Università di Milano - Bicocca
Dipartimento di Informatica

Text Mining and Search

Borruso William Joseph
Galli Luca
Suriani Eugenia

Academic Year 2023-2024

Contents

1 Text Pre-Processing and Representation **4**
1.1 Text Pre-Processing 4
1.2 Data Exploration 5
1.3 Text Representation 6

2 Text Classification **8**
2.1 Models 8
2.2 Performance 10
2.2.1 TF-IDF 10
2.2.2 Bag of Words 13
2.2.3 TF-IDF Bigram 15
2.2.4 Bag of Words Bigram 18
2.2.5 Binary Vectorizer 20
2.2.6 Word2Vec 22
2.3 Evaluation 24

3 Text Clustering **26**
3.1 K-means 26
3.1.1 Bag of Words 27
3.1.2 TF-IDF 28
3.1.3 Word2Vec 28
3.2 DBSCAN 30

4 Topic Modeling **34**

Introduction

For our project we decided to work on the *IMDB reviews* [1] [1] dataset to carry out text mining techniques.

Among the many techniques that could be carried out, we chose to perform text classification, text clustering and topic modeling, after having performed necessary text pre-processing and text representation tasks.

At a general level, text classification is the process of categorizing or assigning predefined labels or categories to pieces of text based on their content and context. In our case, our type of classification is called a binary classification, as each entry in our dataset belongs only to one of two sets which indicate the sentiment of the review (*Negative* or *Positive*).

Text clustering is a technique in text mining that groups similar pieces of text together into clusters or segments based on their inherent similarities in content, themes, or patterns.

Topic modeling is another unsupervised technique we used to discover latent topics that describe the collection of reviews, where a topic represents a group of words that frequently occur together.

The dataset is composed by a total of 50000 user reviews coming from the IMDB platform, pre-split into training and testing sets (25000 each). Furthermore, train and test are also split in half to represent negative and positive reviews (12500 each).

The original data presented only the text of the review and the sentiment. During the import phase we enriched the information for each review, in case we would need it later on: from the file name we extracted the rating, and from a separate file, contained in the same folder as the data, we obtained the urls of the reviews, and from there extracted the film or tv series IDs.

[1] https://ai.stanford.edu/ amaas/data/sentiment/

Chapter 1

Text Pre-Processing and Representation

1.1 Text Pre-Processing

After having imported the training and test data, before performing any other of the intended text mining tasks, a phase of text pre-processing was performed.

Text pre-processing refers to the series of tasks and techniques applied to raw text data before analysis. It involves cleaning, formatting, and transforming the text to make it suitable for natural language processing (NLP) tasks. Text preprocessing aims to enhance the quality of the text data for effective analysis and extraction of meaningful information.

In our case, we performed the following pre-processing tasks:

- lower cased the text;
- removal of URLs;
- removal of html tags;
- removal of numbers;
- removal of punctuation;
- handling of character repetition;

- removal of english *stopwords*;
- removal of extra whitespaces;
- *tokenization*: the process of breaking text into smaller units, words in this case, for analysis and processing in natural language processing (NLP) tasks;
- *lemmatization*: the process of reducing words to their base or dictionary form, known as a lemma, to normalize variations while preserving the linguistic meaning, context and correct part of speech.

In the end we had two versions of the pre-processed datasets, one of which omitting the lemmatization task, which we used for distributed text representation, performed in the following stages of the project.

1.2 Data Exploration

From first exploration of the data, we compared the most common words present in the train and test set:

Train		**Test**	
Word	*Count*	*Word*	*Count*
movie	51706	movie	51564
film	47046	film	46402
time	15963	time	15503
character	14178	character	14178
story	13173	story	12102

As we can see, the exact same words are the most common in both train and test with similar frequencies. All the words are ones we would expect when dealing with a pre-processed movie review dataset.

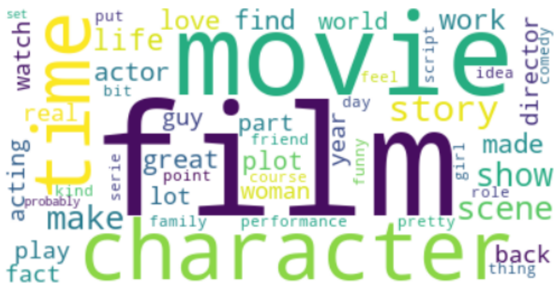


Figure 1.1: Train set wordcloud

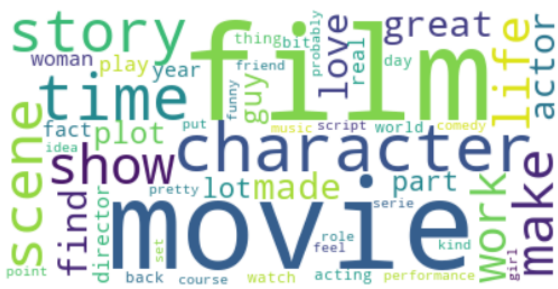


Figure 1.2: Test set wordcloud

Further information consisted in:

	Train	Test
Number of tokens	2322441	2267122
Dictionary size	65216	64238
Average review length	92.90 words	90.68 words

Again, statistics when comparing train and test are very similar. This suggests that a good train-test split was performed.

1.3 Text Representation

Text representation refers to the transformation of raw text data into a structured format, such as numerical vectors or matrices, that can be understood and processed by machine learning algorithms. For our purpose we approached this task with various methods:

- *Binary Vectorizer:* it converts text data into binary vectors, indicating the presence or absence of a term in a document, disregarding term frequency;
- *Count Vectorizer (Bag of Words):* it represents text data as frequency-based vectors, where each feature represents the count of a term in a document, ignoring the order of words;
- *Count Vectorizer Bigram:* similar to Count Vectorizer, but it considers pairs of consecutive words (bigrams) as features together with individual words;
- *TF-IDF (Term Frequency-Inverse Document Frequency):* it's a technique that weighs the importance of terms in a document relative to a collection of documents. It balances term frequency (how often a term occurs in a document) with inverse document frequency (how unique or common a term is across documents);
- *TF-IDF Bigram:* extends TF-IDF to consider pairs of consecutive words (bigrams) as features together with individual words. It applies the TF-IDF weighting scheme to capture the importance of word sequences of length 1-2 in documents;
- *Word2Vec:* a popular technique based on neural networks that transforms words into numerical vectors, often called word embeddings. Word2Vec represents words as dense, continuous-valued vectors in a high-dimensional space, capturing semantic relationships between words based on their contextual usage in a large corpus of text. This method allows words with similar meanings or contexts to have similar vector representations.

For all methods, a maximum number of features of 5000 was considered.

These text representations were used to test the various classification algorithms in the next phase of our project.

Chapter 2

Text Classification

The first main task we perform is text classification. Text classification involves automatically categorizing or assigning predefined labels or categories to pieces of text based on their content. It's a supervised machine learning task where models are trained on a dataset containing text samples and their corresponding labels. These models learn patterns and relationships within the text data to make predictions or classify new, unseen text into predefined categories or classes.

To perform our task we adopted various models. Following is a brief description of each one.

2.1 Models

In text classification, a model refers to a computational framework or algorithm trained to recognize patterns and relationships within text data to make predictions or assign categories to new, unseen text. These models are built using machine learning techniques and are trained on labeled datasets, learning to understand the features and characteristics that distinguish one category from another. Once trained, the model uses this learned knowledge to classify or predict the categories of new, unseen text based on the patterns it has learned during the training phase.

Random Forest

Random Forest is an ensemble learning method used in text classification that combines multiple decision trees to make predictions. In the context of text classification, it works by creating a multitude of decision trees during training, where each tree is built using a random subset of the features (words or other text representations) and a subset of the training data. During classification, each decision tree "votes" on the class label for a given input text, and the final prediction is determined by aggregating the votes across all trees (by majority vote in classification tasks).

MultinomialNB

Multinomial Naive Bayes (MultinomialNB) is a probabilistic classification algorithm commonly used in text classification tasks. Specifically designed for features representing word counts or frequencies (like those in Bag-of-Words or TF-IDF representations), MultinomialNB assumes that the features are generated by a multinomial distribution. In text classification, MultinomialNB estimates the probabilities of a document belonging to each class based on the frequency of words in the document. It calculates the likelihood of observing a particular word given the class and uses Bayes' theorem to determine the probability of a document belonging to a certain class given its word features.

Linear SVC

LinearSVC (Support Vector Classifier) is a linear classification algorithm used in text classification tasks. It works by finding the optimal linear boundary (hyperplane) that best separates different classes in the feature space. In the context of text classification, LinearSVC aims to create a decision boundary that effectively separates text samples belonging to different categories based on the text features (e.g., word frequencies or TF-IDF values). It seeks to maximize the margin between classes, making it robust to outliers and well-suited for high-dimensional data such as text.

Logistic Regression

Logistic Regression is a linear classification algorithm commonly used in text classification tasks. Despite its name, it's primarily used for binary classification problems, although it can be extended to handle multi-class classification by employing techniques like one-vs-rest or multinomial approaches. In text classification, Logistic Regression models the relationship between the text features (like word frequencies or TF-IDF values) and the probability of a text sample belonging to a particular class. It calculates the likelihood of a document belonging to a class using a logistic function, which transforms the output into probabilities between 0 and 1. During training, Logistic Regression learns the weights (coefficients) for each feature, determining their influence on the classification decision. It then predicts the probability of a new text sample belonging to a certain class based on these learned weights and applies a decision threshold to make the final classification prediction.

2.2 Performance

2.2.1 TF-IDF

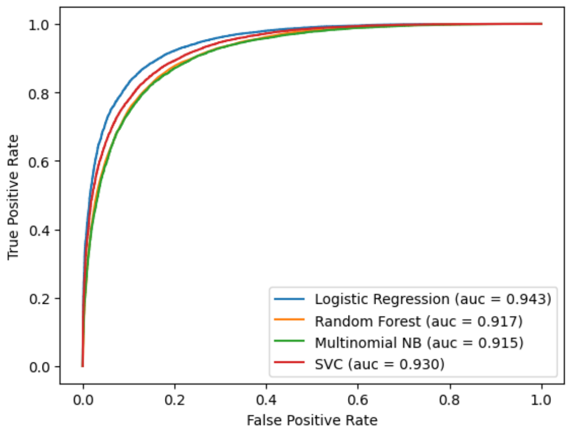


Figure 2.1: Roc Curve

Random Forest

Train Accuracy	1.00000
Test Accuracy	0.83672
Test Precision	0.84964
Test Recall	0.81824
Test F1	0.83365
Test F2	0.82433

	precision	recall	f1-score	support
Negative	0.82472	0.85520	0.83968	12500
Positive	0.84964	0.81824	0.83365	12500
accuracy			0.83672	25000
macro avg	0.83718	0.83672	0.83666	25000
weighted avg	0.83718	0.83672	0.83666	25000

MultinomialNB

Train Accuracy	0.86824
Test Accuracy	0.83604
Test Precision	0.84758
Test Recall	0.81944
Test F1	0.83327
Test F2	0.82492

	precision	recall	f1-score	support
Negative	0.82524	0.85264	0.83872	12500
Positive	0.84758	0.81944	0.83327	12500
accuracy			0.83604	25000
macro avg	0.83641	0.83604	0.83599	25000
weighted avg	0.83641	0.83604	0.83599	25000

LinearSVC

Train Accuracy	0.93812
Test Accuracy	0.84920
Test Precision	0.85621
Test Recall	0.83936
Test F1	0.84770
Test F2	0.84268

	precision	recall	f1-score	support
Negative	0.84246	0.85904	0.85067	12500
Positive	0.85621	0.83936	0.84770	12500
accuracy			0.84920	25000
macro avg	0.84934	0.84920	0.84919	25000
weighted avg	0.84934	0.84920	0.84919	25000

Logistic Regression

Train Accuracy	0.91168
Test Accuracy	0.86864
Test Precision	0.86694
Test Recall	0.87096
Test F1	0.86894
Test F2	0.87015

	precision	recall	f1-score	support
Negative	0.87036	0.86632	0.86833	12500
Positive	0.86694	0.87096	0.86894	12500
accuracy			0.86864	25000
macro avg	0.86865	0.86864	0.86864	25000
weighted avg	0.86865	0.86864	0.86864	25000

2.2.2 Bag of Words

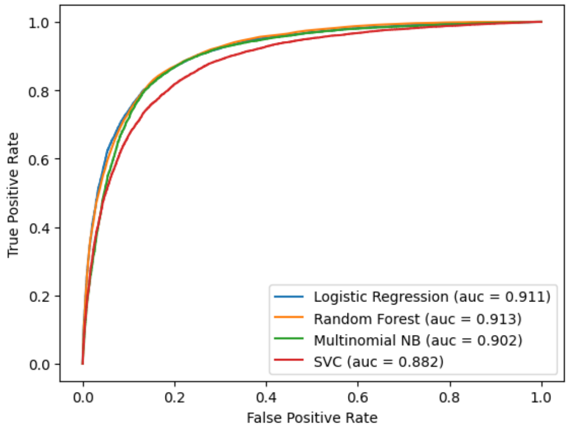


Figure 2.2: Roc Curve

Random Forest

Train Accuracy	1.00000
Test Accuracy	0.83832
Test Precision	0.84382
Test Recall	0.83032
Test F1	0.83702
Test F2	0.83299

	precision	recall	f1-score	support
Negative	0.83299	0.84632	0.83960	12500
Positive	0.84382	0.83032	0.83702	12500
accuracy			0.83832	25000
macro avg	0.83841	0.83832	0.83831	25000
weighted avg	0.83841	0.83832	0.83831	25000

MultinomialNB

Train Accuracy	0.85804
Test Accuracy	0.83220
Test Precision	0.85128
Test Recall	0.80504
Test F1	0.82752
Test F2	0.81388

	precision	recall	f1-score	support
Negative	0.81508	0.85936	0.83664	12500
Positive	0.85128	0.80504	0.82752	12500
accuracy			0.83220	25000
macro avg	0.83318	0.83220	0.83208	25000
weighted avg	0.83318	0.83220	0.83208	25000

LinearSVC

Train Accuracy	0.96416
Test Accuracy	0.80716
Test Precision	0.81908
Test Recall	0.78848
Test F1	0.80349
Test F2	0.79442

	precision	recall	f1-score	support
Negative	0.79610	0.82584	0.81070	12500
Positive	0.81908	0.78848	0.80349	12500
accuracy			0.80716	25000
macro avg	0.80759	0.80716	0.80709	25000
weighted avg	0.80759	0.80716	0.80709	25000

Logistic Regression

Train Accuracy	0.95156
Test Accuracy	0.83444
Test Precision	0.84461
Test Recall	0.81968
Test F1	0.83196
Test F2	0.82455

	precision	recall	f1-score	support
Negative	0.82485	0.84920	0.83685	12500
Positive	0.84461	0.81968	0.83196	12500
accuracy			0.83444	25000
macro avg	0.83473	0.83444	0.83440	25000
weighted avg	0.83473	0.83444	0.83440	25000

2.2.3 TF-IDF Bigram

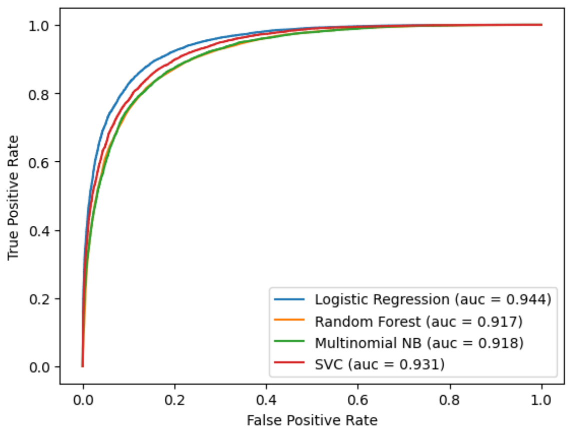


Figure 2.3: Roc Curve

Random Forest

Train Accuracy	1.00000
Test Accuracy	0.83888
Test Precision	0.84773
Test Recall	0.82616
Test F1	0.83680
Test F2	0.83038

	precision	recall	f1-score	support
Negative	0.83047	0.85160	0.84090	12500
Positive	0.84773	0.82616	0.83680	12500
accuracy			0.83888	25000
macro avg	0.83910	0.83888	0.83885	25000
weighted avg	0.83910	0.83888	0.83885	25000

MultinomialNB

Train Accuracy	0.86620
Test Accuracy	0.83856
Test Precision	0.84262
Test Recall	0.83264
Test F1	0.83760
Test F2	0.83462

	precision	recall	f1-score	support
Negative	0.83460	0.84448	0.83951	12500
Positive	0.84262	0.83264	0.83760	12500
accuracy			0.83856	25000
macro avg	0.83861	0.83856	0.83855	25000
weighted avg	0.83861	0.83856	0.83855	25000

LinearSVC

Train Accuracy	0.93940
Test Accuracy	0.85040
Test Precision	0.85610
Test Recall	0.84240
Test F1	0.84919
Test F2	0.84510

	precision	recall	f1-score	support
Negative	0.84488	0.85840	0.85159	12500
Positive	0.85610	0.84240	0.84919	12500
accuracy			0.85040	25000
macro avg	0.85049	0.85040	0.85039	25000
weighted avg	0.85049	0.85040	0.85039	25000

Logistic Regression

Train Accuracy	0.91088
Test Accuracy	0.86904
Test Precision	0.86611
Test Recall	0.87304
Test F1	0.86956
Test F2	0.87165

	precision	recall	f1-score	support
Negative	0.87202	0.86504	0.86851	12500
Positive	0.86611	0.87304	0.86956	12500
accuracy			0.86904	25000
macro avg	0.86906	0.86904	0.86904	25000
weighted avg	0.86906	0.86904	0.86904	25000

2.2.4 Bag of Words Bigram

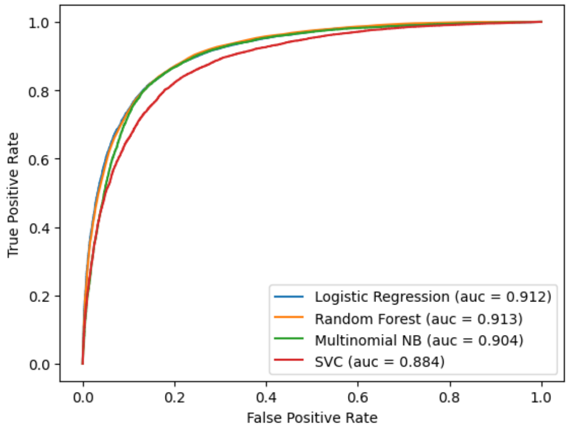


Figure 2.4: Roc Curve

Random Forest

Train Accuracy	1.00000
Test Accuracy	0.83624
Test Precision	0.84199
Test Recall	0.82784
Test F1	0.83485
Test F2	0.83063

	precision	recall	f1-score	support
Negative	0.83068	0.84464	0.83760	12500
Positive	0.84199	0.82784	0.83485	12500
accuracy			0.83624	25000
macro avg	0.83633	0.83624	0.83623	25000
weighted avg	0.83633	0.83624	0.83623	25000

MultinomialNB

Train Accuracy	0.85696
Test Accuracy	0.83508
Test Precision	0.84739
Test Recall	0.81736
Test F1	0.83210
Test F2	0.82319

	precision	recall	f1-score	support
Negative	0.82361	0.85280	0.83795	12500
Positive	0.84739	0.81736	0.83210	12500
accuracy			0.83508	25000
macro avg	0.83550	0.83508	0.83503	25000
weighted avg	0.83550	0.83508	0.83503	25000

LinearSVC

Train Accuracy	0.96208
Test Accuracy	0.80884
Test Precision	0.82061
Test Recall	0.79048
Test F1	0.80526
Test F2	0.79633

	precision	recall	f1-score	support
Negative	0.79790	0.82720	0.81229	12500
Positive	0.82061	0.79048	0.80526	12500
accuracy			0.80884	25000
macro avg	0.80926	0.80884	0.80878	25000
weighted avg	0.80926	0.80884	0.80878	25000

Logistic Regression

Train Accuracy	0.95160
Test Accuracy	0.83596
Test Precision	0.84453
Test Recall	0.82352
Test F1	0.83389
Test F2	0.82764

	precision	recall	f1-score	support
Negative	0.82780	0.84840	0.83798	12500
Positive	0.84453	0.82352	0.83389	12500
accuracy			0.83596	25000
macro avg	0.83617	0.83596	0.83593	25000
weighted avg	0.83617	0.83596	0.83593	25000

2.2.5 Binary Vectorizer

Random Forest

Train Accuracy	1.00000
Test Accuracy	0.83560
Test Precision	0.84273
Test Recall	0.82520
Test F1	0.83387
Test F2	0.82865

	precision	recall	f1-score	support
Negative	0.82876	0.84600	0.83729	12500
Positive	0.84273	0.82520	0.83387	12500
accuracy			0.83560	25000
macro avg	0.83575	0.83560	0.83558	25000
weighted avg	0.83575	0.83560	0.83558	25000

MultinomialNB

Train Accuracy	0.86400
Test Accuracy	0.84608
Test Precision	0.85459
Test Recall	0.83408
Test F1	0.84421
Test F2	0.83810

	precision	recall	f1-score	support
Negative	0.83797	0.85808	0.84791	12500
Positive	0.85459	0.83408	0.84421	12500
accuracy			0.84608	25000
macro avg	0.84628	0.84608	0.84606	25000
weighted avg	0.84628	0.84608	0.84606	25000

LinearSVC

Train Accuracy	0.95876
Test Accuracy	0.81592
Test Precision	0.82127
Test Recall	0.80760
Test F1	0.81438
Test F2	0.81030

	precision	recall	f1-score	support
Negative	0.81075	0.82424	0.81744	12500
Positive	0.82127	0.80760	0.81438	12500
accuracy			0.81592	25000
macro avg	0.81601	0.81592	0.81591	25000
weighted avg	0.81601	0.81592	0.81591	25000

Logistic Regression

Train Accuracy	0.94896
Test Accuracy	0.84356
Test Precision	0.84486
Test Recall	0.84168
Test F1	0.84327
Test F2	0.84231

	precision	recall	f1-score	support
Negative	0.84227	0.84544	0.84385	12500
Positive	0.84486	0.84168	0.84327	12500
accuracy			0.84356	25000
macro avg	0.84356	0.84356	0.84356	25000
weighted avg	0.84356	0.84356	0.84356	25000

2.2.6 Word2Vec

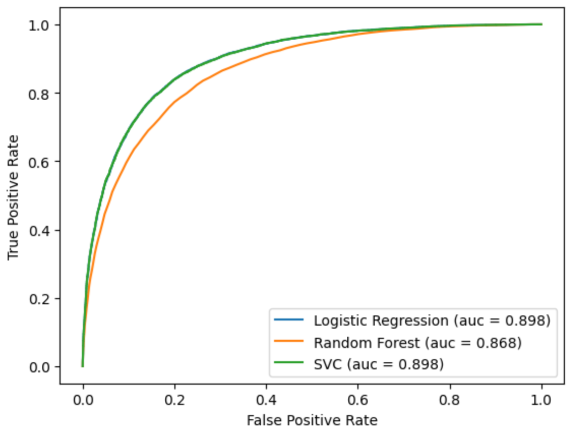


Figure 2.5: Roc Curve

Logistic Regression

Train Accuracy	0.82280
Test Accuracy	0.81856
Test Precision	0.81861
Test Recall	0.81848
Test F1	0.81855
Test F2	0.81851

	precision	recall	f1-score	support
Negative	0.81851	0.81864	0.81857	12500
Positive	0.81861	0.81848	0.81855	12500
accuracy			0.81856	25000
macro avg	0.81856	0.81856	0.81856	25000
weighted avg	0.81856	0.81856	0.81856	25000

Random Forest

Train Accuracy	1.00000
Test Accuracy	0.78580
Test Precision	0.77904
Test Recall	0.79792
Test F1	0.78837
Test F2	0.79407

	precision	recall	f1-score	support
Negative	0.79290	0.77368	0.78317	12500
Positive	0.77904	0.79792	0.78837	12500
accuracy			0.78580	25000
macro avg	0.78597	0.78580	0.78577	25000
weighted avg	0.78597	0.78580	0.78577	25000

MultinomialNB

MultinomialNB assumes that data follows a multinomial distribution, that is a generalization of the binomial distribution. Therefore we cannot apply this model to the word2vec representation since it can present also negative values.

LinearSVC

Train Accuracy	0.82348
Test Accuracy	0.81740
Test Precision	0.81641
Test Recall	0.81896
Test F1	0.81768
Test F2	0.81845

	precision	recall	f1-score	support
Negative	0.81839	0.81584	0.81711	12500
Positive	0.81641	0.81896	0.81768	12500
accuracy			0.81740	25000
macro avg	0.81740	0.81740	0.81740	25000
weighted avg	0.81740	0.81740	0.81740	25000

2.3 Evaluation

After having tested our models on all our text representations, we were able to draw some conclusions:

- The highest *train accuracy* was yielded by the Random Forest model for all text representation, with a value of 1.0. Since, however, this is a characteristic of the Random Forest, we see that the second best train accuracy was given by LinearSVC model on the Bag of Words representation, reaching an accuracy of 0.96416.

- The highest *test accuracy* was given by the Logistic Regression with the TF-IDF Bigram representation, with an accuracy of 0.86904.

- The highest *F1 score* was yielded again by the Logistic Regression with the TF-IDF Bigram representation, gaining a score of 0.86956.

Especially in text classification tasks, while all metrics are important, the F1 score tends to be a popular choice to evaluate model performances because it strikes a balance between precision (how many of the predicted positives instances are actually positive) and recall (how many of the actual positive instances were predicted correctly).

Chapter 3

Text Clustering

Text clustering is a method used in natural language processing (NLP) to group similar pieces of text together into clusters or segments based on their content, topics, or themes. This unsupervised learning technique involves analyzing textual data to identify similarities between documents, sentences, or words, aiming to organize unstructured text into meaningful groups. Text clustering doesn't rely on predefined categories but rather discovers inherent structures within the text.

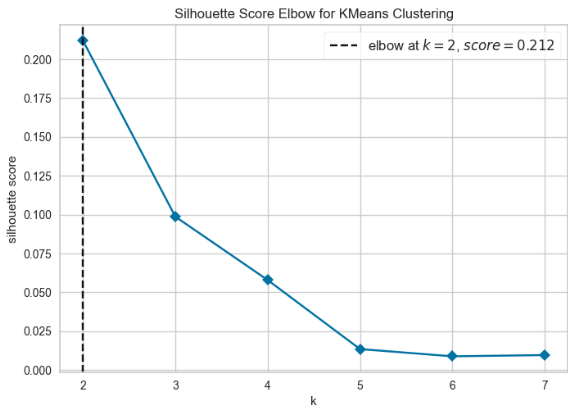
Before executing our clustering algorithms we performed singular value decomposition through TruncatedSVD. It's a dimensionality reduction technique commonly used in text processing and other high-dimensional data analysis tasks. It's particularly effective when dealing with sparse matrices, such as those generated in text processing applications like TF-IDF matrices. TruncatedSVD differs from regular SVD by keeping only a specified number of the largest singular values and their corresponding singular vectors, discarding the smaller ones. By doing so, it retains the most significant information in the data while reducing its dimensionality.

3.1 K-means

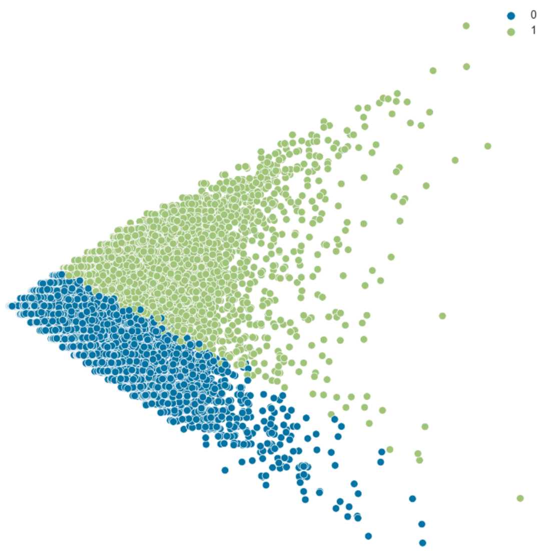
K-means is an unsupervised machine learning algorithm used for clustering similar data points into groups or clusters. K-means aims to minimize the within-cluster variance or the sum of squared distances between data points and their respective centroids. However, the algorithm's performance can be sensitive to the initial placement of centroids, and it may converge to local optima based on the initial random selection.

3.1.1 Bag of Words

The dimensionality reduction to 2000 features applied on the Bag of Words text representation yielded an explained variance of 0.8752. By plotting out the Silhouette Scores (a metric that measures the cohesion and separation of clusters, providing a value between -1 and 1, where higher values indicate better-defined and well-separated clusters) we obtained the following plot:

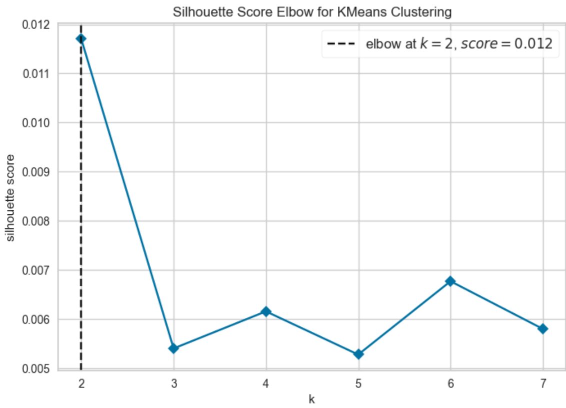


As we can see, the optimal number of cluster is the one that leads to the highest Silhouette Coefficient, in our case 2, represented as follows:

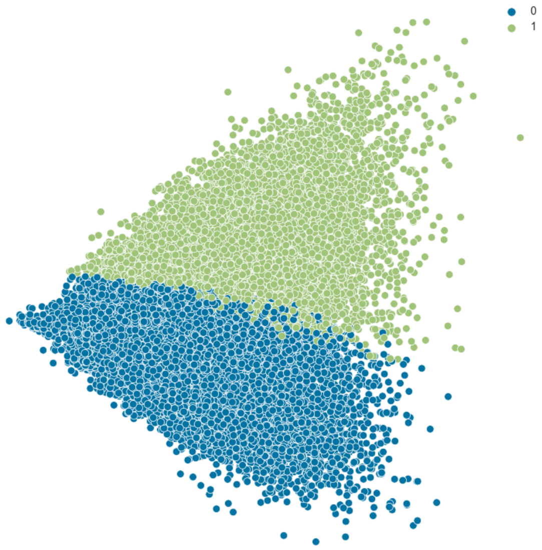


3.1.2 TF-IDF

The result of the dimensionality reduction to 2000 features on the TF-IDF representation yielded an explained variance of 0.7373, and by plotting out the Silhouette Scores we obtain the following:

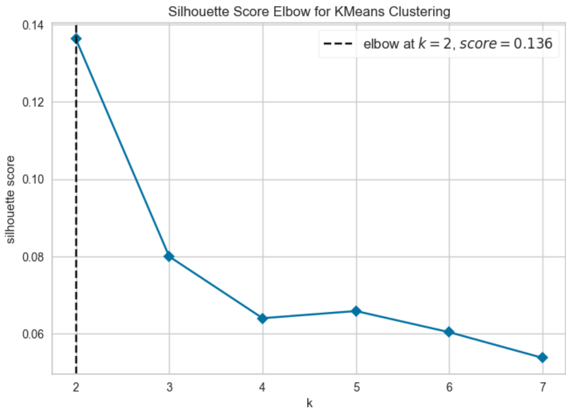


Again, the optimal number of clusters is once again 2, represented as so:

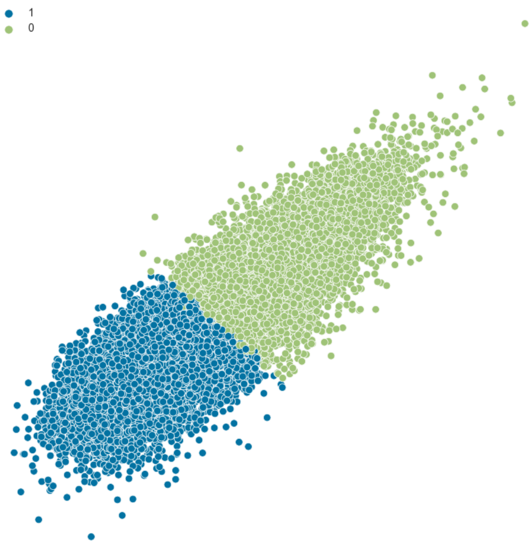


3.1.3 Word2Vec

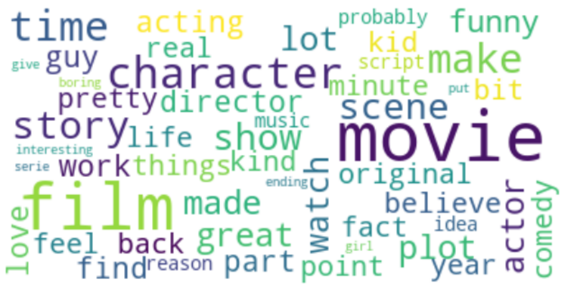
We performed clustering and for the Word2Vec representation. The Silhouette Scores came out as following:

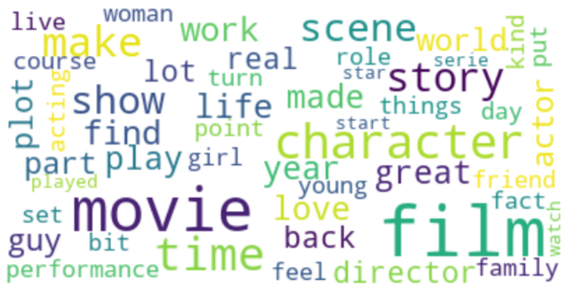


Once again, the highest Silhouette Coefficient is reached when considering 2 clusters, which for this final representation are divided as follows:



For the last clustering, we created two word clouds to represent the most common words in the two clusters:





3.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an unsupervised clustering algorithm known for its ability to discover clusters of arbitrary shapes and handle noise within datasets. Unlike K-means, DBSCAN doesn't require specifying the number of clusters beforehand and can identify clusters of varying shapes and sizes. DBSCAN's ability to handle noise and identify clusters of varying shapes and sizes makes it robust in many scenarios. However, setting appropriate values for the parameters *epsilon* and *minimum number of samples* can be critical for its performance, and it might struggle with datasets of varying densities or high-dimensional data due to the "curse of dimensionality".

Due to computational limitations, we performed DBSCAN clustering on a TF-IDF representation reduced from 5000 features to 200 features: this allows us to optimize clustering by tweaking the parameters, giving up on a big part of the explained variance.

As stated in literature, the parameter *minPts*, which is the minimum number of samples required to form a cluster, should be mainly chosen through domain knowledge. Given that a high number of features usually requires a high minPts value, we decided to test values of $minPts \geq 30$, since the number of reviews per single movie in our dataset doesn't go above 30: in this way, we try to avoid too specific clusters built around a single movie or movie character.

A good value for *epsilon*, which is the maximum distance two points can be from one another while still belonging to the same cluster, can be identified as the elbow of a *k-distance graph*, which plots the sorted distances between each point to the $k = minPts - 1$ nearest neighbors.

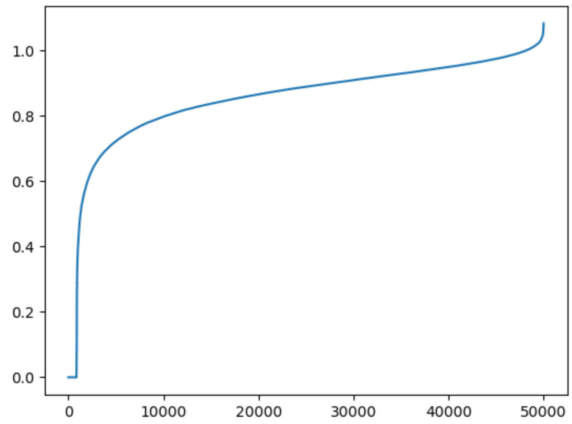


Figure 3.1: DBSCAN selection of epsilon

Despite that, computational costs limited us to testing values between $0.0 \leq \varepsilon \leq 0.5$.

Silhouette coefficient isn't a good evaluation metric when clusters can be non-convex or with irregular shapes; instead we plotted two heatmaps which show respectively, for each combination of tested values for minPts and epsilon, the average distance between noise points and their 6-Nearest Neighbors on the left, and the number of clusters on the right. We want to choose a combination which respects our assumptions, and forms a relatively low number of clusters while maximizing the average distance from the noise points.

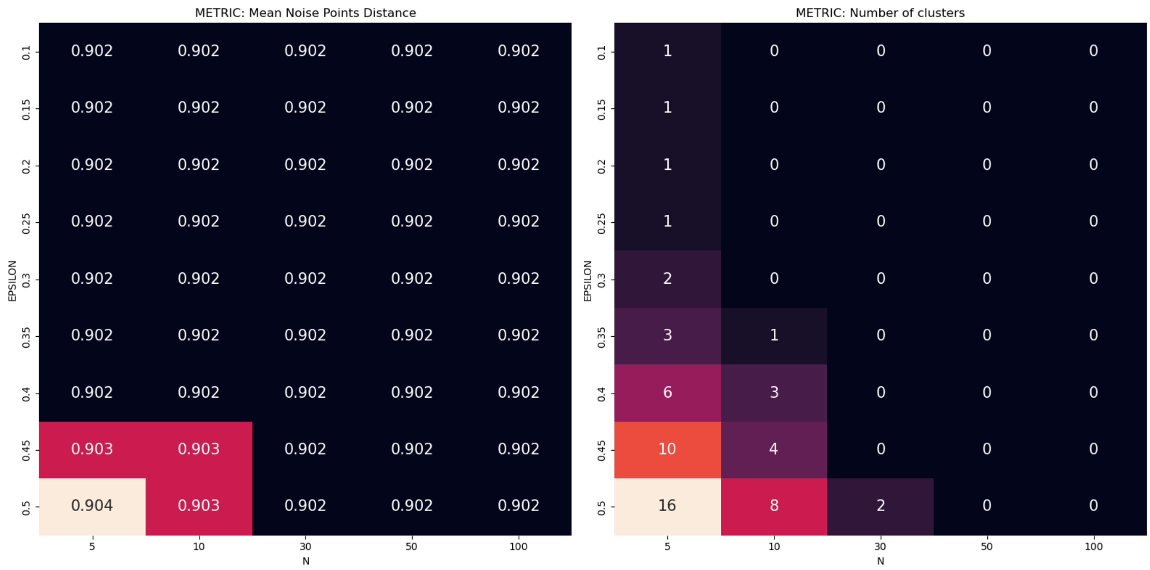


Figure 3.2: DBSCAN performance with different parameters

Based on the results, we choose the highest tested value for epsilon, which is 0.5, aware that we would probably have had a better optimization of the "Mean Noise Points Distance" with higher values of epsilon, as suggested by the plots, and way higher non-noise points.

We choose 30 as the optimal value for N, in accordance to our assumptions, since Mean Noise Points distance and the number of non-noise points are not significantly lower.

The two obtained clusters respectively contain 42 and 32 points, while all the other points are considered as noise points. Due to restricted choice of parameters, the output clusters are small and, as expected due to the low epsilon, very specific. We see that the first one is about videogames, while the second is about zombies and contains some specific character names as most frequent words.

Figure 3.3: DBSCAN clusters wordclouds

Note: with less computational restrictions, the next step would be performing DBSCAN on higher parameters values, possibly by truncating less features with SVD for a higher explained variance.

Chapter 4

Topic Modeling

Topic modeling is a statistical technique used to uncover latent thematic structures or topics within a collection of documents or text data. It's an unsupervised learning approach that aims to automatically identify common themes or topics present in the text corpus without prior labeling or supervision.

One of the most widely used algorithms for topic modeling is Latent Dirichlet Allocation (LDA). LDA assumes that each document is a mixture of various topics, and each word in the document is attributable to one of these topics. The algorithm's goal is to infer the distribution of topics in the corpus and the distribution of words in each topic, thereby revealing the underlying themes present in the text.
The algorithm is characterized by two hyperparameters: α that represents document-topic density (higher the value of α, greater the number of topics that composed documents), and β that represents topic-word density (higher the value of β, greater the number of words that composed the topic). However there also exists other parameters, that have been used in our work, that allow to select the number of topics.
In order to identify the best model, we considered different numbers of topics (from 5 to 12) and the following intrinsic evaluation metrics have been used:

- **Perplexity**, which is a statistical of how well a probability model predicts a sample. It aims to capture how "surprised" a model is of new data it has not seen before. This means that a lower value of perplexity return a better model. However, given that perplexity kept increasing together with the number of topics, and that it has been shown that perplexity and human judgment of the quality of topics are often not correlated [2], this was not used as the main evaluation metric.

- **Coherence**, that assign a score to each single topic by measuring the degree of semantic similarity between top words in the topic. These measurements help distinguish between topics that are semantically human-interpretable and topics that are the results of statistical inference. In this case, higher value of coherence are associated to better models.

The following tables show the values obtained for the mentioned metrics, computed for a number of predefined topics that goes from 5 to 12, both for the BoW and the TF-IDF text representations.

k	**Perplexity**	**Coherence**
5	1847.956	0.557
6	1839.356	0.55
7	1831.372	0.544
8	1828.594	0.544
9	1826.156	0.548
10	1817.113	0.55
11	1814.652	0.552
12	1814.387	0.558

Table 4.1: Tuning of k for BOW representation

k	**Perplexity**	**Coherence**
5	4815.237	0.568
6	5124.927	0.571
7	5445.484	0.582
8	5680.408	0.549
9	5761.23	0.529
10	5979.068	0.556
11	6193.32	0.565
12	6396.446	0.535

Table 4.2: Tuning of k for TF-IDF representation

For each text representation, we select the best value of k as the one that produces a model with high coherence and lower perplexity with respect to the other models. We selected k=12 for the BoW representation and both k=6 and k=7 for the TF-IDF representations. We notice that TF-IDF representation seems to lead to higher coherence scores but worse perplexity scores, while BoW representation shows better perplexity but lower coherence.

As a last step, keeping the optimal k values fixed, we tried to fine-tune parameters α and β by tweaking several values between 0 and 1 for both parameters. The default values remained the best for the BoW representation, while an improvement was made for the TF-IDF representation, which obtained its best result on $(K = 6, \alpha = 0.91, \beta = 0.61)$. The final results are shown in the table below.

	K	**alpha**	**beta**	**seed**	**Coherence**	**Perplexity**
BoW	12	0.08	0.08	123	0.558	1814.387
TF-IDF	6	0.91	0.61	123	0.603	4734.234

Table 4.3: capt

The final steps consists in plotting the top words representing the topics, to try to understand what the topic are about. We only show the Top 15 words below. By looking at the top words, in our case of movie reviews, topics are not easy to identify.

In the BoW model, some topics are more human-understandable, for example Topic 1 appears to be about family, Topic 4 about Sci-Fi action games, Topic 5 about animated films, Topic 7 about TV series and so on.

	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10	Topic 11
Word 0	movie	life	film	character	game	cartoon	film	show	film	movie	film	film
Word 1	time	woman	movie	film	time	character	role	episode	time	horror	performance	scene
Word 2	watch	film	music	story	movie	animation	performance	series	police	scene	murder	character
Word 3	great	love	time	movie	world	disney	actor	funny	american	make	great	time
Word 4	make	family	great	make	soldier	kid	play	tv	movie	guy	role	make
Word 5	acting	girl	dvd	life	alien	child	great	comedy	action	film	michael	plot
Word 6	character	young	year	time	effect	story	cast	time	western	acting	thriller	made
Word 7	story	child	story	scene	space	animated	love	character	black	plot	cast	acting
Word 8	actor	father	song	book	human	voice	character	season	guy	time	killer	director
Word 9	made	mother	documentary	work	action	original	story	watch	cop	effect	plot	actor
Word 10	watching	friend	made	point	film	batman	scene	joke	scene	pretty	play	script
Word 11	funny	story	version	director	scene	short	star	great	car	made	story	minute
Word 12	plot	find	video	plot	sci	make	time	make	town	budget	director	work
Word 13	love	year	watch	feel	earth	comic	version	guy	make	minute	horror	sex
Word 14	part	boy	love	real	german	time	year	year	city	zombie	time	making

Table 4.4: Top 15 topics for BoW model

In the TF-IDF model, instead, words are pretty generic, except for few words suggesting the movie genre or the users opinion (great, funny...).

	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Word 0	film	movie	movie	film	movie	movie
Word 1	movie	film	film	movie	film	film
Word 2	time	time	time	character	character	great
Word 3	story	character	story	time	scene	make
Word 4	show	scene	make	story	time	story
Word 5	character	story	show	acting	make	character
Word 6	great	watching	great	actor	great	funny
Word 7	make	watch	character	make	acting	scene
Word 8	life	work	book	year	love	show
Word 9	love	make	made	made	actor	made
Word 10	watch	plot	scene	scene	show	watch
Word 11	scene	young	year	great	life	time
Word 12	plot	life	part	world	watch	love
Word 13	acting	find	watch	family	series	actor
Word 14	actor	black	horror	lot	made	guy

Table 4.5: Top 15 topics for TF-IDF model

Bibliography

[1] Andrew L. 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, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[2] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-graber, and David Blei. Reading tea leaves: How humans interpret topic models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.