process of opinion mining, which determines whether the opinion is positive, negative or neutral [3]. In addition, SA is mostly performed on textual data with the utilization of NLP methods. A simple example of a SA application, is the monitoring of sentiment in product reviews during customer feedback, to obtain knowledge about the customer needs [4].

With an ever-increasing production of movies and online open data, there is an excess amount of reviews and opinions on films. Film reviews are extremely useful in helping people organize and choose movies. Apart from that, sentiment classification in reviews helps to extract meaningful information from opinions, and it is a critical portion of analysis.

With the foregoing in mind, it was decided that this work will explore different text manipulation and language processing techniques which will be used for the task of text analytics and classification of a movie review dataset.

1.3 Methodology and Limitations

The methodology of the work is illustrated in the scheme of Figure 1. Summarizing, firstly the initial questions were set in respect to the used dataset. Subsequentially, the data scrapping and data collection were performed. In addition, after the data preprocessing steps were performed, different data analytics and analysis were employed in order to better understand the data insights. Finally, during the final analysis, different methodologies and models were utilized in order to classify the textual data based on the sentiment. It is crucial to mention that the whole processed followed a cyclical scheme.

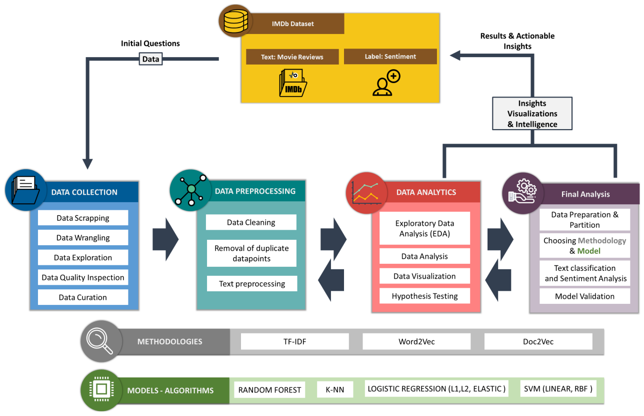


Figure 1: Summary of the followed methodology

Undoubtedly, the specific work had some limitations. Regarding the movie review dataset, it only includes highly polar film reviews from a single website, where the data did not include any neutral reviews, therefore studies could have been conducted only for reviews from either the positive or negative class. Furthermore, the reviews were mostly written before 2011, thus any extracted information or insights might not be the most ideal-representative for all of the recently released films. In addition, no meta-data were included for the different datapoints (e.g. information of the movie or reviewer). Finally, regarding the computational resources used, the whole implementation and experiments were conducted locally, using only a single CPU of 16 cores. Therefore, since no Cloud Services or GPU was used for any of the components of the work, there were some limitations for the implementation of the Machine Learning algorithms.

2 Related Work

2.1 Inspiration

First, it was very enthusiastic watching the lectures about Information Retrieval and NLP during the class and lab practical sessions. My curiosity rose on how the theory could be practically applied in real-world applications. Furthermore, as a personal goal to learn more about Machine Learning, NLP and textual data analysis, it was very motivating and inspiring to work with text data.

Last but not least, it was inspiring to look at relative work of other individuals. For example, it was astounding watching the relative work of my cowrorker, Mr. Demetris Paschalides [1], who is specialized in applications that involve textual data, as well as the work of my ex-supervisor, Associate Professor Mr. Grigorios Tsoumakas [2], who is specialized in Machine Learning and has worked around NLP.

2.2 Approaches for text classification

This work is mostly focused on the task of text classification. Similarly to all the supervised learning problems, classification tasks and algorithms require a set of features which will be used to predict the target variable. It is well known that computers simply cannot process text data in raw format. Instead, it is required to encode the textual data into a numerical format which can be processed by the machine.

The traditional approaches for the modelling, are the Bag-of-Words (BoW) and TF-IDF [5]. Both BoW and TF-IDF are techniques that can easily convert text into numeric vectors.

The idea behind BoW is the simplest form of text representation in numbers, where documents are represented as bag of words vectors. In simple words, the frequency of each word of the vocabulary is represented as a feature in the numeric vector, which represents the document. Of course, due to its simplicity, there are some major drawbacks of using this method for text encoding. The major drawback, is that terms frequency is not necessarily the best representation for text. In fact, frequent words in the document which are also common words in the corpus (the total set of documents), can have only little predictive power over the target variable [6].

The ”advanced” alternative of the BoW, which is TF-IDF [3] representation, addresses the above-mentioned issue. Specifically, instead of simple word-term frequencies, the TF-IDF score for a term t in a document d increases proportionally to the frequency of the term in the document (denoted as $tf(t,d)$), but it is inversely proportional to the frequency of the term in the corpus D (denoted as $idf(t,D)$).

The problem of both methods, is that since most of the vocabulary words are used as a feature, a significant dimensionality problem arises; the more documents, the larger is the vocabulary, so the feature vectors will form a huge sparse matrix with many zeros. Therefore, the BoW and TF-IDF models are usually preceded by an extensive pre-processing procedure (term cleaning-normalization, stop words removal, stemming or lemmatization), which aim to reduce the total features used. Finally, no information on the grammar of the sentences nor on the ordering or context of the terms in the text are utilized in these methods.

Apart from the traditional methods that were mentioned, there are modern approaches that focus on mapping the words of a vocabulary to vectors. These feature learning methods belong to the Word Embedding (WE) techniques. In WE methods, the resultant word vectors, which will appear close in the vector space, usually appear before or after another and have similar context in the initial corpus [7]. There are several Word Embedding algorithms (e.g. Word2Vec or GloVe [8]), and most of the methods are using a Neural-Network of three layers (input, hidden and output layers) in order to generate the word vectors. What is interesting and important about WE, is that by averaging the word vectors of a document, it is possible to obtain a reduced number of features, compared to the traditional BoW or TF-IDF approaches. Moreover, pretrained WE which were constructed in one task, can be used or fine-tuned for solving another similar task, which is a method of Transfer Learning.

Finally, the most advanced and latest methods are the Language Models (Dynamic Word Embeddings), which overcome the biggest limitation of polysemy disambiguation, that classic WE are accused of. Language Models do not use fixed embeddings, but by using bidirectional Deep Neural Networks, they look at the specific sentence and then assign an embedding to each word of the sequence. Currently, the state-of-the-art Language Model is Google's BERT, which uses Transformers to dynamically assign different vectors to words based on the context [9].

2.3 Existing work on review datasets

There have been many studies and works which target the classification of review sentiment datasets. In particular, there have been studies in which tree based classifiers (e.g. Decision Trees and Random Forest) were employed for SA tasks [10]. Also, there are several existing studies which employ Deep Learning models (e.g. Traditional MultiLayer Perceptron, Recurrent Long Short-Term Memory Neural Networks and Convolutional Networks) [11] [12]. Finally, during the last couple of years, pretrained BERT models were studied and employed in order to be fine-tuned for sentiment datasets [13].

3 Initial questions

As mentioned, the work was mostly focused on the task of text classification and to the question of how the different methods of the literature could be applied in real-world data. Some related initial questions were the following:

- *”What are the real difficulties of using Machine Learning and NLP methods for big textual data?”*
- *”Which classification models are the ideal to use for the sentiment data. What are their constraints and their particularities?”*
- ”How do people express themselves in online movie reviews?”
- ”Could some interesting discoveries about online film reviews be extracted?”

Undoubtedly, during the implementation and analysis of the work, more questions appeared. The original idea was to implement algorithms only around the traditional text encoding methods (BoW and TF-IDF). But, as it will be explained later, the extensive size of the corpus' vocabulary did not permit for the employment of many algorithms.

- *”Will Word Embeddings allow for easier experiment of more algorithms and models?”*
- *”How does transfer learning with the already implemented word embeddings perform on the sentiment data?”*
- *”Can BERT be easily employed and used?”*

4 Data

For this work, a large dataset which consists of movie reviews was used. Specifically, the publicly available Internet Movie Database (IMDB) review dataset [14].

[1]`http://linc.ucy.ac.cy/index.php?id=144`
[2]`https://intelligence.csd.auth.gr/people/tsoumakas/`
[3]`https://en.wikipedia.org/wiki/Tf-idf`

4.1 The IMDB dataset

The IMDB dataset consists of 50,000 highly polar film reviews. It is mostly used for binary sentiment classification tasks, with the employment of supervised learning methods.

The data was originally scraped from the *IMDB* website[4], which is the world's most popular source for film content, containing ratings, reviews and news for every released movie. During the scraping procedure, the dataset's developers collected only strongly positive or strongly negative reviews from the IMDB's website. In particular, a film review's sentiment l with an original review score s, was labelled as following:

$$l(s) = \begin{cases} positive, & |s| \leq 7 \\ negative & |s| \geq 4 \end{cases}, s \in \mathbb{S}$$

where:

$$\mathbb{S} = \{x \in \mathbb{N} \mid 0 \leq x \leq 10\}$$

The original data collectors did not include any neutral reviews in this dataset, therefore only reviews from either the positive or negative class are involved.

For this work, the dataset was downloaded-scraped directly from Kaggle's website, where it is available in a structured comma-separated values (CSV) file.

4.2 Dataset inspection and cleaning

Before the analysis phase, it was essential for the scrapped data to be inspected and cleaned due to its textual nature. The initial dataset consisted of only two features, the review and sentiment's label. Both of the features contain textual data, where the label may only have one of the two unique values (positive or negative), while the review feature contains a large amount of characters. A segment of some data points is presented in Table1.

Throughout the data inspection, it was first examined and assured that the dataset does not contain any null or non-registered values. Moreover, by looking into the unique values of the different features, it was observed that there were several duplicate data points. Eventually, it was distinguished that many unpleasable text characters, such as HTML tags, URLs or non-English characters existed (Table1).

Review	Sentiment
A wonderful little production. <br /> <br />The filming technique is very unassuming- very old-time-B...	positive
A rating of "1" does not begin to express how dull, depressing and relentlessly bad this movie is.	negative

Table 1: Segment of initial dataset

4.3 Data cleaning

During the clean-up phase, the different findings of the data inspection were taken into account in order to produce a better and more clean dataset. As Tomas Mikolov [5] mentioned, some text cleaning is required in NLP [15]. The steps of the cleaning phase were the following:

- **Removal of duplicates**: Since exact duplicate data points are an extreme case for the specific dataset, and since they bias machine learning models, it was crucial to remove duplicated rows from the dataset.
- **Strip of HTML and links**: The HTML tags and URLs-links were removed, since they are regarded as "noise" and can introduce bias to the models. In fact, links could have been replaced with a specific token, but the links were removed completely after it was observed that the event of a URL link to be included in the review was not so likely to happen (only 187 out of the 50000 reviews had included URL links).
- **Expanding contractions**: In order to reduce the vocabulary, and keep the identical words in the same form, contractions were expanded to separated words. For instance, the text *"I'm very positive that this movie wasn't good"*, was expanded to *"I am very positive that this movie was not good"*.
- **Removal of unpleasable characters**: Since the dataset was meant to be constructed from film reviews in English, non UTF-8 characters were removed. Furthermore, punctuation marks and unnecessary white spaces were removed.
- **Conversion to lowercase**: All characters were converted to lowercase. In general, it is usually a good practise to include only lowercase characters, as Mikolov mentioned [6].
- **Lemmatization of words**: Lemmatization is one of the most widely used text pre-processing techniques used in NLP. With lemmatization, the inflectional forms of a word are reduced, and sometimes different related forms of a word are reduced-transformed to a common base form. Consequently, related words (e.g. *studies*, *studying*) will be transformed to the same word and the total word vocabulary is greatly reduced.
- **Tokenization**: Tokenizing words of the documents. In other words, a conversion of documents to a list of smaller units (tokens) was performed.
- **Partial use of N-grams**: Since by adding all the bigrams or trigrams would lead to a huge vocabulary that could not be easily processed, only the most frequent bigrams/trigrams of words were detected and used as separated tokens (e.g. *"not good"* or *"very bad"*). More on that will be presented during the next chapters.
- **Removal of stop words**: Words that are frequently used in general were removed since they will not provide any additional information for the models, and the smaller the vocabulary is, the lower is the memory complexity. For that, the NLTK library [16] was utilized, since several frequently used words are already present in the library's stop words list (e.g. *my, he, she, it*). Finally, the NTLK's list contains the word *"not"*, where in this work it was decided to keep it, since many bi-grams which contain the *"not"* word can have the reserve meaning for a sentence. In addition, words that their length is unusually long (greater than 15) or lower than 2 (e.g. 'I'), were also removed.

5 Exploratory Data Analysis

During this stage, the data was analysed and examined, in order to summarize and make conclusions about the main characteristics, as well as to determine if the statistical techniques which were considered are appropriate.

Initially, it was ascertained if the dataset had missing or null values that resulted from the pre-processing operations or were present from the start. Remarkably, there were no missing or null values in the dataset.

In general, because the dataset is primarily textual, the visualization options were relatively constrained. Nevertheless, some key data visualization methods and hypotheses tests were performed in order to draw out some patterns and conclusions.

[4] `https://www.imdb.com/`

[5] `https://dblp.org/pid/45/8055.html`

[6] `https://groups.google.com/g/word2vec-toolkit/c/jPfyP6FoB94/m/tGzZxSc0OGsJ`

5.1 Distribution of dataset's classes

Most importantly, it was firstly explored how the population of the two sentiment samples fluctuates.

The duplicate sample removal phase was avoided in the majority of the existing analysis for the specified dataset. Therefore, the existing work showed that the two classes (positively and negatively labelled samples) are perfectly balanced. On the contrary, this work shows and proves that by eliminating the duplicate data (Section 4.3), the distribution of examples across the known classes is not perfectly balanced, as it is shown in the pie-chart of Figure 2. Even though the frequency between the two classes only varies by 0.4%, it should be taken into consideration, as a classification model will require a prior stratification process before the training.

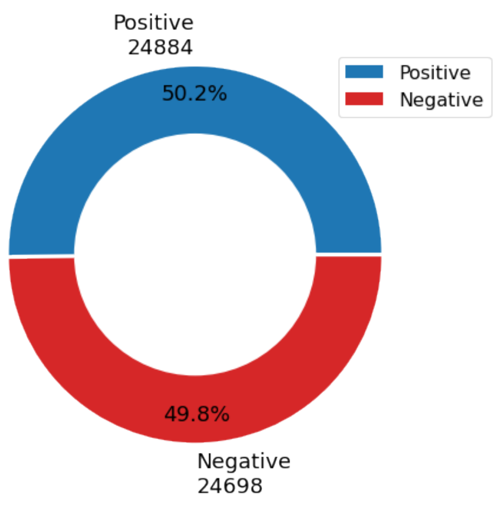
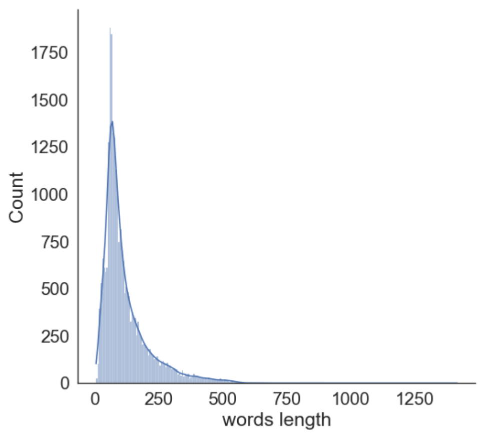


Figure 2: Distribution of known classes

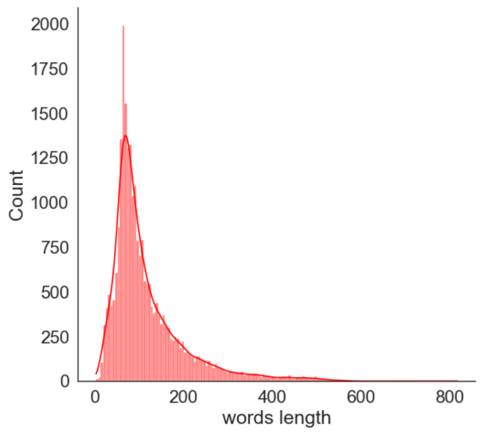
5.2 Amount of words in reviews

Following, in was examined and observed how the total amount of words has an impact on the sentiment of the reviews. For that, a histogram was plotted individually for the two different kind of reviews (Figure 3). On top of the histograms, a kernel density estimate (KDE) was plotted, as a way to visualize the distribution of observations in the two populations, with a continuous Probability Density Function curve.

By observing both of the histogram plots of Figure 3, it is clear that both the distributions are skewed to the right, thus are positively skewed. For that reason, both of the distributions have numerous occurrences in the lower value bins (left side) and few in the upper value bins (right side). Additionally, it also indicates that the median of the two populations is less than the mean, and that the direction of outliers are mostly presented on the right side of the distribution. Eventually, it seems that the positive reviews' histogram is a bit more positively skewed than the positive. Thus, on comparison with the negative reviews, there are more positive reviews where the reviewers wrote an extensively large document.



(a) Words in positive reviews



(b) Words in negative reviews

Figure 3: Words length of reviews

This was further analysed by looking at the box-plots of the two populations, and by running a hypothesis test, in order to identify the significance of the difference of means in the two populations. The box-plots of Figure 4 showed that the interquartile range (IQR) of the samples is not very different. Even so, the major difference of the two populations is on the level of the outliers, which happen to be outside their upper hinge, as it was already mentioned.

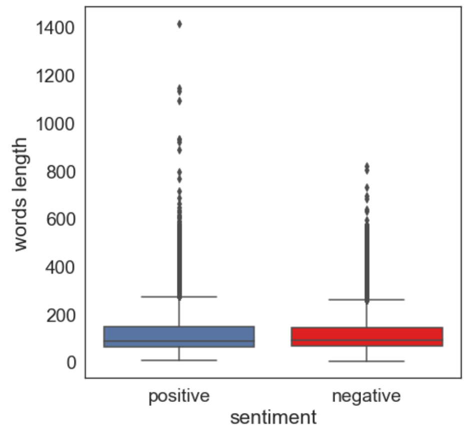


Figure 4: Words in reviews - Boxplots

Next, by running a two-tailed t-test it was determined how significant the differences between the two groups are:

- H_0 **(null-hypothesis)**: The difference in the means of the two groups is zero
- H_a **(alternative-hypothesis)**: The difference in the means of the two groups is not zero
- CI **(confidence interval)**: This is the range of numbers within which the true difference in means will be CI% of the time. In this case CI = 99% was chosen. Consequently, p-value must be less than 0.01 in order for the H_0 to be rejected.
- Results:
  - Statistic: t = 3.60156
  - P-value: pvalue = 0.00032

The t-test showed that the p-value is much smaller than 0.01, so the null hypothesis H_0 of no difference was rejected. With a high degree of confidence, the true difference in means of the groups is not equal to zero. Therefore, as a finding, due to the fact that there are some cases (outliers) where the positive reviews happened to have an extensive amount of words included, positive reviews have larger total word count in general.

5.3 Analysis of frequent n-grams

In order to explore the writing behaviour of the reviewers, several bar-plots which contain the frequency of the different n-grams were constructed. Specifically, the bar plots were created individually for the different sentiment groups. The frequencies of the top ten unigrams, bigrams and trigrams are illustrated in Figures 1, 2 and 3 respectively. All of those commonly used words and n-grams will be useful throughout the sentiment classification process.

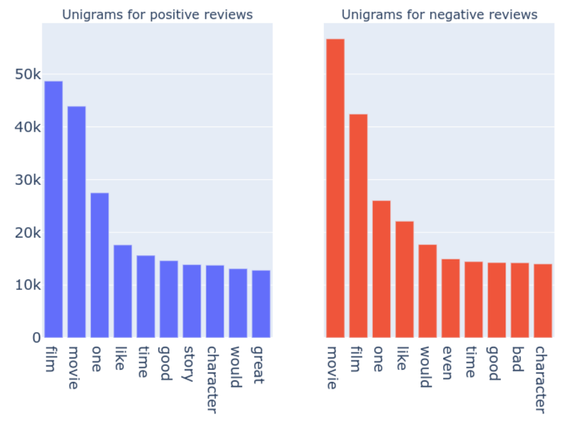


Figure 5: Most common unigrams sentiment wised

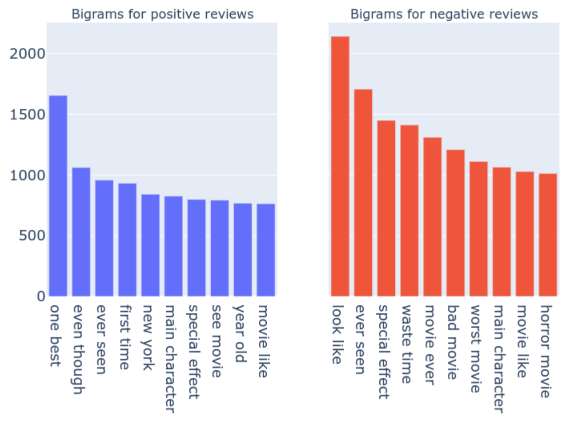


Figure 6: Most common bigrams sentiment wised

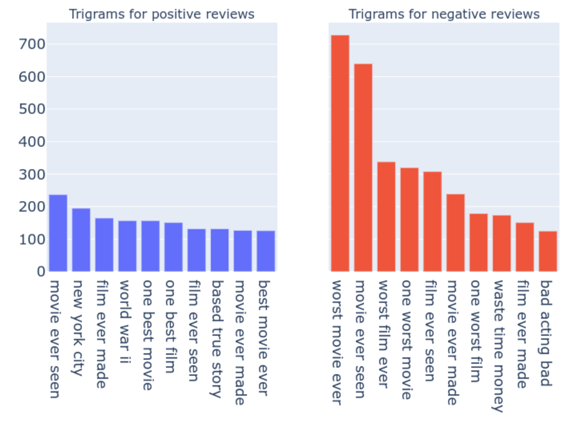


Figure 7: Most common trigrams sentiment wised

By looking at the common unigrams, not many conclusions can be raised. However, it is shown that the word *"character"* is presented at the top-10 most frequent words of both of the review subsets, which shows how important the characters of a film are for the reviewers. Into the bargain, there are commonly used words which express opinion or sentiment (e.g. *"like", "great", "good", "bad"*) in both of the review sub-sets.

Following the bigrams include much more information. It can be clearly seen the most frequently used bigram of the positive reviews is the *"one best"*, which shows strongly positive opinion. On the other hand, the negative reviews' text tend to include the bigram *waste time*. An interesting discovery is that both of the barplots show that the bigram *"special effect"* is widely employed, which can be explained by the fact that special effects in movies is an important factor for the reviewers. Ultimately, another fascinating observation, is the fact that the *"horror movie"* bigram is heavily used in the negative reviews. That is defended by the fact that many low-rated movies[7] in IMDB are horror movies.

In closing, even though the frequently used trigrams are not so heavily used, there are some phrases which include very strong sentiment. For instance, the negative reviews often include phrases like *"worst movie ever", "worst film ever"* and *"waste time money"*,

[7] https://www.imdb.com/chart/bottom

while the positive reviews often include trigrams like *"one best movie", "one best film"* and *"best movie ever"*.

6 Final Analysis

During the final part of the analysis, the main work which was the sentiment detection in the text was conducted.

6.1 Modelling as a supervised task and data split

In order to answer the question of which model performed best for the classification task, the occurred tokens of the reviews-documents were treated as features (independent variables), while the sentiment label was regarded as the target variable to the supervised learning problem. The pre-processing procedure of Section 4.3 had produced 95972 tokens-features which also included frequent bi-grams.

During the fit and evaluation of the classification algorithms, it was decided to split the data samples, as it is described in table 2. Notice that there is no validation portion so far, since the k-fold cross validation (CV) procedure will be performed later on the train portion of the data samples. Furthermore, since the explanatory analysis of Section 5.1 showed that the labelled sentiments are not exactly balanced, stratification was performed during the split procedure. Stratification in data split ensures that the training, validation and test sets will have the same proportion of the target variable as in the original dataset [17]. Doing this with the sentiment feature ensures that the result is a close approximation of generalization error.

	PROPORTION	QUANTITY
Total	100%	49582
Training	80%	39665
Test	20%	9917

Table 2: Proportions of data split

6.2 The traditional approaches

The initial idea was to train and compare different algorithms with both BoW and TF-IDF approaches, which were mentioned in Section 2.2. Firstly, the two approaches were trained and evaluated with the Logistic Regression (LR) and Random Forest Trees (RF) aglorithms, in order to get an idea of how strain and computationally complex the task is. The accuracy and performance of the algorithms are demonstrated in table 3.

ALGORITHM	ACCURACY	F1-SCORE	TRAINING TIME
LR - BoW	0.885	0.885	201sec
RF - BoW	0.880	0.879	540sec
LR - TFIDF	0.894	0.894	198sec
RF - TFIDF	0.892	0.892	535sec

Table 3: Evaluation of BoW and TF-IDF methods

Consequently, the TF-IDF approach has lead to slightly better accuracy results. Evidently, the feature space was enormously large, since every training sample which was fitted in the classification algorithms had 95972 features. That led to memory and performance issues, thus it was hard to run more experiments with other algorithms or to search for the optimal hyperparameters of a specific model with the use of a k-fold cross validation. Moreover, since for the above-mentioned reasons, it was not possible to perform a hyperparameter tuning at this stage of the analysis. Thus, in order to reduce the training time of the classification algorithms, and use better algorithm hyperparameters, it was decided to reduce the feature dimensions by taking advantage of the Word Embeddings.

6.3 Implementation of Word Embeddings

Towards obtaining word vectors, several WE techniques were employed. Distinctively, the following methods were applied:

- Averaging Word-Vectors (Mean Word2vec)
- TF-IDF Weighted Averaging Word Vectors (TF-IDF Mean Word2vec)
- Averaging of GloVe Word-Vectors (GloVe Mean Word2vec)
- Documents as vectors (Doc2vec)
- Concatenation of TF-IDF Mean Word2vec and Doc2vec

The Word2Vec and Doc2Vec vectors were extracted by training Deep Neural Networks and by utilizing Gensim [18], which has efficient multicore implementations of the popular algorithms. The pre-trained GloVe word vectors [8] were scrapped directly from Stanford's website [8]. Due to hardware limitations, only the light version of GloVe (6B tokens, 400K vocabulary) was used.

Training the Neural Networks and extracting the vectors took about 3-4 hours, but after that the dimensions of the original dataset were greatly reduced, and thus more experiments and algorithm trials could be performed. Specifically, in order to look over different vector dimensions, it was chosen to examine both 100, 200 and 300 vector sizes. Firstly, the embedding methods were evaluated with the Logistic Regression algorithm in order to see how the results varied from the approach of 6.2. The results of the initial evaluation are presented in Table 4, while the training times of the methods are presented in Table 5

WORD EMBEDDINGS DIMENSIONALITY = 100		
METHOD	ACCURACY	F1-SCORE
Mean Word2vec	0.880	0.880
TF-IDF Mean Word2vec	0.878	0.878
GloVe Mean Word2vec	0.797	0.796
Doc2vec	0.872	0.872
TF-IDF Word2vec + Doc2vec	**0.892**	**0.892**

WORD EMBEDDINGS DIMENSIONALITY = 200		
METHOD	ACCURACY	F1-SCORE
Mean Word2vec	0.885	0.885
TF-IDF Mean Word2vec	0.881	0.881
GloVe Mean Word2vec	0.814	0.814
Doc2vec	0.866	0.866
TF-IDF Word2vec + Doc2vec	**0.894**	**0.894**

WORD EMBEDDINGS DIMENSIONALITY = 300		
METHOD	ACCURACY	F1-SCORE
Mean Word2vec	0.884	0.884
TF-IDF Mean Word2vec	0.881	0.881
GloVe Mean Word2vec	0.822	0.822
Doc2vec	0.869	0.869
TF-IDF Word2vec + Doc2vec	**0.892**	**0.892**

Table 4: Logistic Regression for Embedding methods

[8]https://nlp.stanford.edu/projects/glove/

WORD EMBEDDINGS DIMENSIONALITY = 100	
METHOD	TRAINING TIME (SECONDS)
Mean Word2vec	1.73
TF-IDF Mean Word2vec	0.97
GloVe Mean Word2vec	2.16
Doc2vec	0.51
TF-IDF Word2vec + Doc2vec	2.31

WORD EMBEDDINGS DIMENSIONALITY = 200	
METHOD	TRAINING TIME (SECONDS)
Mean Word2vec	3.759
TF-IDF Mean Word2vec	2.373
GloVe Mean Word2vec	4.51
Doc2vec	0.779
TF-IDF Word2vec + Doc2vec	6.16

WORD EMBEDDINGS DIMENSIONALITY = 300	
METHOD	TRAINING TIME (SECONDS)
Mean Word2vec	3.49
TF-IDF Mean Word2vec	2.49
GloVe Mean Word2vec	5.58
Doc2vec	0.76
TF-IDF Word2vec + Doc2vec	9.37

Table 5: Training time of Embedding methods

As it was expected, even though most of the methods had similar-same performance results with the traditional approaches of section 6.2, the running times were greatly reduced, since for all the methods, the algorithms require only a few seconds to be trained. Visibly and as expected, as the length of the word vectors is increasing, the training times of the classification algorithms are increasing as well. It is significant to mention that the word embeddings seem to work almost similarly well, regardless of their dimensionality. Furthermore, the approach of combining the embeddings of TF-IDF Mean Word2vec and Doc2vec had the highest accuracy. Another point is that the GloVe embeddings did not perform as good as the other methods, probably due to the fact that only the light version was used. But, it is fair to mention that the GloVE embedding vectors are employed in a Transfer Learning manner, and were not constructed-trained at all for the specific dataset. That fact shows the power behind Transfer Learning and generalization in Machine Learning.

6.4 Comparison of classification algorithms for combined embeddings of TF-IDF Mean Word2vec and Doc2vec

During this phase, since the combination-concatenation of the TF-IDF Mean Word2vec and Doc2vec showed promising results, more classification algorithms were employed to study their behaviour. It is crucial to mention that since the features were far less compared to the approach of Section 6.2, thus training times were greatly reduced, a 10-fold CV was furthered performed during the next steps in order to get more ideal results. It was chosen to continue with the embeddings of size 100, since the tradeoff between the performance and efficiency is relatively low.

The algorithms which were employed by using grid-search and cross validation for the optimal hyperparameter tuning were the following:

- Lasso-L1 Regression (L1-LR)
- Ridge-L2 Regression (L2-LR)
- Elastic-Net Regression (ELNET-LR)
- Random Forest Trees (RF)
- K-NN Classifier (K-NN)
- Weighted K-NN Classifier (W-KNN)
- Support Vector Classifier with Linear Kernel (Linear-SVC)
- Support Vector Classifier with Radial Basis Function Kernel (RBF- SVC)

The Language Model of BERT was also tried to be trained and used. Unfortunately, the whole implementation is undoubtedly GPU dependant, and would have required a lot of effort and training time. Therefore, BERT was not employed for this work.

After implementing and applying the above-mentioned algorithms, the optimal hyperparameters were found, and the models were employed on the unseen data of the test set. The models were ranked based on their accuracy results in table 6.

ALGORITHM	ACCURACY	F1-SCORE
L1-LR	0.892	0.892
L2-L	0.892	0.892
ELNET-LR	0.892	0.892
RF	0.849	0.849
K-NN	0.848	0.847
W-KNN	0.850	0.849
Linear-SVC	0.893	0.892
RBF-SVC	**0.899**	**0.899**

Table 6: Evaluation after 10-fold CV hyperparameter tuning

All the linear models achieved similar results. It was noticed that best results were achieved with the RBF-SVC which was the most ideal model for this task after the tuning. Surprisingly, the RF model did not perform as well as most of the models, which is probably due to the fact that the transformed data were linearly separable. Finally, the Weighted-KNN performed slightly better than the simple KNN.

The RBF-SVC which was the most ideal model, was optimized by tuning its C hyperparameter. The C hyperparameter, which is very critical in support vector machines, trades off misclassification of training samples against simplicity of the decision surface. The graph of Figure 8 shows how the training and test accuracies varied with different C values for the specific dataset. If the C value was chosen to be very large, then the model would have overfit to the training data (increased variance). On the other hand if the value of C was too low, then the model would not have been complex enough to capture the data's particularities (increased bias). That is why C was chosen to be 1, which is the ideal value that balances the trade-off between the variance and bias in the data.

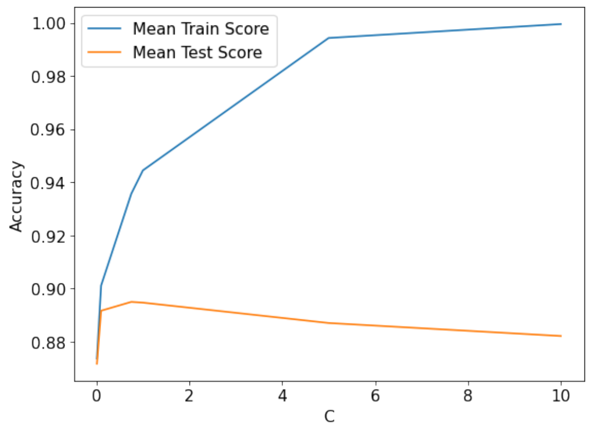


Figure 8: Mean accuracy score for different C values

Finally, the confusion matrix for the test set is illustrated in Figure 9. It can be seen that the classification between the two classes is well-balanced, which can be also confirmed from the occurred F1-score, since it has the same score as the accuracy.

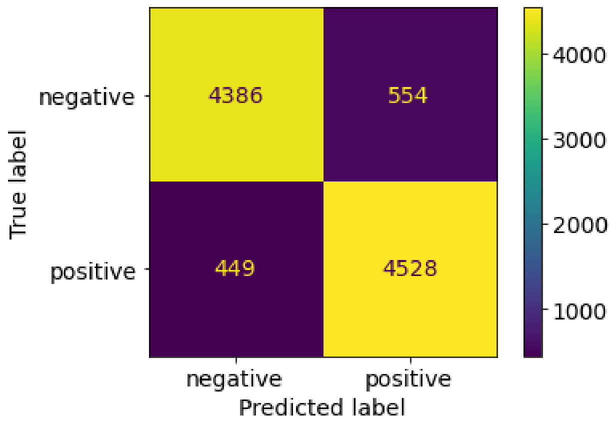


Figure 9: Confusion matrix

6.5 Conclusions

In this work, some discoveries concerning online highly polar movie reviews were to be found. Besides, questions regarding the methodologies and techniques for classification of textual data were to be answered.

It was found with a statistical significance from the analysis of section 5, that in general, positively reviewed movies have more total words than the negatively reviewed. In addition, some interesting finding is that reviewers had extensively included information about the characters of the films, which shows their importance. To boot, it was found that usually the positive and negative reviews include words and phrases which express opinions or strong sentiment (e.g. ”like”, ”bad”, ”worst film ever”, ”best movie ever”). Conclusively, there were negatively reviewed movies which had the ”horror movie” bi-gram included, which shows how the horror movies are treated as ”bad” movies.

On the sentiment classification task, it was found that the traditional BoW and TF-IDF require unbelievably amount of computational resources to be employed. Even though the preprocessing phase had included stop-word removal, due to the huge vocabulary of the dataset the time complexity of the employed algorithms was enormous. Instead, the word vectors approach was found to be an ideal choice for experimenting with different algorithms and models. Surprisingly, the procedure of training and extracting word vectors was found to be straightforward. With reference to the classification algorithms, it was concluded that the Support Vector Classifier with Radial Basis Function Kernel was the most ideal among the employed models, which were cross validated and fine-tuned.

Transfer Learning with GloVe word embeddings was found to be easily applicable, and the methodology could have probably been even more efficient if the larger GloVe file was used. In closing, as it was mentioned, BERT is a Language Model that generates contextualized word embeddings. Its disadvantage is that it is a very computationally intensive, and therefore not everyone can take advantage of it.

References

[1] Elizabeth D Liddy. Natural language processing. 2001.

[2] Haiqin Yang, Linkai Luo, Lap Pong Chueng, David Ling, and Francis Chin. Deep learning and its applications to natural language processing. In *Deep learning: Fundamentals, theory and applications*, pages 89–109. Springer, 2019.

[3] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014.

[4] Dongjoo Lee, Ok-Ran Jeong, and Sang-goo Lee. Opinion mining of customer feedback data on the web. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 230–235, 2008.

[5] Cong-ying Shi, Chao-jun Xu, and Xiao-Jiang Yang. Study of tfidf algorithm. *Journal of Computer Applications*, 29(6):167–170, 2009.

[6] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842, 2010.

[7] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[8] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Arman S Zharmagambetov and Alexandr A Pak. Sentiment analysis of a document using deep learning approach and decision trees. In *2015 Twelve international conference on electronics computer and computation (ICECCO)*, pages 1–4. IEEE, 2015.

[11] Nehal Mohamed Ali, Marwa Mostafa Abd El Hamid, and Aliaa Youssif. Sentiment analysis for movies reviews dataset using deep learning models. *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol*, 9, 2019.

[12] Alec Yenter and Abhishek Verma. Deep cnn-lstm with combined kernels from multiple branches for imdb review sentiment analysis. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 540–546. IEEE, 2017.

[13] Hairong Huo and Mizuho Iwaihara. Utilizing bert pretrained models with various fine-tune methods for subjectivity detection. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, pages 270–284. Springer, 2020.

[14] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[15] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*, 2018.

[16] Edward Loper and Steven Bird. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[17] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 145–158. Springer, 2011.

[18] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.