Text Mining Project

Text Classification and Text Clustering on Steam Reviews

Potertì Daniele, 844892
Sanvito Alessio, 844785

Department of Computer Science, Faculty of Data Science, Università degli Studi di Milano-Bicocca

01/2023

Abstract

This project aims to classify reviews from a Steam dataset through some text classification techniques; in particular, different BERT models have been used to compare their performance. The results performed by the models implemented are evaluated even in comparison with some other results obtained by different models implemented in papers online. This kind of approach has been used even for the second task of the project, which is text clustering, whose aim is to group reviews that express similar sentiments.

Keywords: *Text Mining, Text Classification, Text Clustering, Text Representation*

1. Introduction

Two main algorithms have been implemented in this project: text classification and text clustering.

- Text classification: The paper "Sentiment Analysis of Steam Review Datasets using Naive Bayes and Decision Tree Classifier" presents a method for analyzing the sentiment of reviews of video games on the Steam platform using traditional machine learning classifiers. However, in recent years, deep learning approaches have become increasingly popular for natural language processing tasks due to their ability to capture complex patterns in text data. Therefore, for this project, we decided to investigate the use of BERT, DistilBERT, RoBERTa, and other variants for sentiment analysis on the Steam reviews dataset. These models are pre-trained on large amounts of data and can be fine-tuned for specific tasks (such as sentiment analysis). We think that by using these deep learning approaches, we can improve the performance of the sentiment analysis task compared to the traditional classifiers used in the original paper. We will compare the results of our modified pipeline to those presented in the original paper to see if there is a significant difference in performance.
- Text clustering: using the paper "Clustering and identify games review on Steam Store" as a referral, different clustering techniques are performed to determine if the majority of reviews are Recommended or Not Recommended reviews. The research also found that the accuracy of the clustering algorithm is high with a sample data of 500, but the accuracy can vary with different sample data. In our project, after choosing an appropriate size to create a data sample (to be able to create a representation with the TF-IDF and clustering) and clustering on the data sample, various measures of clustering performance evaluation such as silhouette score, rand index were used. With this kind of approach, it is possible to underline the structure of the data and group similar documents together into clusters.

2. Dataset

This dataset contains a collection of reviews for a game on the Steam platform. Each row in the dataset represents a single review, providing various details about the review such as the number of comments on the review, the language in which the review is written, the timestamp of when the game was last played by the user who wrote the review, and the total playtime of the game by the user who wrote the review. Additionally, the dataset includes information about the user who wrote the review, such as the number of games owned by the user and the number of reviews written by the user. The dataset also contains information about how the game was obtained by the user, whether it was received for free or purchased on Steam. Furthermore, the dataset includes information about the votes received by the review, such as the number of upvotes and funny votes, as well as a score calculated based on the votes received by the review. Additionally, the dataset includes information about whether the review was written during the game's early access phase. It's worth noting that this dataset seems to be preprocessed, as there are columns such as 'cleaned' and 'num_words' which are not original in the dataset, they have been added to the dataset after some process.

3. Preprocessing

We performed several steps of text processing on a dataset of reviews to prepare the data for sentiment analysis. First, we removed special characters and digits and transformed all letters into lowercase. We then removed stop words and applied them to the stem to reduce the size of the words. Links were removed using regular expressions, and we also removed the most frequently and infrequently appearing words by setting thresholds that were determined through cross-validation. Finally, we removed short reviews that contained fewer than a certain number of words to minimize the noise in the dataset. We implemented these text processing steps as described in Zuo's paper since there was no available code on GitHub that we could use.

4. Text Representation

Text representation is the process of converting a piece of text into a numerical form that can be used as input to a machine learning algorithm. There are many different ways to represent text for machine learning, and the choice of representation can depend on the specific task, the type of text, and the type of algorithm being used.

In this project the reviews were represented in a structured form in two ways:

- Bag of words: method of representing text as a numerical feature vector; it does not take into account the order of the words in the text, but rather, it treats the text as an unordered collection, or bag, of words. In this representation, each document is represented as a vector of word counts, where each bin in the vector corresponds to a specific word in the vocabulary. The value in each bin is the number of times that word occurs in the document.
- TF-IDF: method of representing the importance of words in a document based on their frequency in the document and a collection of documents. In this representation, the term frequency (TF) of each word in the document is calculated as the number of times that word occurs in the document, and the inverse document frequency (IDF) is calculated as the logarithm of the ratio of the total number of documents in the collection to the number of documents in which the word occurs. The final TF-IDF score for each word is obtained by multiplying the TF and IDF values.

The results of the text representation were passed to an SVD to reduce their dimensionality; in particular, an optimal number of components to perform SVD using the co-occurrence matrix was calculated and the results after SVD were:

- Bag of Words obtained an explained variance ratio of 0.9789503414025237
- TF-IDF obtained an explained variance ratio of 0.6141355453052243

After applying both of them to the clustering data, the results were similar, so just the TF-IDF version was maintained.

5. Text Classification

5.1. Data Preparation

Before continuing we remove unnecessary columns from the DataFrame before performing further analysis on the remaining data. Additionally, we check for any missing values or 'nan' values in the data, which could potentially skew or compromise the results of any subsequent analysis. This step is an important part of the data preparation phase, as it ensures the quality and integrity of the data being used for the analysis.

5.2. Data Distribution and Split

We use the 'train_test_split' function from the sci-kit-learn library to randomly split the DataFrame into three new DataFrames: 'df_train', 'df_dev', and 'df_test' with respective sizes of 50000, 10000, and the remaining. The data frame 'df_test' later in the code will be sampled to be 10000.

We also use a downsampling technique to balance the class distribution of the 'df_train' and 'df_test' data frames. We create a new data frame 'df_majority_downsampled' by randomly selecting samples from the majority class to match the number of samples in the minority class. This ensures that the class distribution is balanced in the training dataset, which is important for building a good-performing model (as can be seen in Figure 1).

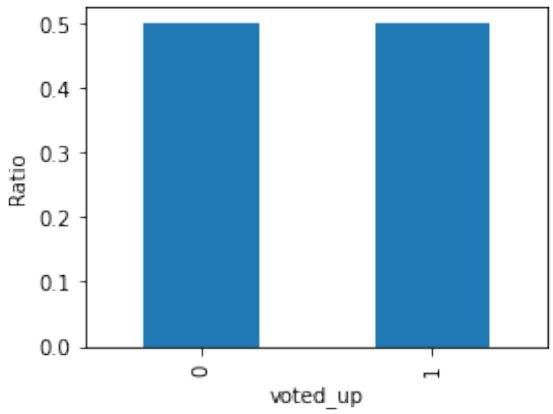

Figure 1. The values on the training dataset are balanced

Splitting data into training, evaluation, and test sets is important in machine learning because it allows for the assessment of the performance of a model on unseen data. The training set is used to train the model, the evaluation set is used to fine-tune the model's hyperparameters, and the test set is used to evaluate the performance of the final model.

5.3. Tokenization

In this phase of the process, we undertake several crucial steps in preprocessing text data for use in a language model, such as BERT. These steps include tokenizing the text, counting the number of tokens (Figure 2), and visualizing the distribution of the token count.

Additionally, we calculate the ratio of reviews that have less than 128, 256, and 384 tokens, which can be beneficial in determining the maximum input length for a language model such as BERT.

- Less than 128: 94,4%

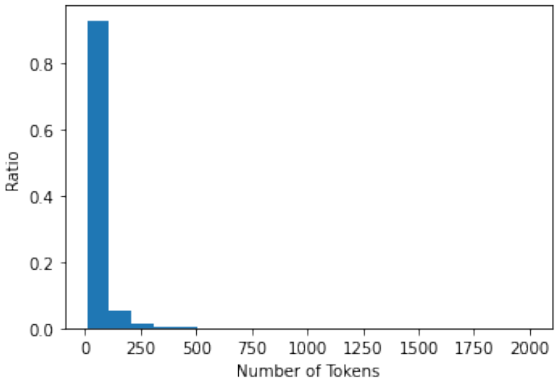

Figure 2. The number of tokens

- Less than 256: 98,7%
- Less than 384: 99,6%

As a transformer-based model, BERT has a maximum input length, and any document exceeding this length must be truncated or divided into multiple segments for proper processing by the model. By counting the number of tokens in each review, we can identify which reviews require truncation or division, thereby maximizing the capabilities of the BERT model.

This ensures that the model can analyze the entirety of the input data, leading to more accurate predictions and a better understanding of the text's meaning. Conversely, if the input data exceeds the maximum length, the model may not be able to fully analyze the input and may not produce accurate predictions or understand the text's meaning.

5.4. Experiments and results

The selection of the maximum sequence length (max_seq_length) is a crucial aspect when utilizing a language model such as BERT. This parameter defines the upper limit of text that the model can process in a single iteration. Utilizing an overly high max_seq_length can result in excessive consumption of memory resources, leading to potential instability in the python kernel. In light of this, a max_seq_length of 64 was chosen for the BERT model. This decision allowed us to effectively mitigate memory-related issues while maintaining the integrity of the model's predictions and ensuring the stability of the kernel. Later we tried with success, a max_seq_length of 256 which led us to similar results. In both cases, a train_batch_size of 16 was chosen for kernel stability reasons.

Where it is available we used the uncased version of BERT.

5.4.1. BERT

In this configuration, we used the pre-trained BERT model with the "bert-base-uncased" version. This model has been fine-tuned for a binary classification task, with two labels to predict. The model is implemented using the Hugging Face's ClassificationModel class, which allows an easy fine-tuning of the pre-trained BERT model on a given dataset.

The maximum sequence length for input data is set to 64, and the batch size for training is 32. The model will be trained for 1 epoch.

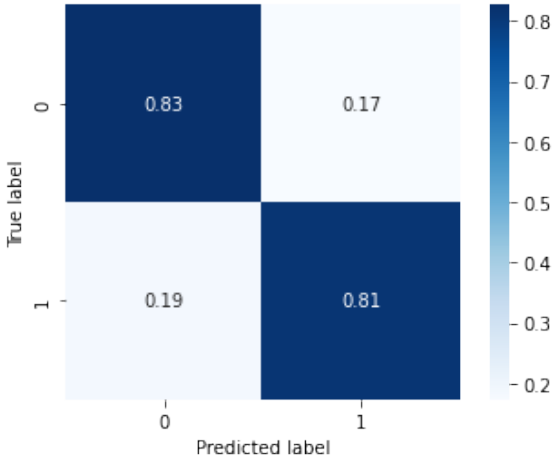

Figure 3. Accuracy of BERT

The results of the training (in Figure 3) show an overall accuracy of 0.8196 on the testing set. This means that the model correctly predicted the outcome of 81.96% of the cases. The model was trained to classify data into two categories, represented by 0 and 1 in the table above. The f1-score is a harmonic mean of precision and recall and it is often used as a measure of a model's accuracy. The f1-score for category 0 is 0.8209 and for category 1 is 0.8183. The support indicates the number of instances of each category in the testing set. The model was trained and tested on 8316 instances in total, with 4158 instances of category 0 and 4158 instances of category 1.

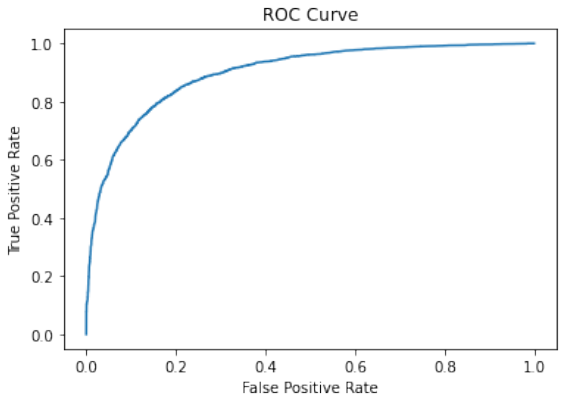

Figure 4. ROC curve of BERT

Overall, the results show that the model performs well in terms of accuracy and precision (as in Figure 4), however, there is room for improvement in terms of recall for both categories. The model can correctly predict the majority of the instances, but it does miss a portion of them.

5.4.2. BERT-LARGE

The model being used is BERT with the 'bert-large-uncased' pre-trained weights. It is a binary classification model with two labels. The model is trained with a maximum sequence length of 64, a batch size of 32, and for 1 epoch.

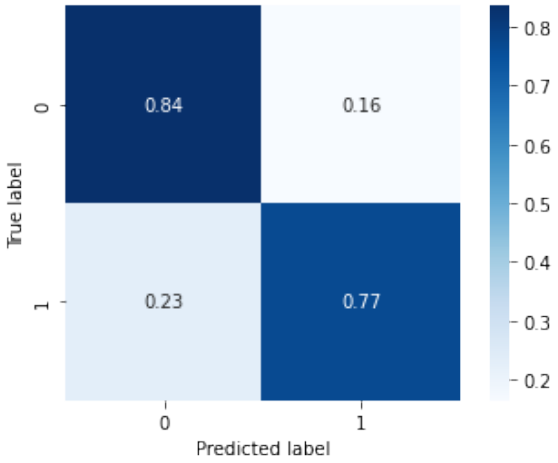

Figure 5. Accuracy of BERT-LARGE

The results (in Figure 5) show that the model has an overall accuracy of 0.80 on the testing set, which is a good performance. The precision and recall for class 0 are 0.7832 and 0.8360 respectively, while for class 1 it is 0.8241 and 0.7686 respectively. This means that the model has higher precision and recall for class 0 than for class 1. The f1-score is a measure of the balance between precision and recall, and the model has an f1-score of 0.8087 for class 0 and 0.7954 for class 1. The f1-score for class 0 is higher than for class 1. It is also worth noting that the support for each class is 4158, which indicates that the classes in the testing set are balanced. This means that the model's performance is not affected by an imbalanced class distribution. The ROC curve of this model is shown in Figure 6.

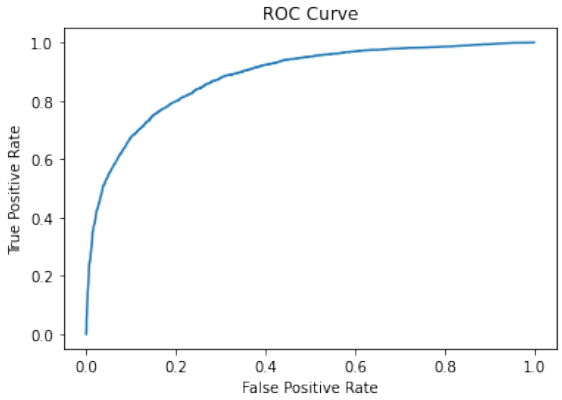

Figure 6. ROC curve of BERT-LARGE

5.4.3. DistilBERT

The model being used is BERT with the 'distilbertbaseuncased' pre-trained weights. It is a binary classification model with two labels. The model is trained with a maximum sequence length of 64, a batch size of 32, and for 1 epoch.

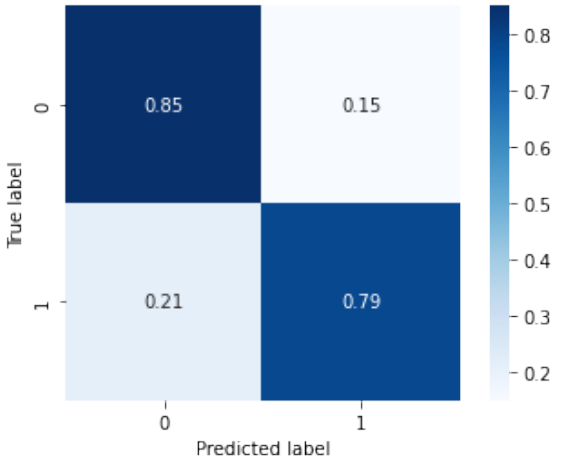

Figure 7. Accuracy of DistilBERT

The model's overall accuracy on the testing set is 0.8201, which means that the model correctly predicted the class of 82.01% of the samples in the testing set. This is a relatively high accuracy, indicating that the model is performing well on this dataset (as shown in Figure 7).

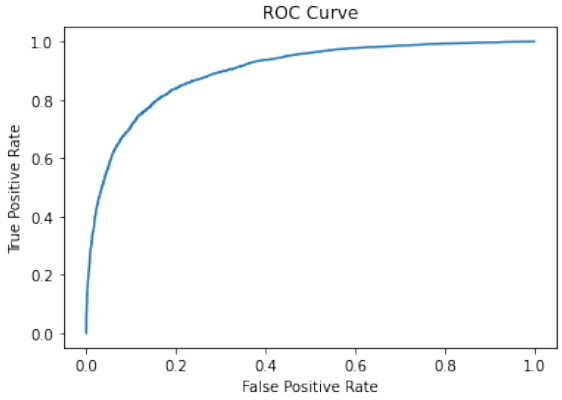

Figure 8. ROC curve of DistilBERT

The F1-score is a harmonic mean of precision and recall, and it is a commonly used metric for evaluating the performance of a binary classification model. The F1-score for class 0 and class 1 are 0.8256 and 0.8142 respectively.

In summary, the model has relatively high accuracy, precision, and recall, indicating that it performs well on the testing set. The ROC curve of this model is shown in Figure 8.

5.4.4. RoBERTa

The model used is BERT with the 'distilbertbaseuncased' pre-trained weights. It is a binary classification model with two labels.

The model's overall accuracy on the testing set is 0.8195, which means that the model correctly predicted the class of 81.95% of the samples in the testing set. This is a relatively high accuracy, indicating that the model is performing well on this dataset (as shown in Figure 9).

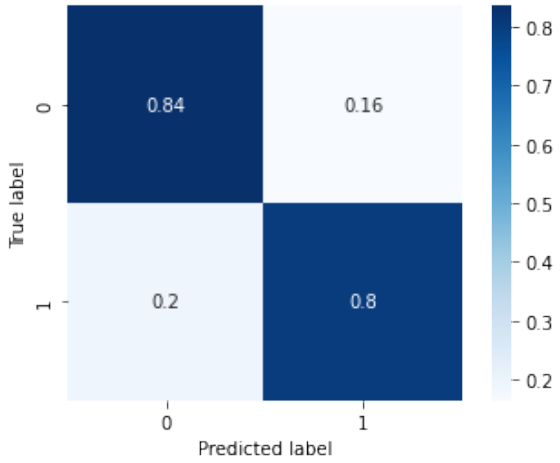

Figure 9. Accuracy of RoBERTa

The F1-score is a harmonic mean of precision and recall, and it is a commonly used metric for evaluating the performance of a binary classification model. The F1-score for class 0 and class 1 are 0.8225 and 0.8164 respectively.

In summary, the model has relatively high accuracy, precision, and recall, indicating that it performs well on the testing set (as shown in Figure 10).

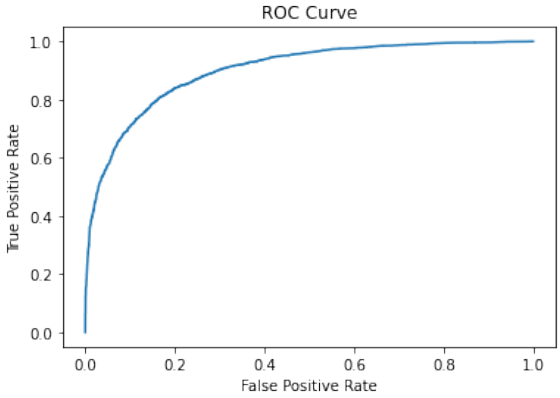

Figure 10. Accuracy of RoBERTa

6. Text Clustering

6.1. Data Preparation

In this phase of the project, the starting data were those provided as output by the preprocessing phase. Since the two classes of interest (i.e. those with the attribute voted_up = 0 and those with voted_up = 1) had different numbers (about 600,000 rows for the first, 400,000 for the second), the first operation was to balance the dataset and consider only a part of this dataset (so a balanced sample) to make all the operations necessary for the clustering sustainable (both in terms of computational and temporal aspects).

Singular value decomposition (SVD) was applied to the data before the actual clustering process to reduce dimensionality and prevent overfitting. SVD is a linear algebra technique that allows the matrix to be factorized and the amount of data to be reduced. SVD was chosen over principal component analysis (PCA) because it is more efficient with sparse matrices like in this case. By using SVD, the dimensions of the co-occurrence matrix X were reduced, resulting in more robust and computationally feasible estimates for the subsequent clustering steps. The number of components to be used was calculated using the co-occurrence matrix; in particular, an empty co-occurrence matrix has been initialized setting the co-occurrence window to 4 (that is the most used value in this kind of task). The co-occurrence matrix is calculated for each review in a DataFrame by iterating through each review and sentence; then a list of possible values for the number of components has been initialized to find the optimal value for the number of components to be retained. Then an SVD was performed with the values in the list and the value that was chosen (350) was the one that obtained a value of the variance of 0.97 (considered a good value even watching the plot of the number of components versus explained variance). Then TruncatedSVD was used to reduce the dimensions of the input data matrix; it returns the transformed data and prints the explained variance ratio of the TruncatedSVD model.

So, after performing the necessary preparation, several text clustering methods were applied.

6.2. Experiments and Results

The choice of the number of clusters is a crucial step in the application of clustering algorithms, both in flat clustering algorithms such as k-means and in general. Finding the appropriate number of clusters is a major challenge that must be addressed in order to achieve meaningful results. The optimal number of clusters was calculated using the k-elbow method. This method is a heuristic for choosing the optimal number of clusters for a k-means clustering analysis. It does this by fitting the k-means model for a range of values of k and measuring the within-cluster sum of squares (WCSS) for each value. The WCSS is a measure of the compactness of the clusters, with lower values indicating more compact clusters. In this project, a number of clusters between 2 and 11 were selected and different metrics for selecting the optimal number were tested (distortion, calinski_harabasz, silhouette that can be seen in Figure 11).

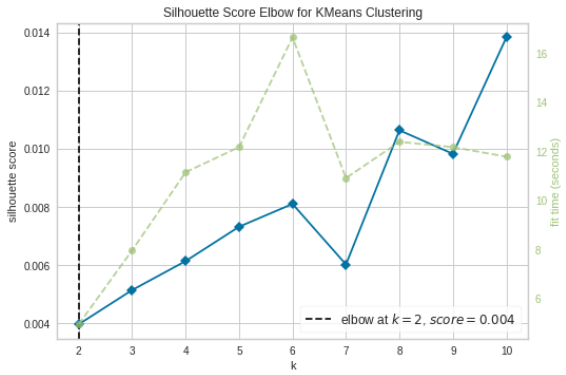

Figure 11. K-elbow method with silhouette metric

With the Silhouette metric, the recommended optimal number of clusters was 2 and even with

the Calinski Harabasz metric it was determined to be 2; so, since the dataset has just two instances of the voted_up class, a number of 2 clusters was used.

Another important factor is to use evaluation metrics to verify the quality of the clustering obtained; these metrics can be used to assess consistency within clusters, completeness between clusters, the separation between clusters and other aspects of clustering. Several clustering evaluation metrics were used, both supervised and unsupervised. The supervised metrics were used to compare the clustering result (i.e. the division into clusters made by this algorithm) with the true values of the class labels. These thus give an idea of the quality of the clusters generated with respect to a set of labels. Conversely, unsupervised clustering evaluation metrics are used to assess the quality of clusters generated by a clustering algorithm without reference to a known set of labels. These metrics provide information on the quality of the generated clusters, but do not necessarily indicate whether the generated clusters correspond to the actual classes, thus giving an idea of how compact and well-separated the generated clusters are.

To obtain a better representation of the results, these were visualized with both word clouds and scatterplots.

6.2.1. K-means

The k-means algorithm is a method for partitioning a set of text documents into a specified number of clusters (k) based on the similarity of the documents. It does this by iteratively assigning each document to the cluster with the nearest mean, recalculating the cluster means, and repeating this process until convergence.

So a clustering with 2 clusters was carried out by assigning the cluster labels to the data frame in order to find out via a contingency table which cluster was associated with which value in the voted_up column. The result is that the first cluster (cluster 0) is associated with the rows with voted_up = 0, and the second with the other rows (as shown in Table 1).

Table 1. Division in clusters for k-means

Cluster	voted_up	counts
0	False	7332
0	True	5211
1	False	2723
1	True	4734

This distinction is by no means so clear-cut and this can also be seen in the clustering results (Table 2. In fact, as far as the supervised metrics are concerned, it can be seen that the quality of clustering is good (Rand Index and Fowlkes Mallows have values above 0.5), whereas the consistency within clusters is poor (both homogeneity and completeness have a score of 0.03). The V measure is a combination of homogeneity and completeness and indicates an overall low quality of clustering. In general, it can be said that the results show a good ability to separate clusters but poor consistency within clusters.

Table 2. Metrics evaluation for k-means

Metric	score
Rand index	0.5213178458922946
Adjusted Mutual Info	0.033537300793458825
Homogeneity	0.03278162229110691
Completeness	0.03440353998606366
V measure	0.03357300377269041
Fowlkes Mallows	0.5365589969852546

Using supervised measures, the results aren't different, as these show a poor quality of clustering; an example is the Silhouette Score; this is very low (0.004), which indicates that the points within the clusters have very little similarity to each other and that the clusters are not well separated from each other.

As far as the word clouds are concerned, they succeed in highlighting certain keywords that underline the (not entirely optimal) division of the clusters.

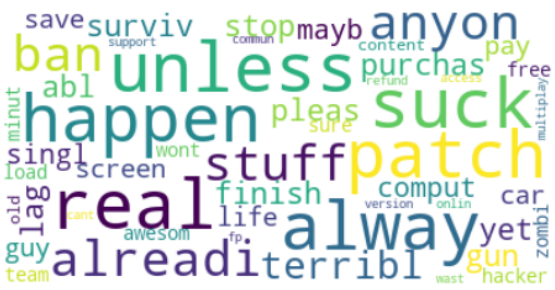

Figure 12. Cluster 0: negative values

This word cloud in Figure 12 seems accurate in that most of the terms have a negative connotation in a gaming context; examples are ban, hacker, suck, cant, lag, refund, terribl and patch. In the word cloud of cluster 1 (the one in Figure 13 that should have positive values) some of the highlighted terms have a positive mood such as fan, kind, beauti, but it is evident that there are also negative terms (such as lack).

Figure 13. Cluster 1: positive values

Finally, the scatterplot (Figure 14) shows clusters quite separate at the extremes, but overlapping in the middle, indicating that a clear division between the two clusters was not possible.

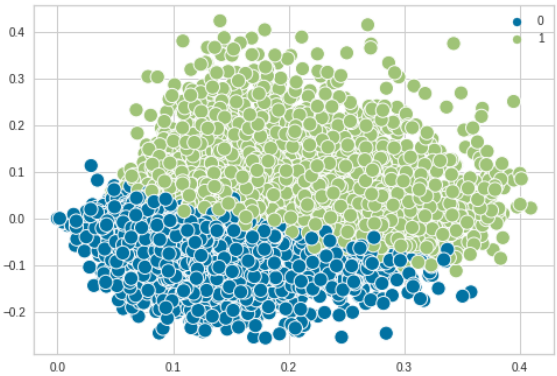

Figure 14. Scatterplot k-means

6.2.2. Hierarchical

Hierarchical clustering is a method for partitioning a set of text documents into clusters based on their similarity. It does this by creating a tree-like structure called a dendrogram that encodes the relationships between the documents and their similarity to one another.

In the context of text clustering, hierarchical clustering can be a useful method for partitioning documents into clusters based on their similarity. It is able to identify clusters of varying densities and shapes, and it does not require the user to specify the number of clusters in advance. However, it can be slower than some other clustering algorithms and may not always produce the most intuitive or meaningful clusters.

In this approach, 2 is the number of clusters used; moreover, different affinity measures (useful for specifying the similarity between the objects that we want to group) and different linkage techniques (which allow specifying how to merge two clusters to form a new cluster) have been experimented with. This was done to explore different options and evaluate which one provides the best results for our use case.

The results obtained by combining these different parameters were almost identical to each other and very similar to those obtained with k-means. For the sake of completeness, only the results of the algorithm that obtained the best results (i.e. Hierarchical Clustering with Euclidean Affinity and Ward Linkage) are reported in Table 3.

Table 3. Metrics evaluations for Hierarchical

Metric	score
Rand index	0.5000322466123306
Adjusted Mutual Info	0.0011441142354577253
Homogeneity	0.0006466529592096146
Completeness	0.009619517834688679
V measure	0.0012118422338479409
Fowlkes Mallows	0.7014953270898591

The Hierarchical Clustering results show average classification quality according to the Rand index and Fowlkes Mallows but poor consistency within clusters according to homogeneity and completeness. The V measure is a combination of homogeneity and completeness and indicates an overall low quality of clustering. The Adjusted Mutual Info value is very low, which indicates that the clustering is not very meaningful with respect to the random distribution of the data. Overall, it can be said that the results show a medium quality of clustering from the point of view of both consistency within clusters and overall classification.

This consideration can also be extended to the results obtained with the unsupervised metrics and is also evident when evaluating the clustering of reviews; in particular, all hierarchical algorithms (and also in Birch) show a strong aggregation of points (both positive and negative) within cluster 0 (as shown in Table 4).

Table 4. Division in clusters for Hierarchical

Cluster	voted_up	counts
0	False	10001
0	True	9839
1	False	54
1	True	106

This could be a sign that the clustering was not able to capture the diversity between the points.

As for word clouds, again, they do not seem to be fully explanatory and fail to separate words with negative from those with positive meanings (Figures 15 and 16).

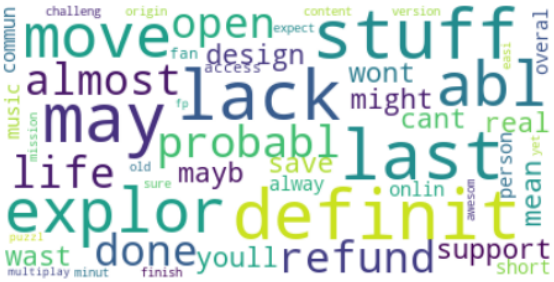

Figure 15. Cluster 0: negative values

Figure 16. Cluster 1: positive values

The scatterplot in Figure 17 shows a very high density of points belonging to cluster 0 and a minority belonging to cluster 1. Thus, this clustering technique produces unbalanced results.

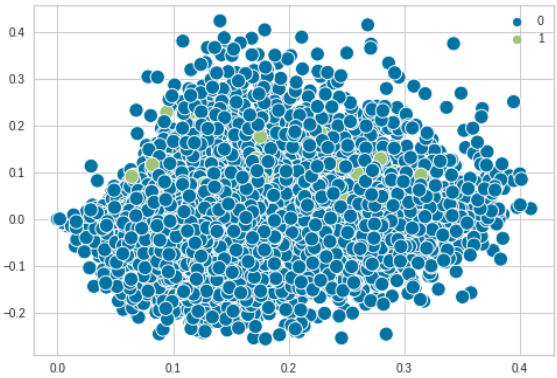

Figure 17. Scatterplot Hierarchical

6.2.3. Birch

In the context of text clustering, Birch can partition a set of documents into clusters based on their similarity. It requires the user to specify the number of clusters to be created and can be faster than some other clustering algorithms, but it may be sensitive to the initial choice of cluster means and may not always produce the most intuitive or meaningful clusters. Other clustering algorithms, such as k-means or DBSCAN, may be more appropriate in some cases.

In Birch's implementation, 2 was also selected as the number of clusters to be used.

The results (shown in Table 5) of the clustering evaluation with both supervised and unsupervised metrics are almost completely identical to those obtained with Hierarchical Clustering.

Table 5. Metrics evaluation for Birch

Metric	score
Rand index	0.49997741887094355
Adjusted Mutual Info	8.851536954494379e-05
Homogeneity	8.666477313487239e-05
Completeness	0.0006090927099536045
V measure	0.00015173931379627306
Fowlkes Mallows	0.6929877807254788

Even with this algorithm, the clustering has created two unbalanced clusters in terms of numbers, underlining the inability to capture the diversity between the points (Table 6.

Table 6. Division in clusters for Birch

Cluster	voted_up	counts
0	False	9837
0	True	9760
1	False	218
1	True	185

The clustering results show an asymmetry in the distribution of points between the two clusters. One cluster contains significantly more points than the other. This indicates that the clustering algorithm was not able to distribute the points equally between the two clusters. In addition, it can be seen that there are very few points classified as belonging to cluster 1, for both positives and negatives. This could be an indicator that the clustering was not able to capture the diversity between the points.

Again, the use of word clouds (Figures 18 and 19) does not provide a complete understanding and fails to distinguish between words with positive and negative meanings.

Figure 18. Cluster 0: negative values

Finally, the scatterplot (Figures 20) shows a clear majority of points assigned to cluster 0, underlining a sub-optimal partitioning.

Figure 19. Cluster 1: positive values

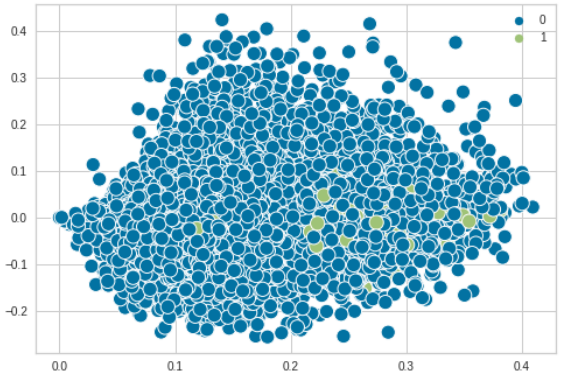

Figure 20. Scatterplot Birch

6.2.4. Optics

Optics is used to identify clusters of points in a data set based on the density of the points and their relative distances to one another.

6.2.5. DBSCAN

In the context of text clustering, DBSCAN can be used to partition a set of documents into clusters based on their similarity. It does not require the user to specify the number of clusters in advance and is able to identify clusters of varying densities and shapes, but it can be sensitive to the choice of the distance parameter (eps) and may not always produce the most intuitive or meaningful clusters. Other clustering algorithms, such as k-means or OPTICS, may be more appropriate in some cases.

Also with DBSCAN, as with Hierarchical, several metrics were used to test which one would be best for this use case

For both OPTICS and DBSCAN, no number of clusters can be set. In fact, these algorithms search for an optimal number of clusters and perform clustering on that number of clusters. However, this does not lead to any substantial advantages as the performance obtained is no better than that of the other algorithms (so the fact that the optimal number of clusters is automatically chosen is not an advantage).

In conclusion, it can be stated that all the text clustering algorithms used achieved significantly lower results than the reference paper (which used a sample of 100 and a sample of 500 items and achieved an accuracy of 84.5% and 100% respectively).

7. Conclusions

7.1. Text Classification

In summary, the models have relatively high accuracy, precision, and recall, indicating that they are performing well on the testing set. However, there is still room for improvement.

Looking at the results some models are misclassifying samples from class 1. Since we are using BERT, which is a pre-trained transformer model, it is possible that the model has not been fine-tuned enough on this specific dataset and task. Fine-tuning BERT on a large dataset can be computationally expensive, but it can help the model better adapt to the specific task and data at hand.

Another thing that could be beneficial is to perform additional experiments like trying different architectures and configurations to improve the performance of class 1.

However, our BERT models performed better than the Decision Tree model, as they achieved an accuracy of around 80%, while the Decision Tree model achieved the highest accuracy of 75% according to Zhen Zuo's paper.

7.2. Text Clustering

In conclusion, the analysis conducted on the use of clustering methods to effectively separate reviews into positive and negative revealed poor quality of clustering using word frequency (TF-IDF) and the statistical properties of TF-IDF.

The task of text clustering applied to this dataset did not bring a great added value for a better understanding and interpretation of the data, as all approaches identify a main group that contains almost all observations.

In general, hierarchical methods do not seem to be suitable for addressing this type of problem, nor do density-based models like DBSCAN or OPTICS seem to be appropriate.

References

[1] R. R. Septiawan, "Sentiment Analysis of Steam Review Datasets using Naive Bayes and Decision Tree Classifier" [Online]. Available: https://www.ideals.illinois.edu/items/106100.

[2] PRAYUDA, ADVERINO PUTRATAMA (2021), "CLUSTERING AND IDENTIFY GAMES REVIEW ON STEAM STORE" [Online]. Available: http://repository.unika.ac.id/27133/.