



UNIVERSITÀ DI PISA

Goodreads - Book Reviews

Text Analytics Project

PROFESSOR

LUCIA PASSARO

TEAM

LUDOVICO LEMMA

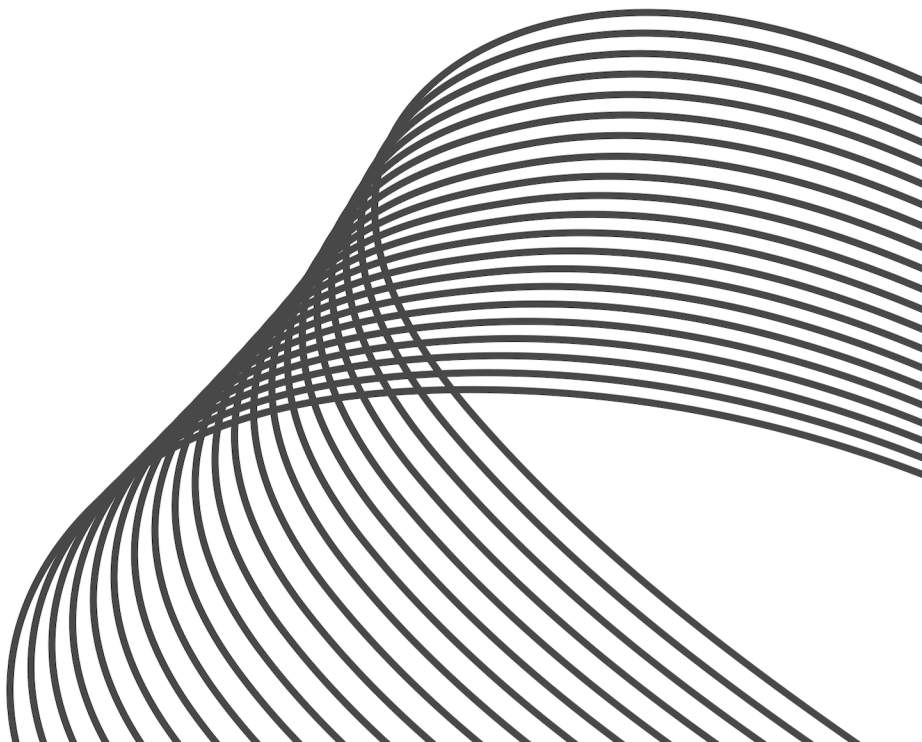
DAVIDE INNOCENTI

MARIA GRAZIA ANTICO

LIA TRAPANESE

CHIARA GERMELLI

2022/2023



Contents

1	Dataset	1
2	Data Scraping	1
3	Exploratory Data Analysis	1
3.1	Data understanding	1
3.2	Preprocessing	1
3.2.1	Preprocess the reviews	1
3.2.2	Preprocessing the target classes	2
3.2.3	Balancing the target variable and Under-sampling	3
3.2.4	Tokenizing and Lemmatizing	3
3.3	Topic Modeling	3
4	Anomalies	4
5	Classification	4
5.1	LSTM, SVM, Naive Bayes: Comparisons for Review Genres Classification	4
5.1.1	Top2Vec: Search of Semantically Similar Words	5
5.1.2	Consideration about Random Forest	5
5.2	Bert: Classifying Review and Book Genres	6
5.3	Sentiment Analysis and Rating Classification	7
6	Conclusions	7

Introduction

Nowadays shopping is an integral part of individuals' lives, and online shopping is constantly evolving: dozens, hundreds, if not thousands of items are sold every second on the web. In particular, websites provide spaces for users to rate and review their purchases. Even in the literary world, platforms that allow users to post reviews or comments on what they read are spreading. All of this allows not only the user to make the right choice when buying the book, but also to collect a vast amount of data for analytical purposes. This report aims to evaluate reviews by web users using artificial intelligence and natural language processing (NLP) techniques with the ultimate goal of developing classification models useful in genre and rating prediction.

1 Dataset

We had a dataset already split in training and test set by the challenge proposers. The datasets consist of 10 categorical and numerical features: user_id, book_id, review_id, review_text, date_added, date_updated, read_at, n_votes and n_comments. We also have two target variables, the genre and the rating. The training set consists of 900,000 rows, while the test set is made up of 478,033 rows.

2 Data Scraping

The dataset we decided to analyze was retrieved through a challenge proposed on the [Kaggle website](#).

The challenge itself was aimed towards the prediction of user ratings. We decided to further develop the challenge into a genre prediction task instead, our objective with this change of the original challenge was to get a possible classifier that make it possible to evaluate a book genre just from its reviews.

The first step of the process we established was to retrieve the genre. To do this we wrote a scraping algorithm. We had first to retrieve the unique book ids we had in our training set. Then, through the url structure we retrieved the source code of the page and then we extracted the most popular genre evaluation of the book as put in the "shelves" by the users. A summary of this process is shown in *Figure 1*.

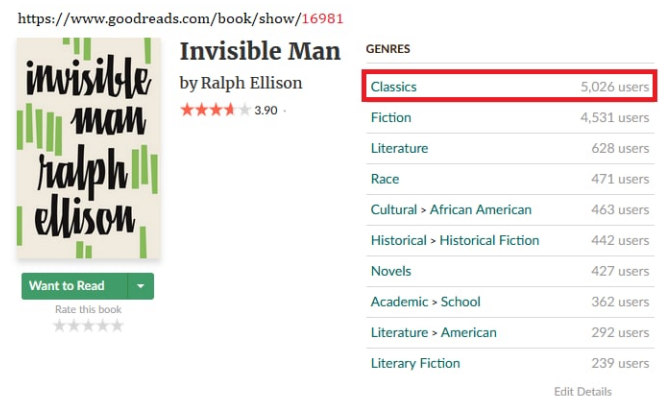


Figure 1. Example of Goodreads interface to execute the genre scraping.

3 Exploratory Data Analysis

3.1 Data understanding

As we can see from the heat-map in *Figure 2* we had some NaN values in the columns read_at, started_at and genre; we decided not to use the first two as they were not required for our research.

We checked the users' behavior in relation to the number of reviews they made, and we obtained a power-law distribution showing that with the increase of the users, the number of reviews decreases: this means that very few users review a lot of books, while many users review fewer books.

We observed the same phenomenon with the books and the number of reviews: many books had few reviews, but a handful of them had too many.

3.2 Preprocessing

3.2.1 Preprocess the reviews

Regarding the fact we had too many reviews per book we made some further analysis : on average we had 35 reviews per book.

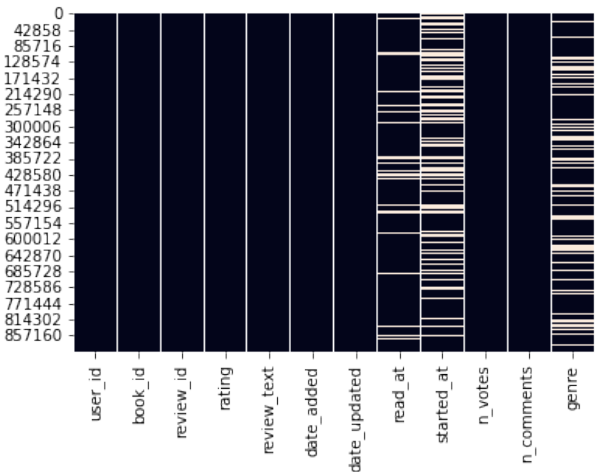


Figure 2. Heat-map showing Nan values in the columns considered.

Moreover, we noticed that the most popular books, namely the ones with the highest number of reviews, had some words (for example the names of the characters, places etc) that repeated themselves and the model adapted to these individual cases. To solve this issue, we performed a sampling of at most 35 reviews per book, and we went from 900,000 to 501,745 rows in the training set.

Right after it we performed various types of data cleaning and measured multiple times the performances. We decided to adopt the following preprocessing of the text:

In the first place we decided to tackle the problem of NaN genres for some reviews, they were 71,220 and we cut them out entirely reducing the dataset to 430,525 total rows;

We performed a cleanup of the reviews by eliminating the characters or terms that compromised the most the performances of the classifiers. For example, we removed urls, html blocks, numbers within reviews, and words that contained these.

We eliminated reviews that had the word "spoiler", as it was too frequent (it is a common practice to avoid people reading beyond, but for our scopes, it wasn't needed as the books we classified were all of the narrative kind) and dealt with punctuation in the text. We also removed explicit ratings in the review text (useful later for the rating task). The result of all this is a non-capitalized text cleaned of possible problems for our tasks.

By looking at the data we figured out there were some duplicate reviews, we had 6.234 of them, we had to delete them as they created a risk of overstating the importance of some books and genres.

we also considered reviews referencing to reviews themselves. We checked those contained a lot of spam eventually redirecting to some external source "where the full review would be presented") and delete them as well, dropping 81,033 rows.

We then moved on to constructing some basic measures to better analyze the corpus: we indeed calculated the minimum and maximum character length of the single reviews, they were in a range of [0, 15,028] characters (after cleaning), mean and median.

The most decisive step of the preprocessing was the selection of records within a specific range of characters. The improvement of the accuracy on the validation set of most classifiers was as high as 10%. The thresholds of the best number of characters were between the median and the 90th quantile. We selected thus only those reviews. This leads us to a new dataset with a count of 137115 records.

3.2.2 Preprocessing the target classes

Our dataset was initially composed of 74 different genres with various frequency distributions. This led us to merge the classes with fewer records with those we considered most similar having a greater number of reviews.

Unfortunately for those genres having literally less than 10 reviews we had no better solution than removing them from the dataset since we couldn't find any merging reasons. For the merging operation we used a hash-map to map our 74 genres into a narrower subset of genres of only 10 units (6 units later on instead). The resulting, in 1 includes the following reviews per genre:

First Merge of Genres	
Destination Genres	Original Genres
"Fantasy"	"Fantasy", "Superheroes", "Shapeshifters", "Science Fiction Fantasy"
"Romance"	"Romance", "Erotica", "Polyamorous", "Category Romance"
"Fiction"	"Fiction", "Young Adult", "New Adult", "Womens Fiction", "Adult Fiction", "Christian Fiction", "Realistic Fiction", "Fan Ficon", "Magical Realism"
"Art"	"Sequential Art", "Music", "Couture"
"Thriller"	"Thriller", "Mystery", "Crime"
"Science Fiction"	"Science Fiction"
"Horror"	"Horror", "Paranormal", "Dark", "Suspense"
"Literature"	"Classics", "Contemporary", "Poetry", "Plays"
"NonFiction"	"Nonfiction", "Autobiography", "Biography"
"History"	"Historical", "History", "War", "Mythology"

Table 1. First merge of the target classes

After this first merge, though we reached some vague improvement in the performances we considered it fruitful to reduce further the complexity of the classification. This was done after a first run of the Bert classifier which provided a low accuracy close to 50% on the validation set.

Thus we performed a further join based on the visual reduction we obtained through the UMAP as shown in Figure 3, the closest categories with a semantic relation were merged:

- Fantasy was merged with Fiction
- Horror and Thriller were fused into a Mystery class
- Literature and History were merged with NonFiction

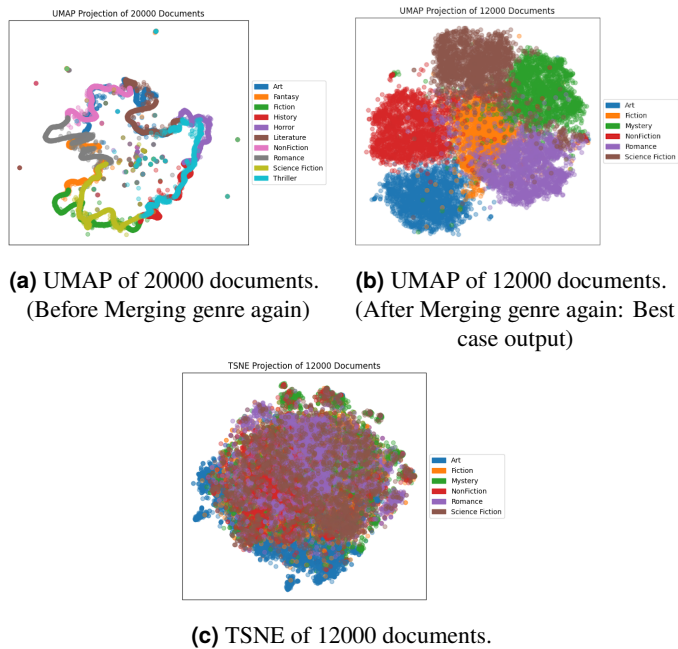


Figure 3. Before-After further genre reduction.

3.2.3 Balancing the target variable and Undersampling

Since this is an NLP task we have to deal with complex algorithms that require a great deal of computational effort. In addition, we had a dataset that was as large as it was unbalanced *Figure 4* at the class level.

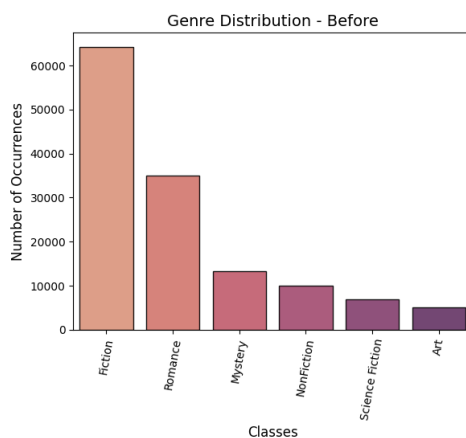


Figure 4. Genre classes unbalanced.

For this reason we were forced to apply an important under-sampling on our dataset, keeping about 1.3% of the initial data, by considering for each class 2,000 records belonging to it, with a final result of a balanced training dataset with 12,000 records.

3.2.4 Tokenizing and Lemmatizing

Having restricted the amount of tuples we now have, we can now use the NLTK module to set up our dataset for the classification tasks.

First, we proceeded with the **tokenization**, which is to split paragraphs and sentences into smaller units that can be more easily processed by machines. In this part we also took care of stop words by removing them from the dataset. For doing that we used the Gensim library rather than NLTK because the former was definitely more complete.

Then, we wanted to construct a **POS Tagged** representation of our tokenized text. It provided a contextualized representation of words such as nouns, verbs, adjectives and satellite adjectives. This was a necessary step for our objective of Lemmatizing the tokens.

Finally we added the column of the lemmatized text. **Lemmatization** is responsible for grouping different inflected forms of words into the root form, identified by the word's lemma, or dictionary form. This particular technique forgets about the context of the words, but it can be very useful for some classifier.

3.3 Topic Modeling

As we will show later this task was relevant for the improvement of the classification task of various non-pre-trained classifiers. We used LDA to discover the top 6 topics. In order to reach this objective it was necessary to build a dictionary from the corpus. The process was straightforward: We extracted first the words by their frequency and distribution across the texts in the corpus. Then we selected a minimum threshold for the word to be considered part of the dictionary, considering the number of reviews we had (12,000 after the main preprocessing), we decided that the presence of the word in at least 5 reviews would be a good way to exclude apax legomena and words too much related with a specific book rather than a genre (considering we still have at most 35 reviews per book), then we selected as another filter to exclude a word from the dictionary a maximum threshold of 99% of frequency across improvement reviews to reduce the number of outlier words that appears in almost any review, later we further reduced the maximum frequency to 70% to avoid catching the words that appear in most reviews. We included bigrams at first but as in the case of the classification part adding up bigrams or even worse trigrams reduce the performances.

genre	freq	cluster	topic_word
Art	36	4	ghost
Fiction	64	5	narrator
NonFiction	85	5	narrator
Romance	211	2	sexy
Science Fiction	254	1	alien
Mystery	98	4	ghost

Table 2. Most frequent topic word per class

Later we directly applied the LDA model to get the topics and the topic words. We found out that the topic words were almost always correlated with the genres. After getting the top topic words we indeed extracted the top topic word of each topic found and we measured the frequency of each across each genre class.

The words that would result as the most important according to the model are not predetermined and thus at each new run new top topic words would be discovered. We resumed the top topic words with the highest frequency for each genre in Table 2. In particular, after multiple runs of the model, science fiction emerged almost all the time with a word semantically connected with the genre. After this task we also tried to perform a qualitative analysis by plotting various word clouds of the genre so that we could get a measure of how representative the frequency of the words would in the genre, the result weren't as exciting as in the case of the topic modeling but by reducing the top frequency of words we start to get some more genre-related words. An example of the type of qualitative analysis is shown in Figure 5.



Figure 5. Word clouds.

4 Anomalies

During the pre-processing and subsequent steps, we noticed various problems with our current dataset, problems which we solved with our heavy pre-processing but that would present themselves greatly on the test set. In particular, the dataset has some evident outliers:

In many cases there are reviews that may be considered spam, we filtered them out simply by removing some of the shortest reviews after dealing with the removal of URLs, but this approach causes also the removal of some short and concise positive or negative reviews. In fact, by selecting reviews of medium length we probably selected a subset of users with some sort of bias, which in turn influenced our performances on the test set, but most important a spam detection task would need to be built on top and before the actual classification task so that spam reviews would be classified. This would need an entirely different project though, so we decided to keep the spam reviews in the test set as they were after cleaning, indeed we consider that one of the reasons for the difference in performances between the validation and test set performances with some classifiers (in others like the Naive Bayes we consider just an overfitting the cause of the decay in performances).

Another problem we detected in the dataset is the presence of another kind of review with the rating explicit in the text of the review itself, sometimes with just the rating expressed without further opinions. It seems a minor problem at first but the rating

classification task would be greatly affected by this problem as the presence of explicit ratings in the corpus creates the possibility of cheating through the extraction of the rating from the review themselves. To quantify and measure this problem we computed the number of reviews with numbers from 0 to 5 inside them, after a simple inspection of the resulting reviews it is clear the magnitude of the problem: around 1/3rd of both the training (354,298 over 900,000) and test set (187,608 over 478,033) has probably the rating explicit (or at least numbers from 0 to 5), invalidating the classification task, as such we decided to perform a simple analysis regarding the ratings, correlating it with the sentiment analysis after removing the numbers inside the reviews and we focused more on the classification of genres.

In conclusion we decided to keep the anomalies as they were, even though we were conscious of them.

5 Classification

To try to overcome the problems encountered and described in the previous section, different classifiers were tested and the variation in performance for some of them was observed when varying different embeddings and strategies. In particular, the LSTM, SVM, Naive Bayes, and BERT were used for genre classification, and finally, by exploiting the probabilities extracted from the BERT predictions, the genre classification of books was attempted.

5.1 LSTM, SVM, Naive Bayes: Comparisons for Review Genres Classification

For genre classification, lemmatized text was used for each classifier processed using the following:

- **Tokenizer** (provided by Keras, in general used for deep neural networks)
- **CountVectorizer**
- **CountVectorizer with Umap Reduction**
- **Tf-idf**
- **Tf-idf with Umap Reduction**
- **Top2Vec** creating a new filtered column based on words similar to the different genre classes

The idea of using, for both CountVectorizer and Tf-idf a dimensionality reduction with Umap, comes from the observation of Figure 3b in which a possible distinction between the classes is noted. Since Umap reduces dimensionality based on the Topics identified in the text, it was thought that this method might be useful in detecting the class of the genre on which the prediction difficulty lies. The idea, for the case of **SVM**, worked in that one is able to achieve, using 2 components 32% accuracy on validation and using, instead, 6 components (as many as there are classes), the model reaches 53% accuracy using the rbf kernel.

As for **Naive Bayes**, the best results on validation are obtained using only CountVectorizer or Tf-IDF (in both cases, about a 94% accuracy on training and 73% on validation), without using Umap reduction. The related confusion matrix is shown in Figure 6.

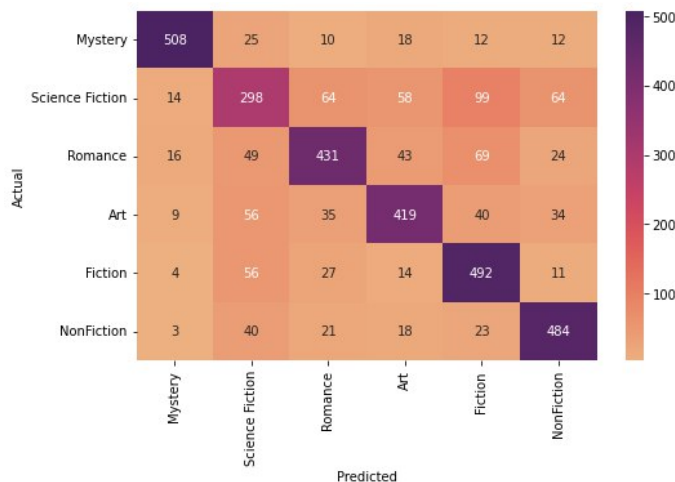


Figure 6. Confusion Matrix of the Naive Bayes Classifier.

For the **LSTM**, it is possible to deduce how, in our case, it fails to recognize the genres of the reviews well since, despite using different models for the same, different layers, dropouts and optimizers, the threshold of accuracy of validation never exceeds about 30% except in one case (by inserting, as input, the matrix obtained from the CountVectorizer reducing dimensionality with Umap to 6 components). Although varying the Patience values, as the epochs go on in all cases the validation loss never decreases but on the contrary increases, which is why it was thought, as will be described in the next section, to implement a more complex LSTM (exploiting the BERT probabilities).

5.1.1 Top2Vec: Search of Semantically Similar Words

In general, in cases where Umap reduction has been applied, there is a noticeable improvement in performance particularly for SVM. For this reason, always taking advantage of the possible inference that, thanks to topic recognition, a better classification can be obtained for the genre of reviews, a different strategy was considered using the **Top2Vec** library, which allows topic recognition from text, semantic search and also generates jointly embedded topic, document and word vectors.

Initially, we applied the model on the text using as a pretrained embedding model universal-sentence-encoder (the only one that could be used in the out case being that, the other two models available, concerned multilingual cases). Next, thanks to the `.similar_words` function available in the library (with parameters `keywords` and `n_words`), semantically similar words were identified for each keyword, which in our case were those corresponding to genders, so as to obtain a set of semantically related words that could help in genre prediction. For this purpose, a new column was created in the dataset where, for each record, only the words that were identified by the previous function are filtered from the lemmatized text. Thus, in this way, we wanted to observe how and whether the results of the genre classification of reviews could improve by giving as input to the classifiers the set of these words that are semantically similar to genres. This inference was found to be correct again in the case of SVM, where, on

validation, having the new filtered column as input and using CountVectorizer, **55%** was achieved using 200 similar words for each genre (`n_words=200`) while, using a number of 500 similar words for each genre, **61%** validation accuracy was achieved (compared to 32% found on the original lemmatized text).

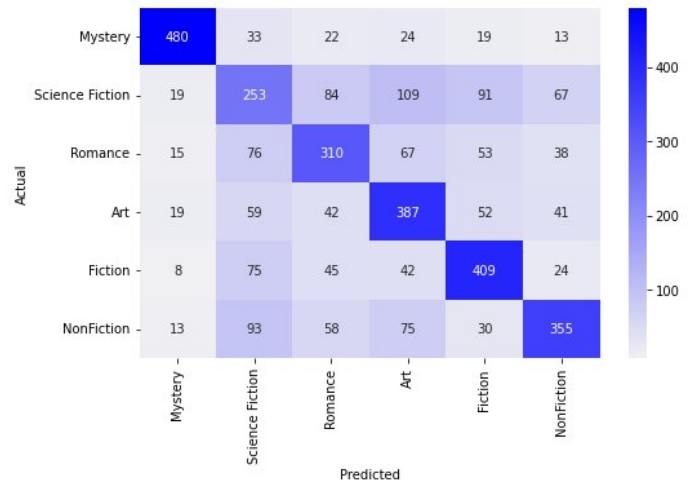


Figure 7. Confusion Matrix of Top2Vec.

Finally, to observe the results of the **Test set**, only the models in which the best results were obtained for validation were selected, obtaining, however, as was possible to expect given the composition of the dataset, poor results including a 20% for Naive Bayes. The best score was obtained, using as input the new column filtered thanks to the similar keywords with Top2Vec and transformed then with CountVectorizer reducing the dimensionality to 2 components with Umap, obtaining a **55%** final accuracy. The latter does not, however, match the accuracy of the validation of the same model (CountVectorizer + 2-component Umap), as it turns out to be 32% (with 6 components instead it reaches 54%). Having used 1,000 semantically similar words for the test set and not 500 (given the larger test set size compared to the training set), we assumed that the test score is higher due to the fact that, a filtering of a larger set of semantically similar words for the test set, helped to improve its prediction, perhaps due to the composition of the test set of many words that are semantically similar to the genres. Using the same number of words for each genre that was used for training, i.e. 500, in fact performance only reaches 28%.

5.1.2 Consideration about Random Forest

We also decided to implement a Random Forest classifier using both the CountVectorizer and the TF-IDF technique. The model was trained on the preprocessed training set consisting of 12000 rows and tested, initially on the Validation set and later on the cleaned Test set. The entire analysis was carried out on the "lemmatized_text" column and considering the target variable "genre". No extensive preprocessing was required: we cleaned the column 'lemmatized_text' of special characters and then passed it as a list of reviews to the CountVectorizer and TF-IDF in the

required manner. After training the model, we noticed that the best performance was obtained with the CountVectorizer where an accuracy of 63% was achieved on the validation set and 40% on the test set, a performance that was optimal for our goal. In contrast, with the TF-IDF technique, we achieved an accuracy of 42% on the validation set and 41% on the test set.

5.2 Bert: Classifying Review and Book Genres

BERT (Bidirectional Encoder Representations from Transformers) is a computational model developed in 2018 by researchers at Google belonging to the HuggingFace package. Many pre-trained models have been released since then, each of them regarding specific NLP tasks. In our case, we chose one of the most famous ones, not only for its own popularity but also for its purpose since it's the most adaptive one: "Bert base uncased", which of course tends to perform better than the cased version.

The basic idea of this classification task was to retrieve a list of logits extracted from the predictions of the reviews to train an LSTM on the records connected to each book to predict the book genre.

BERT is applied on the raw (even though preprocessed as presented in the previous) text since its pre-trained status enables to process the input text automatically to make it possible to be trained again on the classification task on hand.

We first split the Training set by considering 20% of the data as Validation set, then the following steps have been performed:

1. Add special tokens at the beginning and the end of each Tokenized text, respectively [CLS] and [SEP]
2. Set token to have a fixed length, and this is done by adding dummy values [PAD] to the series to match the desired length. In any case, the latter can't be above 512 tokens because of Bert's limitations.
3. Create an attention mask that tells the model to consider or not tokens when learning their contextual representation. This mask is a list of 0/1s.

All the above activities have been followed by creating the proper structures to let the library process the data, for this reason, we transformed all training, validation, and test sets into Torch tensors (matrix-like structures with a fancy name), also creating ram-friendly Data loaders iterators for each of them.

For the fine-tuning task, we have given a look at the official BERT paper which suggests tuning really few parameters with very few associated values; more precisely we tuned the:

- Epochs
- Learning rate
- Weight Decay

Which we have put into a dictionary and implemented as a `hyperparameter_tuning` function to try them out in a combinatorial way to record the result on the validation set.

Something to be said is that this pre-train model tends to rapidly overfit on the train data, so as it's always a good idea not to exceed

on the epochs count, differently from every other neural network paradigm, we considered 3 the highest possible number of epochs.

Running our model on the validation set we reached at most 65% of accuracy; while running on the Test set we achieved **55%** of accuracy and a **0.48 f1_score** on the reviews, collecting true/predicted labels and logits for each record.

Considering the large computational cost of this classifier after some runs we incurred a reduced number of resources to operate on, as such we moved to a simplified version of Bert so that there could be fewer issues and limitations.

From the previously trained BERT, we planned to develop a Long-Short-Term-Memory (LSTM) network to classify the genres of books exploiting the probabilities extracted from the BERT predictions of the reviews. A new train, validation, and test version of the dataset was created by aggregating by book the various logits mutated into lists of probabilities.

This last modification required special preprocessing, since considering the number of reviews for each book, represented by the lists of logits connected to that record belonging to a specific genre, these could obviously vary from book to book, with a maximum ceiling of 8 reviews maximum per book in our training set, all to the detriment of the neural network in question that needs fixed input dimensions.

To do this, a padding function was implemented in order to extrapolate a three-dimensional stable series of the form (5699, 8, 6), more precisely 5699 books for 8 reviews, for 6 classes: `book_ids` that do not reach 8 reviews are filled with dummy lists consisting of a single dummy value that finds no context in the probability field so as to be subsequently detected and ignored by a masking layer. The model also requires one-hot-encoding of the target variable, resulting in a sparse vector with a value of 1 at the membership class.

Finally, the LSTM network was initialized according to a sequential model: Reconnecting to the discussion earlier, as a first step, a Masking Layer was implemented to ignore those records containing previously entered special values for stability issues. Then an internal LSTM layer with 50 neuron units was added to the model, followed by a dense layer with a SoftMax activation function.

The model was then compiled by studying the loss function "binary_crossentropy" (which is common in LSTMs that require precisely coded binary labels) and an "Adam" optimizer, which in the literature seems to be the best performing in a wide variety of applications, launching it by setting an Early Stopping with the patience of 15 epochs useful to avoid possible overfitting.

The model seemed to generalize well on our optimized training set (in particular the median/higher length) with performances around 70% of accuracy for the prediction of book genres, but on the test set the results were much lower, namely 33% of total accuracy, with greater problems of predicting Fiction, Science Fiction, and Non-Fiction, while having fewer problems in predicting books in the Art category.

After some consideration we decided to try an easier strategy that worked incredibly well: we just measured the mode for each book prediction, the highest the number of predictions of a certain

class for that book, the higher the probability the actual class would be really that one. By doing that we reached a 63% of accuracy in the prediction of the book genres, we resumed the classification task with Bert in Table 3

	Predicting Genre of the Reviews				Predicting Genre of the Books			
	precision	recall	f1-score	support	precision	recall	f1-score	support
Art	0.52	0.69	0.59	12633	0.83	0.91	0.87	870
Fiction	0.78	0.51	0.61	213931	0.69	0.58	0.63	9299
Romance	0.56	0.60	0.58	77884	0.76	0.69	0.72	5925
Science Fiction	0.31	0.58	0.40	17822	0.54	0.75	0.63	1036
NonFiction	0.28	0.62	0.39	24121	0.41	0.74	0.53	1465
Mystery	0.32	0.55	0.40	26909	0.45	0.50	0.47	2940
accuracy			0.55	373300			0.63	21535
macro avg	0.46	0.59	0.50	373300	0.61	0.70	0.64	21535
weighted avg	0.63	0.55	0.56	373300	0.66	0.63	0.64	21535

Table 3. Bert Genre Classification

5.3 Sentiment Analysis and Rating Classification

After the preprocessing, thus removing the rating numbers explicit in the reviews and an ad hoc balancing of the dataset according to the rating distribution (2,000 records per rating class also here) we decided to perform a sentiment analysis of the reviews in the training set. Considering the nature of the analysis itself and the absence of training on our part we just ran the model and classified the sentiment of the reviews on a scale from 1 to 5 (in practice how much of a good or bad feeling the review gave). The model used for the task was the BERT Sequence Classification Multilingual Sentiment. After running the tokenizer requested by it we just waited for the computations to end. The results were strangely better than expected, also considering some ulterior runs of the Bert classifier on the same training set which gave poor results (akin to 20%).

With the output labels we got we had the idea of correlating the sentiment with the actual ratings we had, as a result, the Pearson correlation was 33%, finally, we also figured that by joining the rating 0 to the rating 1 we could also get some approximate classification metrics, considering the two types of labels are on similar scales. The results can be seen in Table 4, we got a total accuracy of 31% which seems low but it is higher than most classifiers we ran with the preprocessed training set with our heavy preprocessing. We noticed that it becomes easier to predict the highest ratings while predicting the lowest is more difficult.

	precision	recall	f1-score	support
1	0.09	0.67	0.16	511
2	0.17	0.25	0.20	1031
3	0.36	0.26	0.30	2991
4	0.51	0.27	0.36	4584
5	0.53	0.39	0.45	2883
accuracy			0.31	12000
macro avg	0.33	0.37	0.29	12000
weighted avg	0.43	0.31	0.34	12000

Table 4. Sentiment Classification

As said in the Anomalies section we concluded classifying the rating as they were would be fruitless in a challenge context, as the

obvious choice with such a test set would be to extract the numbers from the reviews. We tried to quickly train a model with Bert on the dataset we cleaned from numbers (thus removing possibilities for cheating) and the performances on the training set weren't that high (around 20%) considering the limited computational resources we couldn't make further tests.

6 Conclusions

In this paper, we went through a lot of NLP techniques to approach a classification task trying to predict the genres of reviews we collected as training set. We pre-processed every review to extract clean text out of each of it, we tried out several tokenization strategies to extract latent text structure and we also rearrange the distribution of the whole dataset in order to have balanced and well-representatives records.

Unfortunately, from what we had witnessed, we had not the best test data to try out our model on, mostly because being textual data it should be not only heavily preprocessed anyway but also cut as needed, even tho this is not to really always supposed to be done in a real context.

Still, even if real data is usually rough to work with, we focused on creating robust models aiming for the best performances on validation sets which managed to go beyond 65% of the main metrics most of the time.

In order to have a proper view of neural language models we also exploited the state of the art in terms of pre-trained models, such as BERT, which is able to extract coherent knowledge even more than we manually could and bring us decent results touching 55% of accuracy, probably due to its rich background of contributions.