Goodreads Review: Text Classification and Text Clustering analysis

Marco Sallustio 906149 , Marco Guarisco 789244 , and Salvatore Rastelli 903949

Dipartimento di Informatica Sistemistica e Comunicazione, DISCO, Università degli Studi di Milano Bicocca

Piazza dell'Ateneo Nuovo, 1, 20126 Milano MI

Abstract
In this project, user reviews from the Social Network **Goodreads** where users can add books they have read or want to read to their profiles, sharing reviews, comments and votes. This reviews were analyzed through the use of text mining techniques. After carrying out an initial phase of text processing and text representation, the project continued with the classification of the reviews, through some text classification techniques - such as Support Vector Machines (SVM), Neural Network, and Logistic Regression. Next, a text clustering phase was carried out through the use of two algorithms: DBSCAN and K-Means.
Keywords
Text Mining – Text Classification – Text Clustering

1. INTRODUCTION

The project aims to analyze the dataset "Goodreads Book Reviews"[2] through text mining techniques, specifically through Text Classification and Text Clustering. Text Classification is the activity of predicting which data items belongs to a predefined finite set of classes. There are many types of classification, in our case it is Binary Classification, where each item belongs to exactly one class in a set of two (positive or negative). Text clustering, on the other hand, is the task of grouping a set of unlabeled texts in such a way that texts in the same cluster are more similar to each other than to those in other clusters.

2. DATA

We chose for this project the Goodreads Datasets which is a big dataset of books; they were collected in late 2017 from goodreads.com. They are also divided in three groups of dataset: (1) meta-data of the books, (2) user-book interactions (users' public shelves) and (3) users' detailed book reviews. Furthermore, they divided them by genre and in particular we used the dataset related to fantasy/paranormal books. This dataset contains a total of 3,424,641 detailed user review and other information divided in the following columns:

- user_id;
- book_id;
- review_id;
- rating;
- review_text;
- data_added;
- data_updated;
- read_at;
- started_at;
- n_votes;
- n_comments.

We chose to use only the columns "review_text", which indicates the text of the review, and "rating", which indicates the number of stars given to the book by the user, in a scale from 0 to 5. Then, we added a column named "sentiment", in which we annotated if a review is Positive or Negative, based on the number of stars given to the book: 3 or more for Positive, 2 or less for Negative; we used this information for the binary classification.

At first, it was thought that we could use the entire dataset to develop the work, but after several attempts, the computational time turned out to be too

long: it was decided to work on a sampled version of the original dataset, consisting in 100,000 reviews.

3. TEXT PRE-PROCESSING

We performed the following operations:

- Language based filtering: The library pycld2 was used to detect the review language, so that we included only English reviews;
- Normalizing case: all reviews are converted to lowercase;
- Remove numbers: all number within the text have been removed;
- Remove punctuation: all punctuation within the text have been removed. particularly useful in tasks for which the semantics of individual words is preponderant compared to that induced by the flow of the text;
- Remove repeated characters: all instances of character repetition are capped to a maximum of two;
- Stop words removal: by removing stop words, all those words that do not contribute to delineating the semantics of a review have been removed since they appear frequently in every textual corpus. The choice of stopwords is not unique, but must be contextualized and verified in relation to the task and the type of texts in the corpus. We used Python's NLTK package.
- Remove extra whitespaces: all extra whitespace within the text have been removed;
- Tokenization: we have divided the text into tokens, that is, units of meaning. The goal of tokenization is to make text analysis easier by breaking the text into more manageable chunks;
- Lemmatization: Lemmatization consists of reduction of inflectional/variant forms to the basic form, to simplify the analysis of words used in reviews. We also used the NLTK package for this phase.

We stored the results of the processing in a different column, and we noticed that 119 rows were null, meaning that those reviews consisted only of stop words and numbers, so they were completely removed in the processing: so, we decided to dump those reviews.

4. EXPLORATORY ANALYSIS

We present exploratory analyses conducted on the dataset both before and after the Pre-Processing phase. Initially, the original dataset exhibited a highly skewed distribution of ratings, resulting in a notably imbalanced binary class, which we termed 'sentiment.'

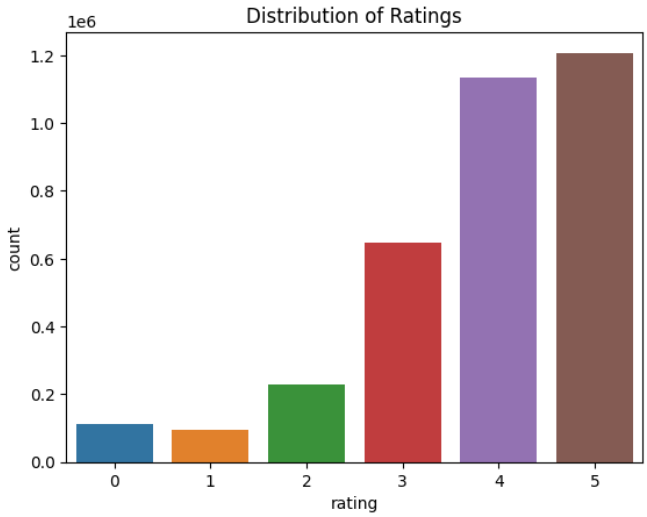

Fig. 1. Original Distribution of Rating

To illustrate this imbalance, refer to Figure 1 displaying the original distribution of ratings. Subsequently, in Figure 2, we depict the corresponding distribution of sentiment derived from this skewed rating distribution.

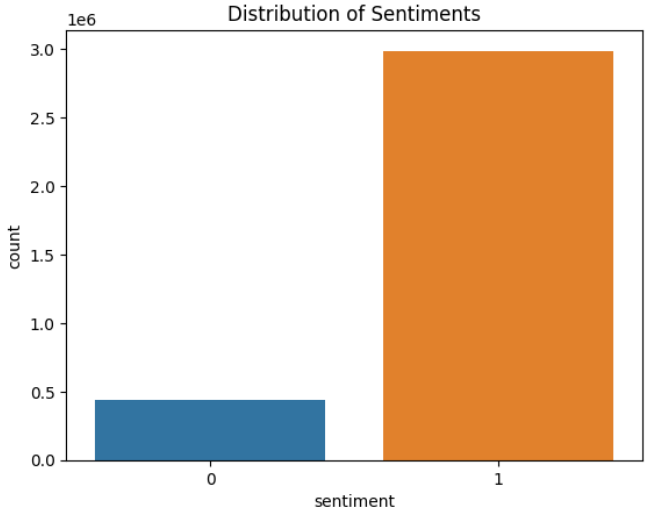

Fig. 2. Original Distribution of Rating

To address the challenges posed by the imbalanced dataset and the computational demands

associated with its large number of observations, we implemented a sampling strategy. This not only reduced computational costs but also effectively solve our class imbalance.

Here we show the new distribution of the created column 'sentiment' that reports only the NEG or POS values based on the corresponding rating value. To do this we created a simple histogram by counting the total POS values and the total NEG values. The result obtained is shown below:

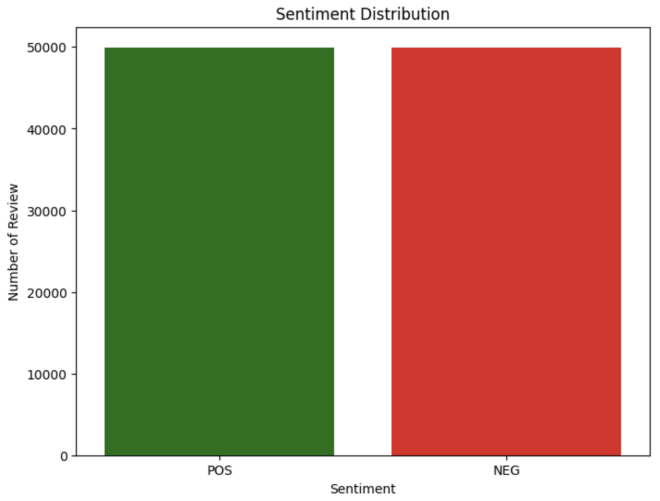

Fig. 3. Distribution of Sentiment after Pre-Processing

As we can see from the Fig.3 both partitions consist of almost perfectly balanced classes with 49919 values equal to NEG and 49962 values equal to POS. We also did the same thing but this time to analyse the distribution of the values of the column 'rating'. We have obtained the following result:

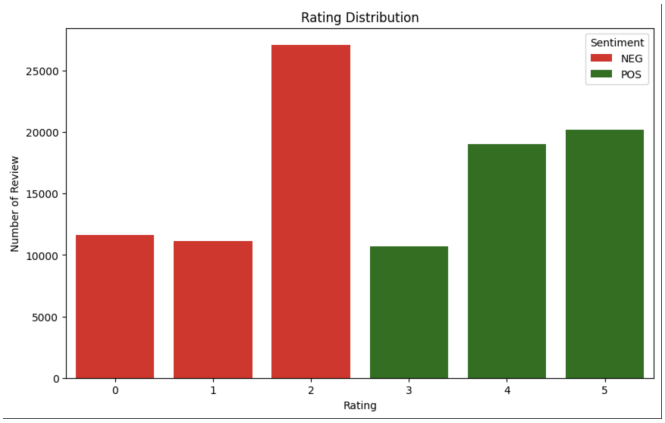

Fig. 4. Distribution of Rating after Pre-Processing

As we can see from the Fig.4 we have the highest count of reviews with a rating of 2 for the NEG class; while as regards the POS class, the highest count is for reviews that have a rating of 5. At this point we also analysed the average length of the review text, i.e. the average number of characters of the reviews distributed by rating. This could be used to understand if there are differences in the average length of reviews depending on the rating.

Fig. 5. Average Length of Reviews by Rating

As we can see from the Fig.5, reviews with a rating value of 0 have a smaller average length compared to the remaining rating values which all have a similar length.

In our final analysis, we conducted also a Trigram Analysis to explore the most frequent trigrams within reviews, aiming to identify significant word associations. We opted to include only the most notable trigrams, selecting those that appeared at least ten times.

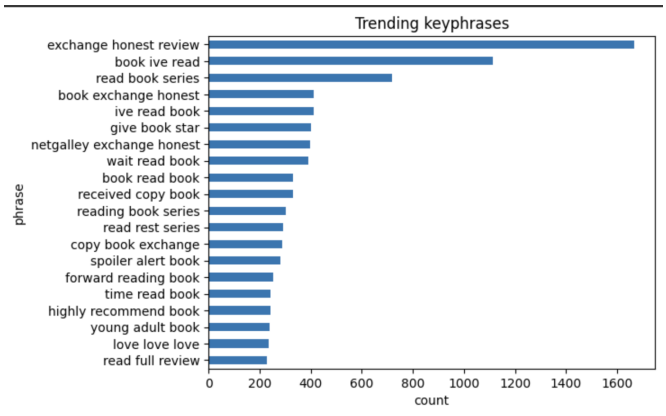

Fig. 6. Trigrams Analysis

5. TEXT REPRESENTATION

This section of the document describes the two text representation models adopted to train the models.

5.1. TF-IDF

TF-IDF is a word importance metric that is used to represent the relevance of a word in a document with respect to a set of documents: the metric is calculated as the product of two components, the frequency of a word within a single document (in our case within a single review) and the inverse of the document frequency, i.e. the unique importance of a word with respect to all documents.

$$W(x,y) = tf(x,y) \times \log(\frac{N}{df(x)})$$

The representation was implemented through the feature TfidfVectorizer of the sklearn package in python. The range of ngrams chosen was 1-3, that is, Unigrams Bi-grams, and Tri-grams. Tri-grams (three consecutive words) provide more context for text classification tasks than single words or bigrams (two consecutive words). By capturing the relationship between three words, tri-grams can better capture the meaning and context of text, improving the accuracy of text classification models. Regarding the variable 'max features' corresponding to the maximum number of terms to be considered (the most frequent grams across the text), 1000 was chosen as the value.

5.2. Doc2Vec

Doc2Vec is an embedding model. Embeddings are an information representation technique that consists of associating each element of a set of objects with a vector of numbers. Doc2Vec is based on an artificial neural network architecture that allows documents to be represented as vectors of numbers. This means that each document is mapped into a vector of fixed size representing its semantic and lexical features. In order to implement Doc2Vec, we first used the TaggedDocument() method, which takes care of associating a unique tag to each review (both test and train, since only one portion was created), and then the doc2vec() method, to which I passed the tagged reviews on the basis of which the model will be trained, the size of the vector that must represent each review (1000), the context window used to define the relationships between words within a review, the minimum number of times a word must be present in documents to be considered (1) and the number of workers (threads) to be used during the training process. Finally, the trained model was used to create the text representation for the set of train and test reviews

6. TEXT CLASSIFICATION

Text classification is the activity of predicting which, among a finite set of groups, a document belongs to. We can distinguish different types of Text Classification; in our case we have a binary classification where each item belong to exactly one class in a set of two: positive or negative. In order to have a binary classification we used the feature "sentiment" added in our dataset by us which can only have two values: NEG (negative) or POS (positive). We used three different classification models on the two different text representations: a Support Vector Machine, a Neural Network and a Logistic Regression. Comparisons between the results of the realised models are showed below.

6.1. TF-IDF

6.1.1. Support Vector Machine

	precision	recall	f1-score	support
Positive	0.764	0.751	0.757	15101
Negative	0.751	0.765	0.758	14864
accuracy			0.758	29965
macro avg	0.758	0.758	0.758	29965
weighted avg	0.758	0.758	0.758	29965

Fig. 7. Classification Report - SVM on TF-IDF.

SVM on TF-IDF achieves a balanced binary classification. Precision, recall, and F1-scores exhibit equilibrium for both classes.

6.1.2. Multi Layer Perceptron

	precision	recall	f1-score	support
Positive	0.757	0.707	0.731	15101
Negative	0.721	0.769	0.744	14864
accuracy			0.738	29965
macro avg	0.739	0.738	0.737	29965
weighted avg	0.739	0.738	0.737	29965

Fig. 8. Classification Report - Multi Layer Perceptron on TF-IDF.

MLP on TF-IDF showcases balanced binary classification metrics. Precision, recall, and F1-scores are evenly distributed.

6.1.3. Logistic Regression

	precision	recall	f1-score	support
Positive	0.764	0.753	0.758	15101
Negative	0.753	0.764	0.758	14864
accuracy			0.758	29965
macro avg	0.758	0.758	0.758	29965
weighted avg	0.758	0.758	0.758	29965

Fig. 9. Classification Report - Logistic Regression on TF-IDF.

Logistic Regression on TF-IDF demonstrates robust binary classification. Precision and recall maintain a well-balanced distribution for both classes. Macro and weighted averages consistently reflect this equilibrium.

6.1.4. Summary

In comparing the performance of the classifiers—Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), and Logistic Regression—it is evident that all three models exhibit similar levels of accuracy. Each classifier demonstrates a balanced approach in precision, recall, and F1-score for both positive and negative classes. SVM, with an accuracy of 75.8%, showcases a well-rounded performance with slightly higher precision and recall for both classes compared to MLP and Logistic Regression. MLP, achieving an accuracy of 73.8%, maintains a satisfactory performance, while Logistic Regression, also with an accuracy of 75.8%, exhibits a robust and balanced classification. Further fine-tuning of parameters and exploration of new classification methods could contribute to refining the models and potentially improving their performance.

6.2. Doc2Vec

6.2.1. Support Vector Machine

	precision	recall	f1-score	support
Positive	0.752	0.731	0.741	15101
Negative	0.734	0.755	0.744	14864
accuracy			0.743	29965
macro avg	0.743	0.743	0.743	29965
weighted avg	0.743	0.743	0.743	29965

Fig. 10. Classification Report - SVM on Doc2Vec.

SVM on Doc2Vec achieves a balanced binary classification. Precision, recall, and F1-scores demonstrate equilibrium for both classes.

6.2.2. Multi Layer Perceptron

	precision	recall	f1-score	support
Positive	0.764	0.741	0.753	15101
Negative	0.745	0.768	0.756	14864
accuracy			0.754	29965
macro avg	0.755	0.755	0.754	29965
weighted avg	0.755	0.754	0.754	29965

Fig. 11. Classification Report - Multi Layer Perceptron on Doc2Vec.

MLP on Doc2Vec shows a balanced binary classification performance. Precision, recall, and F1-scores maintain equilibrium with slight improvements compared to SVM.

6.2.3. Logistic Regression

	precision	recall	f1-score	support
Positive	0.747	0.742	0.745	15101
Negative	0.740	0.745	0.742	14864
accuracy			0.744	29965
macro avg	0.743	0.744	0.743	29965
weighted avg	0.744	0.744	0.744	29965

Fig. 12. Classification Report - Logistic Regression on Doc2Vec.

Logistic Regression on Doc2Vec demonstrates a balanced binary classification with an accuracy of 74.4%. Precision, recall, and F1-scores maintain well-distributed values for both positive and negative classes.

6.2.4. Summary

Comparing the classifiers with Doc2Vec embedding, MLP stands out with the highest accuracy at 75.4%, followed by Logistic Regression at 74.4% and SVM at 74.3%. All three models exhibit balanced precision, recall, and F1-score values for both positive and negative classes. While there are subtle differences in performance, the overall effectiveness of these classifiers remains consistent.

Comparing the two embeddings, Doc2Vec and TF-IDF, both demonstrate reliable performance in binary classification. TF-IDF appears more effective for SVM and Logistic Regression with slightly higher accuracies, while MLP shows a marginal dip with TF-IDF.

7. TEXT CLUSTERING

Text clustering [1] is the most common form of unsupervised learning and is the process of grouping a set of objects (text) into classes of similar objects (text). According to the cluster hypothesis, text in the same cluster behaves similarly with regards to relevance to information needs.

For this phase we chose to use two different algorithms: K-Means e DBSCAN. Before we began, however, we performed dimensionality reduction of the total dataset using the Singular Value Decomposition (SVD) on both the text representations. This process involves transforming the initial high-dimensional matrix into a more concise, lower-dimensional form. This resultant representation becomes important in clustering documents based on their similarity. The reduced-dimensionality output from SVD serves as input for subsequent clustering algorithms.

7.1. DBSCAN

DBSCAN[3], an acronym for Density-Based Spatial Clustering of Applications with Noise, is an unsupervised machine learning algorithm widely used for text clustering. Its operation is based on the aggregation of text documents that are similar to each other by virtue of the density of word occurrences. Its application is effective in clustering data based on density, identifying regions with a large number of samples. This feature makes it particularly suitable for dealing with noisy situations. The concept of “density-based” implies that the algorithm values regions characterized by a considerable sample density, that is, with a large number of closely spaced examples. The “spatial clustering” aspect denotes its operation through operations in feature space. In other words, while some clustering techniques communicate information between points, DBSCAN makes distance measurements in space to determine whether each sample belongs to a cluster.

The procedure begins by randomly selecting a document as the "seed" and subsequently identifying all other documents close to it, with "near" defined by a user-specified distance metric, such as cosine similarity. If the number of documents close to the seed is sufficient, they are aggregated into a cluster. This process is iterated for each cluster until all documents have been assigned to a cluster. The main advantage of DBSCAN lies in its ability to identify clusters of arbitrary shapes, a crucial aspect in those cases where clusters may not follow a spherical shape.

To visualize the clusters, we used the PCA to further reduce the dimensionality of the dataset to only 2 dimensions, and then we made a scatterplot coloring the observations based on the respective cluster assigned by the algorithm.

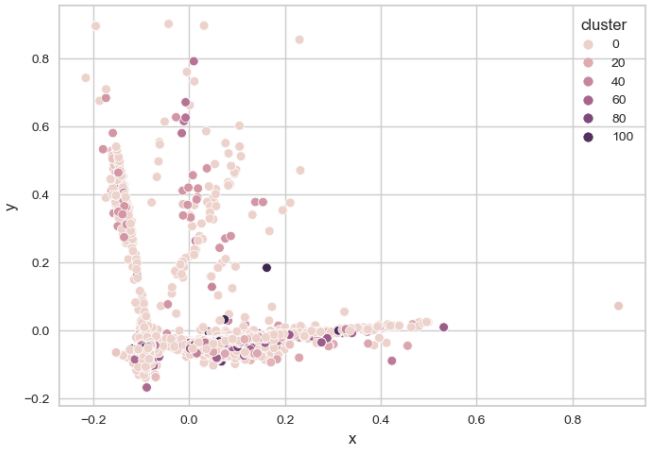

Fig. 13. DBSCAN clustering

7.2. K-means

K-Means is a flat clustering algorithm. The assumption is that documents are represented as length-normalized vectors in a real-valued space and its objective is to minimize the mean-square Euclidean distance of the documents from the cluster center. The k-means algorithm is defined as follows:

1. Select K objects as centroids;

2. Calculate the distance of each point with respect to the centroid;

3. Assign each observation to the cluster based on the minimum distance from the centroid;
4. Update the centroids by averaging the distances between all observations in the cluster and the centroid;
5. Iterate the procedure from step 3 until a stopping criterion is met.

In order to choose the value of K to initialize the algorithm, we used the Elbow method by employing the kelbow_visualizer() function:

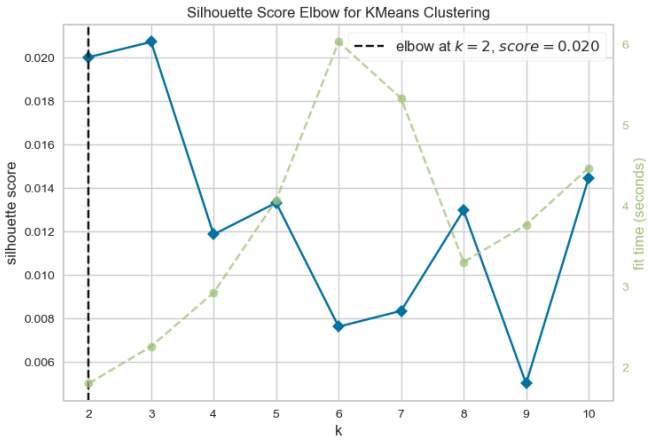

Fig. 14. K value chosen with the Elbow method.

Also in this case, we reduced the dimensionality of the dataset to only 2 dimension using PCA, then we made a scatterplot

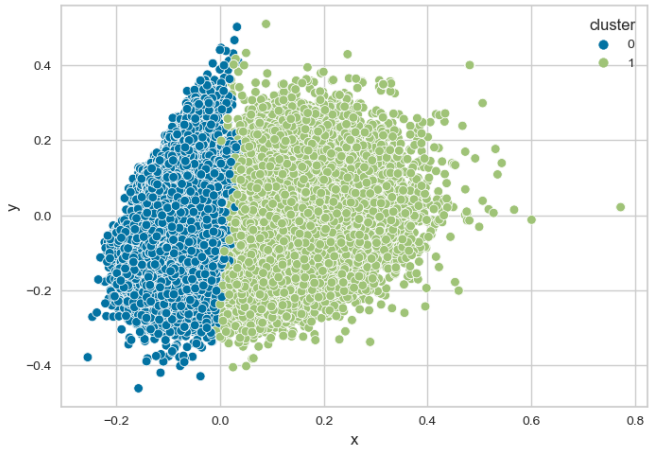

Fig. 15. K-Means Clustering

7.3. Evaluation

A good clustering algorithm will produce high quality clusters in which the intra-class (that is, intra-cluster) similarity is high, and the inter-class similarity is low. The most important Internal Evaluation index is the Silhouette Coefficient, that measures how well an observation is clustered and estimates the average distance between clusters. The Silhouette Coefficient is calculated using the mean intracluster distance *a(i)* and the mean nearest cluster distance *b(i)* for each sample *i*:

$$S(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

S(i) will lie between [-1,1]:

- If the Silhouette value is close to 1, the sample is well clustered and already assigned to a very appropriate cluster;
- If Silhouette value is about to 0, the sample could be assigned to another cluster closest to it and the sample lies equally far away from both the clusters. That means it indicates overlapping clusters;
- If Silhouette value is close to -1, the sample is misclassified and is merely placed somewhere in between the clusters.

We used this coefficient to evaluate the algorithms:

	DBSCAN	K-Means
Silhouette Coefficient	0.204	0.020

The value for the DBSCAN algorithm is higher than for the K-means algorithm, even though only 8800 reviews out of 10000 are assigned to a cluster. However, in both cases the value is near 0, suggesting that the clusters obtained are not well separated.

8. CONCLUSION

In the context of the Text Classification task, all models demonstrate a reasonable ability to predict the binary class. The best performance, however, were obtained using the Support Vector Machine on the Tf-Idf representation.

On the other hand, the Text Clustering task applied to this dataset did not contribute significantly to enhancing our understanding and interpretation of the data. The Silhouette coefficient is near 0 for both algorithms, indicating that the clusters found are not well separated and could be overlapping,

or that there is only a single big cluster in the data containing all the observations. Possible future developments could include different algorithms and different text representation techniques.

REFERENCES

1. Devopedia. Text Clustering. URL: https://devopedia. org/text-clustering.
2. Goodreads Dataset. URL: https://mengtingwan.github.io/data/goodreads
3. Christian Versloot. Performing DBSCAN clustering with python and scikit-learn. URL: https://github.com/christianversloot/machine-learning-articles/blob/main/performing-dbscan-clustering-with-python-and-scikit-learn.md