UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

TEXT MINING

FINAL PROJECT

CLASSIFICATION AND CLUSTERING OF IMDB DATASET

Autori:
Matteo Cesaro - 867350 - m.cesaro1@campus.unimib.it
Francesco Martinelli - 873685 - f.martinelli21@campus.unimib.it
Cristiano Ruttico - 809360 - c.ruttico@campus.unimib.it

Febbraio 2022

Contents

1 Introduction

In recent years, the activity of texting data on the internet has increased dramatically.

As a matter of fact, there are several websites available where anyone can post comments after watching a movie, using a product, or reading a book.

These comments can be analyzed through NLP or text mining in order to better understand people's opinion or feeling about everything.

The main purpose of this study is to analyze the reviews on IBDM.[1]

Figure 1: Movie reviews

IBDM is the largest movie website globally and its archive contains over 150 million movie reviews.

The approaches used for this analysis are:

- **Classification:** the algorithm's aim is to capture the difference between positive and negative reviews on IBDM and to predict them with an higher accuracy.
- **Clustering:** this method sorts the documents into different groups based on different rules.

For this study, both classification and clustering are applied.

2 Dataset

For this analysis, the IMDB dataset available at the following website [2] is considered.

The dataset consists of 50,000 movie reviews divided into two different directories: training set and test set. Each movie review is saved as a txt file.

Both training set and test set are also split into negative reviews and positive reviews.

We chose to put all reviews together in the dataset in order to make a better division of our data, associating negative reviews with the label '1' and positive reviews with the label '0'.

Subsequently, the reviews were split again using the 'train_test_split' helper method from scikit_learn in:

- **Training Set:** used to train the model.
- **Test Set:** used to evaluate and compare the models.

In 'Figure 2' the wordcloud graph of the dataset can be seen. Wordcloud represents the words with the most occurrences in the documents.

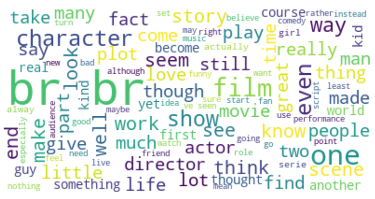


Figure 2: Dataset Wordcloud

3 Preprocessing

In order to improve the performance in the classification and clustering tasks, the first step of a text analysis is to preprocess the text data.

Text preprocessing is a method of cleaning and preparing the text data making them ready to feed the models.

3.1 Normalization

Text data contains noise in various forms like emoji, punctuation or numbers, therefore removing the noise is essential.

First of all, lower case and upper case are treated differently by a text mining algorithm, so it is necessary to transform each review to a fully lower structured sentence.

Another text processing technique is removing punctuations. There are a total of 32 main punctuations that need to be taken care of.

Secondly, the most common English words are removed from each file.

These types of words are called 'stop words' and are not needed to improve the overall performance.

Examples of stop words in our analysis are "a", "the", "is", "are" etc.

NLTK library is a python library that is often used to remove stopwords and considers approximately 180 stop words.

Removal of stop words are important in order to eliminate words that are frequently used but carry very little additional or useful information.

Subsequently, numbers, Urls and emojis are deleted from text data.

To remove numbers might make sense, since numbers contain no valuable information about the task of classifying and clustering positive and negative film reviews.

The same goes for the removal of URLs and emojis.

3.2 Stemming and Lemmatization

Another essential task in text mining analysis is Stemming or doing lemmatizazion of text data.

The first step of stemming and lemmatization is Tokenization.

Tokenization is a key process in text mining analysis. It consists in separating chunks of text into smaller units called tokens.

For this purpose, the following types of tokenization are used:

- **Whitespace Tokenizer:** Assumes space as a delimiter and splits pieces of text basing on space.
- **NLTK Word Tokenizer:** Divides a piece of text into individual words based on a certain delimiter.

Lemmatization or Stemming can then be applied. Both are used for this analysis in order to compare the performance with each other.

Stemming and lemmatizazion aimed at reducing the inflectional forms of each word into a common base or root.

However, these two methods are not exactly the same thing.

Basically, Stemming algorithm works by removing affixes at the end or beginning of a word in order to get its root stem.

On the other hand, lemmatizazion is a systematic process to convert the words into their lemma by matching them with a language dictionary whilst considering the of morphological analysis of each word.

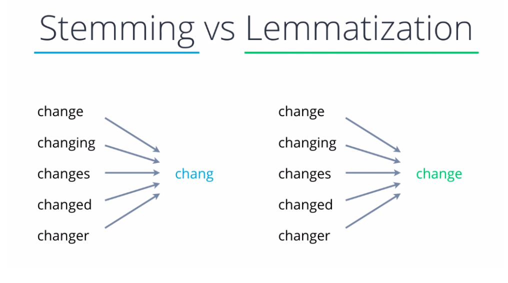


Figure 3: Stemming and Lemmatization

Both Stemming and lemmatizazion are applied using NLTK library.

Precisely, for stemming the nltk_lancaster and nltk_porter method are used.

3.3 Text representation

The last and central step in text mining analysis is text representation. For this study, TF-IDF approach is used.

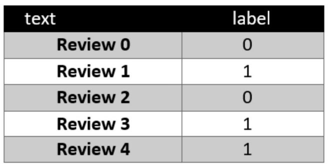
- **TF:** considers how important a word may be through term frequency, without including stop words already removed.
- **IDF:** since TF often incorrectly emphasizes documents which use more frequently word with no valuable information, IDF decreases the weight of this types of word and increases the weight for other less used words.

This combination is called TF-IDF and it describes the frequency of a term adjusted for how rarely it is used.

4 Classification

We have at our disposal a corpus of about 50'000 imdb reviews; the aim of this part of the analysis is to implement and evaluate methods to learn to recognise which reviews are positive and which are negative.

The data obtained after the application of the pre-processing part is presented in tabular form; below is an example of the structure of the data at our disposal.



text	label
Review 0	0
Review 1	1
Review 2	0
Review 3	1
Review 4	1

Figure 4: Data Structure

In the field 'text' we have all the reviews, in the column label we have the value 1 if the review expresses a negative opinion, and instead we have value 0 when it expresses a positive opinion.

4.1 Preprocessing

In order to carry out the analysis and subsequent training of the models, we split the data into training and testing. Splitting is done using the train_test_split method of the sklearn library. A 75-25 split is performed. We allocate 75% of the data to the training phase and the remaining 25% to the test phase.

After an initial investigation, we notice a balance of classes: to be precise, we are faced with exactly 50% negative reviews and 50% positive reviews.

In order to obtain the features to be provided to our classifiers we used the TfidfVectorizer method: it allows us to go from documents to their representation by exploiting the Tfidf weighting: i.e. Term-frequencies(Tf), and inverse term frequencies(idf); this weighting allows us to represent documents as vectors of real numbers and to take into account the frequency of words within a single document and the importance they have among documents.

Below is the formula for calculating Tf-Idf used by sklearn:

$$Tfidf = TF(t,d) * \mathrm{IDF(t)}$$

With t: number of times a doc appears in a document d.

4.2 Models

Various supervised classification techniques were applied; specifically, our analyses were based on the use of the sklearn library.

Several classes of models were used in order to meet the objective:

- Regression-based-models: **Logistic Regression** was used.
- Separation-based-models: **Support Vector Machines** were used.
- Ensemble models: **AdaBoost** was used.
- Euristic models: **Decision Tree** and **Random Forest** were used.

4.3 Performance evaluation

A very important measure for assessing the performance of classification models is Accuracy.

$$Accuracy = \frac{\mathrm{TP + TN}}{\mathrm{TP + FP + TN + FN}}$$

In particular: TP and TN represent the number of instances belonging to the positive and negative class respectively that are correctly classified. FP e FN indicate the number of instances of negative and positive class misclassification.

The calculation of Accuracy as a reference measure for evaluating the model can be used effectively if the two classes are balanced, i.e. they are equally present in the dataset. In cases where the problem presents unbalanced classes, how should we proceed? It is necessary to introduce two further measures:

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP + FN}}$$

Also referred to as the True Positive Rate or sensitivity, it represents the portion of positive records correctly classified by the model; a low Recall value will indicate a high number of false negatives. We then have the

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP + FP}}$$

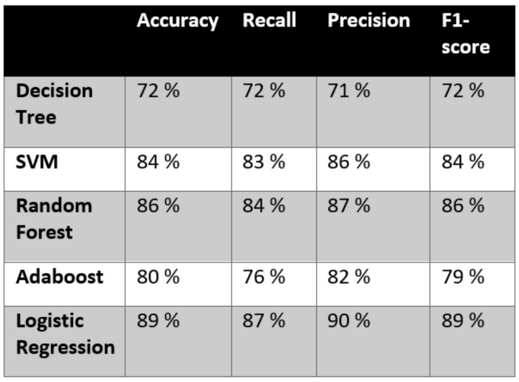
The latter represents the fraction of records that are actually positive out of all those predicted to be positive; the higher its value, the lower the number of false positives. These last two measures can conflict with each other: it is possible to build models that maximise only one of the two measures. To overcome this problem we calculate the F1-measure given by the harmonic mean between Recall and Precision. If a high value of this measure is obtained, one can be reasonably sure that Recall and Precision are approximately high. The F1-measure formula is given by:

$$F1 - measure = \frac{\mathrm{2*Precision*Recall}}{\mathrm{Precision + Recall}}$$

A further method used to evaluate classification models is the ROC curve, which shows on the y-axis, the percentage of the total number of TruePositives (TP), and on the x-axis, the percentage of false positives (FP). The AUC represents the value of the area under the ROC curve and is a good performance measure because it allows us to define the performance of the classifier: higher values correspond to a better model.

4.4 Analysis of results

The results obtained by training the various models were quite satisfactory; below are the results obtained by applying only a whitespace tokenization, taking into account both uni-grams and bi-grams.



	Accuracy	Recall	Precision	F1-score
Decision Tree	72 %	72 %	71 %	72 %
SVM	84 %	83 %	86 %	84 %
Random Forest	86 %	84 %	87 %	86 %
Adaboost	80 %	76 %	82 %	79 %
Logistic Regression	89 %	87 %	90 %	89 %

Figure 5: Classification measures

From 'Figure 5' we notice how the model that guarantees us the best results is the logistic regression; although in the table there is a complete evaluation report of the classification results, the value of interest, since we are facing a problem of balanced classes, is the accuracy. The logistic regression gives us an accuracy value of 89%, followed by the random forest (86%), then we have the SVM and Adaboost, and finally the Decision Tree algorithm with 72%.

The algorithms were then trained on both lemmatised and stemmed (Lancaster) data; the values obtained were not significantly different from those obtained in 'Figure 5', so it was decided not to report them.

In order to try and improve performance, we tried to train the models on pre-processed data with both nltk tokenization and punct based tokenization, and in particular we modified the stemming technique, deciding to use Porter-stemming.

It was decided not to report the results because also in this case they were found to be very similar to those obtained in 'Figure 5'. Even modifying the parameters of the TfidfVectorizer method, considering for example only the part of the Term frequencies(Tf), no significantly different results were obtained from those already obtained.

In order to provide a visual and immediately interpretable comparison of the models considered below in 'Figure 6' we provide the graph of the ROC curve.

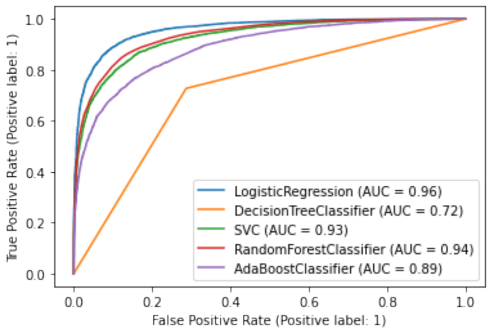
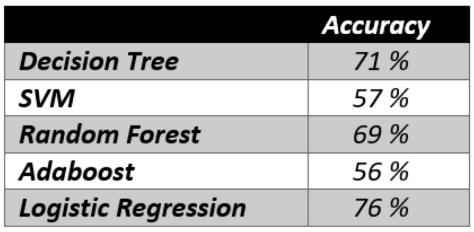


Figure 6: Roc curve unigram and bi-gram

'Figure 6' shows that logistic regression is by far the best model. In order to try to obtain better results, we tried to modify the ngram_range parameter of the TfidfVectorizer method: in particular, we tried to use only tri-grams.

A substantial worsening of the performance was noted: in 'Figure 7' below we report only the accuracy values obtained.



	Accuracy
Decision Tree	*71 %*
SVM	*57 %*
Random Forest	*69 %*
Adaboost	*56 %*
Logistic Regression	*76 %*

Figure 7: Tri-gram accuracy

We note that the Decision Tree does not change its value much and that, once again, logistic regression remains the best model.

Below, in 'Figure 8' we show the ROC curve.

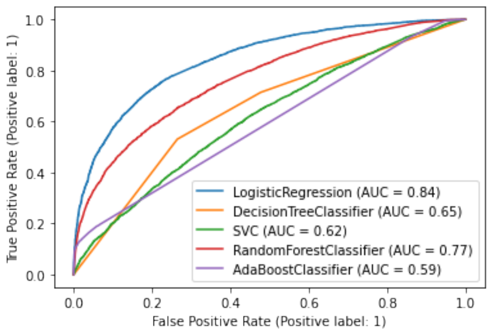


Figure 8: Roc curve tri-gram

4.5 Conclusions

We are quite satisfied with the results obtained, with very high accuracy values for almost all the models trained. In particular, the logistic regression was the best performing model despite the variation in data preprocessing techniques.

In general, it should be added that all models proved to be extremely robust to variations in preprocessing techniques. In order to further improve the performance of the models, we believe that implementing a less standardised preprocessing, therefore deriving from a more in-depth analysis of the available data, could be an alternative to be considered in order to obtain higher accuracy values.

5 Clustering

One of the tasks selected for the dataset is clustering. Clustering consists of a process of unsupervised learning, aimed to find groups with similar features in the instances of the dataset (in this case the documents).

In this case, it is going to be implemented hard clustering, so the groups will be non-overlapping. Clustering exploits usually iterative algorithms based on the concept of distance, a measure of similarity between instances.

The aim is to create group in which the documents within the group are as similar as possible and the difference between difference clusters as remarkable as it can be.

In the following process are described two different kind of clustering: the K-Means clustering algorithm and the Agglomerative Clustering. The process has been applied on stemmed dataset.

Moreover, has been implemented tf-idf BoW representation removing the most common and less common words as a proxy of the Zipf's curve (the words present in more than 97% of document or less than 3% are removed).

5.1 Clustering Visualization

The K-Means algorithm needs as input the number of clusters; for this reason has been implemented a special transformation on the data to achieve a dimensionality reduction.

In fact, while the visualization could help in defining the adequate number of clusters, the numerous dimensions of the data deny the possibility of a simple graphic representation.

To reach a two-dimensional or three-dimensional representation are applied data transformation such as Truncated SVD and TSNE.

The Truncated SVD is a transformer that performs linear dimensionality reduction by means of truncated singular value decomposition (SVD).

Since this estimator does not center the data before computing the singular value decomposition it works efficiently with sparse matrices such as the tf-idf WoB representation.

t-SNE (t-distributed Stochastic Neighbor Embedding) is a tool to visualize high-dimensional data.

It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data.

Have been chosen to reduce the dimensions to 2 and 3 to identify said groups.

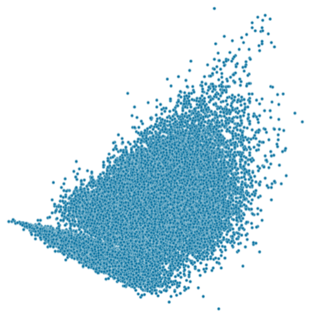


Figure 9: Two-dimensional representation after Truncated SVD transformation

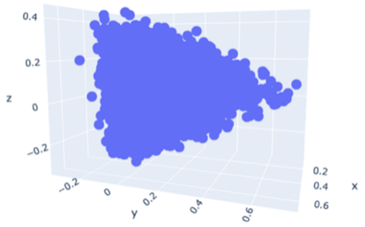


Figure 10: Three-dimensional representation after Truncated SVD transformation

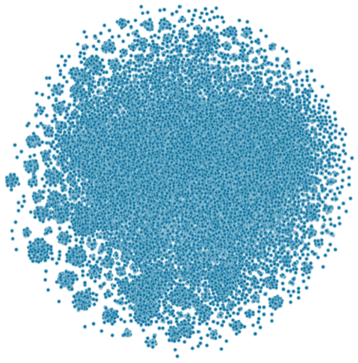


Figure 11: Two-dimensional representation after t-SNE transformation

5.2 Number of clusters determination

As shown in the graphs for both the transformers implemented the reduction did not offer a clear pattern to identify a defined number of clusters. The reason is probably the numerosity of the documents that makes it hard to distinguish edges for the clusters if they exist.

In 'Figure 11' some edges are visible but the whole plot is confused, and the number of clusters seems too high to be considered (if the number of clusters is too elevated usually there is no gain in the information). Nevertheless, a slight pattern can be seen between the lower part and rest of the plot in 'Figure 9'. So has been chosen 2 as number of clusters. To achieve further confirmation about the number of clusters has been decided to implement a grid method to find the best choice.

For this purpose, has been used sklearn function KElbowVisualizer: this function tries all the number of clusters within a given range with a defined clustering method and computes a metric to define which is the best choice. In this case has been used the K-Means algorithm and the silhouette score to find the best number of clusters between 2 and 10. The results are shown in 'Figure 12'.

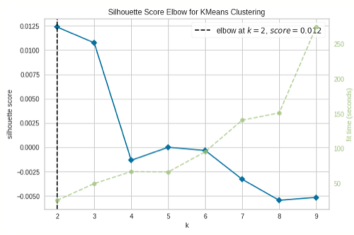


Figure 12: sklearn.KElbowVisualizer output

As shown, on the x-axis there is the number of clusters while on the y-axis there is the silhouette score. The plot confirms 2 as the adequate number of clusters whether the silhouette score seems low. The number happens to coincide with the original binary labels for the reviews (positive or negative).

5.3 K-Means clustering

The K-Means tries to find K not-overlapping subgroups (the so-called clusters) to split the observations in. K must be defined as input variable. The goal is to put in the same cluster the time-series more like each other and more different from the ones in other clusters.

The concept of similarity is computed calculating the Euclidean distance between the observations.

The first step of the algorithm is to randomly select the centroids for the clusters, then it operates in an iterative way: the algorithm calculates the distance between observations and centroids and assigns them to the nearer centroid cluster; the last step of the iteration is to calculate the mean of all data in the same cluster to compute the new centroid. The iteration is repeated until the stop condition is met (usually if no change in the centroid occurs or the algorithm ran the predefined number of iterations).

Below are shown graphically the results of the K-Means clustering algorithm on the dataset compared to the actual labels.

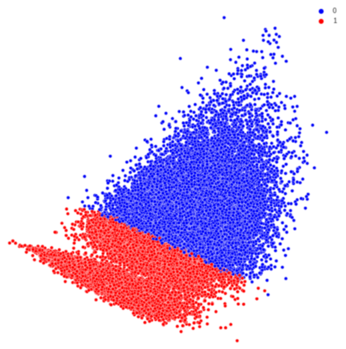


Figure 13: K-Means clustering labels as color for the instances

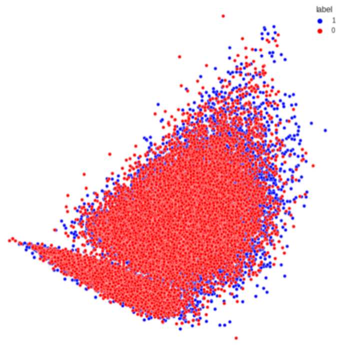


Figure 14: Actual labels as color for the instances

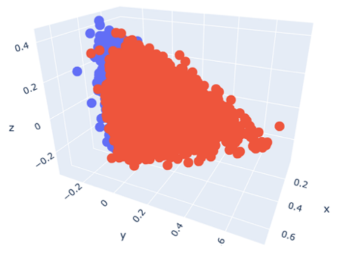


Figure 15: Three-dimensional representation after TruncatedSVD with K-Means clustering labels as color

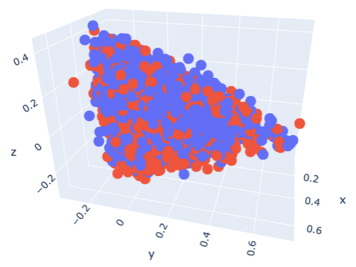


Figure 16: Three-dimensional representation after TruncatedSVD with actual labels as color

With a first impact consideration it seems like the K-Means clustering is working on a different criterion rather than the positiveness of the review, it is finding some other common feature. The result will be further analyzed later.

5.4 Agglomerative Clustering

The Agglomerative Clustering performs a hierarchical clustering using a bottom-up approach: each observation starts in its own cluster, and clusters are successively merged. The merging criteria is based on the metric used to compute the linkage, in this case has been selected the Euclidean Distance so that the merging optimizes the minimization of that measure. Below are shown graphically the results of the Agglomerative clustering algorithm on the dataset compared to the actual labels. Since the method is less scalable than K-Means for the number of instances has been used only 35.000 observations instead of 50.000.

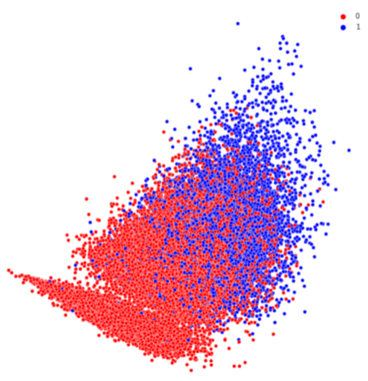


Figure 17: Agglomerative clustering labels as color for the instances of the subset

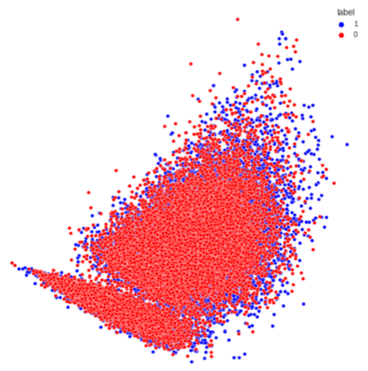


Figure 18: Actual labels as color for the instances of the subset

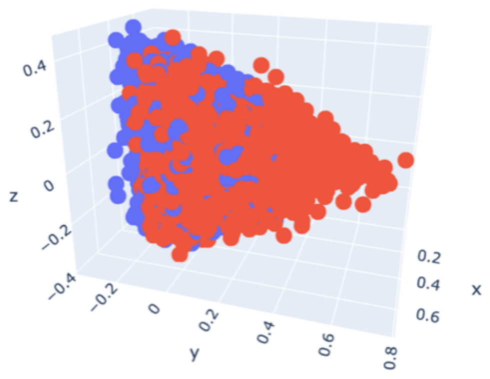


Figure 19: Three-dimensional representation after TruncatedSVD with K-Means clustering labels as color

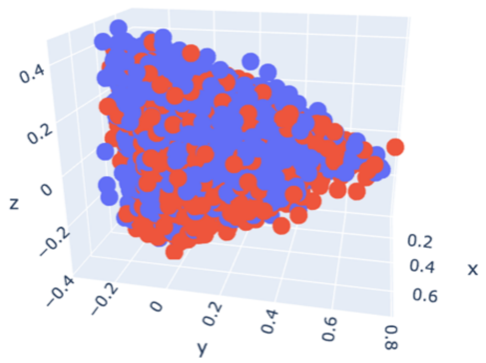


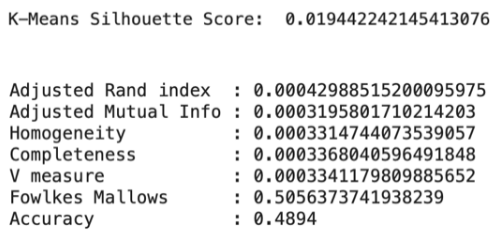
Figure 20: Three-dimensional representation after TruncatedSVD with actual labels as color

With a first consideration it seems like this kind of clustering worked better than the k-Means, but still many observation are incoherent with the original labels so the hypothesis of a different clustering criterion appears to be confirmed.

5.5 Evaluation

Since the number of clusters coincided with the label classes is possible to use both unsupervised learning metrics (such as the silhouette score) and supervised learning metrics obtained by comparing the predicted labels with the true labels.

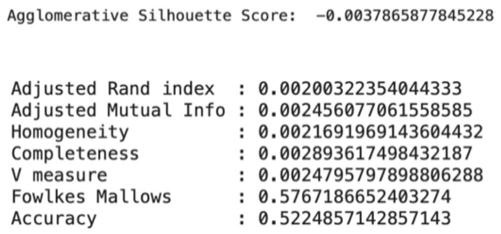
Below are shown the metrics for both kind of clustering.



```
K-Means Silhouette Score:  0.019442242145413076


Adjusted Rand index  : 0.00042988515200095975
Adjusted Mutual Info : 0.0003195801710214203
Homogeneity          : 0.0003314744073539057
Completeness         : 0.0003368040596491848
V measure            : 0.0003341179809885652
Fowlkes Mallows      : 0.5056373741938239
Accuracy             : 0.4894
```

Figure 21: K-Means performance



```
Agglomerative Silhouette Score:  -0.0037865877845228


Adjusted Rand index  : 0.00200322354044333
Adjusted Mutual Info : 0.002456077061558585
Homogeneity          : 0.0021691969143604432
Completeness         : 0.002893617498432187
V measure            : 0.0024795797898806288
Fowlkes Mallows      : 0.5767186652403274
Accuracy             : 0.5224857142857143
```

Figure 22: Agglomerative Clustering performance

As shown, the results between the two types of clustering are similar. The Silhouette Score around 0 highlights how the clustering has not been particularly effective, this value, in fact, means that the clusters are not meaningful since they are not very distinct and separated.

All of the measures resemble a random clustering: from the point of view of the original labels there is no evidence that they were relevant for the clustering purpose, because the accuracy and the Fowlkes-Mallows around 50% are theoretically obtained with random binary classification with balanced classes.

"MR COSTNER DRAGGED MOVIE FAR LONGER NECESSARY ASIDE TERRIFIC SEA RESCUE SEQUENCES CARE CHARACTERS US GHOSTS CLOSET COSTNER S CHARACTER REALIZED EARLY ON FORGOTTEN MUCH LATER TIME CARE CHARACTER REALLY CARE COCKY OVERCONFIDENT. . . "

Text 1. Instance 1; original label: 1; clustered label: 1

"FIRST HATE MORONIC RAPPERS COULD NT ACT GUN PRESSED FOREHEADS CURSE SHOOT ACTING LIKE CLICHÉ E VERSION GANGSTERS BR BR THE MOVIE TAKE FIVE MINUTES EXPLAIN GOING WE RE ALREADY WAREHOUSE SINGLE SYMPATHETIC CHARACTER MOVIE EXCEPT HOMELESS GUY ALSO ONE HALF BRAIN . . . "

Text 2. Instance 3; original label: 1; clustered label: 0

"HILLARIOUS FUNNY BROOKS MOVIE EVER SEEN WATCH RE WATCH TAPE TIMES LAUGH A CRY MOMENTS REALLY GOOD FUNNY MOVIE LIKE BROOKS MUST SHORT BROOKS BILLIONARE GETS STREETS HOMELESS DAYS ORDER WIN ENTIRE POOR DISTRICT COMPETITOR REALITY. . . "

Text 3. Instance 49990; original label: 0; clustered label: 1

As shown in these examples is difficult to understand the common criteria that conceptually guided the clustering other than mere mathematical computation.

5.6 Conclusion

The clustering did not achieve a satisfying result. The metrics were bad for both the clustering methods while the agglomerative learning reached slightly better performance but still not acceptable.

The reason for the bad results could be different: the sparsity in the data, the distribution that makes it hard to find defined and effective clusters, the still consistent percentage of words without semantic usefulness or the metonymies and synonyms.

These are just some of the possible reasons, on the other side the clustering task implemented on short numerous texts is not a simple task, as highlighted from the results.

Some possible solutions could be a different kind of preprocessing (for example implementing a rigorous cut based on Zipf's law), the implementation of NER (has been noted a relevant presence of actors name in the texts or a different representation, while the used vectorization is one of the most common and efficient.

References

[1] IBDM

https://www.imdb.com

[2] Dataset

http://ai.stanford.edu/~amaas/data/sentiment/

[3] t-SNE

https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[4] K-means clustering

https://towardsdatascience.com/understanding-k-means-clustering