Report Text Mining and Search

University of Milano-Bicocca

Marco Martinez 873849
Deborah Testa 817343
Luca Maggi 866654

Abstract: The aim of this paper is to explore and apply two of the main techniques related to textual analysis: text classification and topic modelling. Starting from a dataset containing book reviews all the phases involved in the analysis are discussed. After some cleaning and pre-processing operations a useful filter is created to identify safe-to-read reviews from the ones containing spoilers. In the second part of the paper, on the other hand, a model is implemented in order to identify the most discussed topics in the reviews. Hopefully, following this procedure, the most discussed genres can be identify.

Keywords: SVD | Text Classification | LDA | Topic Modelling

Table of Contents

1. Introduction

The whole project was developed in Python language, through Google Colab tool. Four main steps are followed in this paper:

1. NLP pre-processing
2. Document Representation
3. Text Classification
4. Topic Modeling

A. Dataset. The dataset contains more than 1.3M book reviews written by 18,892 users and about 25,475 books. (download dataset).

The dataset contains the following fields:

- **user_id**: user's identifier.
- **timestamp**: date information expressed in the yyyy-mm-dd format.
- **review_sentences**: reviews content.
- **rating**: score given by the user, it assumes values from 0 to 5.
- **has_spoiler**: true if the review contains a spoiler, false otherwise.
- **book_id**: book's identifier.
- **review_id**: review's identifier.

B. Preliminary operations. Given the size of the original dataset and the limited computational resources provided by Colab a subset of 7000 instances is used. A random sampling is used in order to maintain all the statistical properties present in the original dataset.

2. NLP preprocessing

One of the first steps when dealing with text related tasks consists of a deep cleaning of each review. Most of them do contain, in fact, emoticons, meaningless words, punctuation and other useless character for a proper analysis. To get an usable set of textual data the following steps are applied:

1. **Case Folding**: consists in reducing all letters to lower case. Python allows the *lower()* method that returns a string where all characters are in lower case.

2. **Removal of**:
    - **URL**
    - **Emoji**
    - **Numbers**
    - **Stopwords**: Those very commonly used words that do not provide any useful information because of their little semantic content. *NLTK* (Natural Language Toolkit) provides a list of English stopwords.
    - **Punctuation**
    - **White Spaces**
    - **Meaningless words**: A specific function to remove words that do not hold any sense in the English language (e.g. heavily influenced slang words).
3. **Tokenization**: To extract the tokenized words the *nltk.tokenize.word_tokenize()* method is used.
4. **Lemmatization**: To perform lemmatization a lexical database for the English language called *Wordnet* is used.
5. **POS**: POS tagging is generally useful to identify the syntactic role of each word in the sentence. The *pos_tag()* method contained in the *NLTK* package is used.

3. Text Representation

In order to perform the tasks in the best way possible, a text representation is required. Computers and other devices can not read, nor analyse, plain texts as humans do, therefore, to obtain significant results, a more suitable form is needed. In particular, during this work, two representations will be discussed: the Bag of Words (BOW) and the TF-IDF (Term Frequency, Inverse Document Frequency). This approach has its roots in the algorithms implemented for each task. Topic modelling is efficiently carried out through the LDA function (*Gensim* package) and, as stated by the documentation, this specific function requires a BOW representation. As far as the classification is concerned a Naive Bayes classifier is employed. No specific text representation is strictly required, therefore, instead of a simple BOW, a more informative TF-IDF representation is provided to the classifier.

A. Bag of Words. The BOW representation is created by assigning to every word in a document a raw frequency based on the number of appearances of said word. This process is repeated for each document in the corpus, resulting in a table of frequencies for every word in every document. As previously mentioned, this representation will be employed to compute a *Latent Dirichlet allocation* modelling. This procedure is quite expensive under a computational point of view, for this reason, the BOW is created considering only uni-grams.

B. TF-IDF. The TF-IDF representation, on the other hand, is a little more complex. It takes its name from the approach followed to assign a numerical value to each word in a specific document. In particular the value of the word increases proportionally to the number of times it appears in the document itself and it is offset by the number of documents in the corpus that contain the word. The formula applied is the following:

$$TF(t) = \frac{term\ frequency\ in\ document}{total\ words\ in\ document} \qquad [1]$$

$$IDF(t) = \log_2\left(\frac{total\ documents\ in\ corpus}{documents\ with\ term}\right) \qquad [2]$$

where t = Term

This approach, compared to the BOW, is more informative. It's aim is to weight the words based on the influence they carry in the document but, at the same time, to reduce the weights for those terms that, being in a lot of documents, run the risk of overcoming other informative terms that are just less frequent. The overall computation is performed through the *TfidfVectorizer* (*sklearn* Package). The classification, being a computationally lighter task, lets us compute bi-grams before calculating the TF-IDF values. Using bi-grams provides the unique advantage of better capturing dependencies and correlations between pairs of words. Of all the bi-grams created, the ones with a frequency less than 3 were dropped.

C. SVD. A different approach to text representation is word embedding. The aim of these techniques is to create short and dense vectors out of the long and sparse ones that are generally generated by simpler representations. An embedding procedure for the available corpus is provided as an mere example in this work. This representation will not be used for later tasks, but it is nevertheless shown as a further way to represent textual material.

In the same manner as the previous case the words are grouped in bi-grams, then, instead of the TF-IDF values, PPMI are computed through the following formula:

$$PPMI(w1, w2) = \max\left(\log_2\left(\frac{P(w1, w2)}{P(w1)P(w2)}\right), 0\right) \qquad [3]$$

where w1, w2 = word1 and word2

Then a square matrix indexed on the unique terms in the bi-grams is created, originally empty, and later populated with the proper PPMI value for each word combination. Once the matrix is completed a singular value decomposition (SVD) is applied using the *TruncatedSVD* function (*sklearn* Package), reducing the large original matrix. The new one for each row, representing a word, considers only the most significant 300 features. A later step is to consider only the first component of the decomposition, the U matrix; each row of this final matrix is a short vector representing a word.

4. Text Classification

A binary classification is implemented, the main aim is to deal with the presence of spoilers in books reviews. A classifier is trained on a subset of data to recognise and filter out spoiler reviews from non-spoiler ones. For this purpose the TF-IDF representation previously computed is employed. The first step consists in preforming a split of the data in a training set (66%) and a test set (33%). Then the imbalance nature of the dataset is tackled, a quick analysis reveals that the proportion of reviews containing spoilers on the grand total is pretty small. Proportion wise 94 % of the reviews available do not actually include a spoiler, particularly out of 4690 total reviews 4404 do not have a spoiler. This extreme situation leads to a very biased classification that does not consider the minority class. In order to mitigate this problem a random

oversampling technique is employed, targeting the minority class. The re-sampled dataset is more balanced than the original one, resulting in a 50 % - 50 % division between the classes. Its size results increased by the oversampling, presenting 8808 reviews, 4404 with and 4404 without spoilers. Having the proper set of data a Naive Bayes classifier is trained on the training set and tested on the test set. The Naive Bayes is chosen because it is a simple classifier that works reasonably well even with a small dataset. For its implementation the package *sklearn* is used.

The evaluation measures considered are the following:

- **Precision**:

$$\pi = \frac{TP}{TP + FP} \quad [4]$$

where TP = True positives, FP = False positives

- **Recall**:

$$\rho = \frac{TP}{TP + FN} \quad [5]$$

where TP = True Positives, FN = False negatives

- **F1**: harmonic mean of precision and recall

$$F1 = \frac{\pi\rho}{\pi + \rho} = \frac{2TP}{2TP + FP + FN} \quad [6]$$

- **Accuracy**:

$$F1 = \frac{TP + TN}{TP + TN + FP + FN} \quad [7]$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

```
Train Accuracy      0.97480
Test Accuracy       0.82597
Test Precision      0.05882
Test Recall         0.13636
Test F1             0.08219
Test F2             0.10791
Classification report:
              precision    recall  f1-score   support

       False    0.94311   0.86777   0.90387      2178
        True    0.05882   0.13636   0.08219       132

    accuracy                        0.82597      2310
   macro avg    0.50097   0.50207   0.49303      2310
weighted avg    0.89258   0.82597   0.85692      2310
```

Fig. 1. classification results

The figure 1 shows the values obtained for each evaluation metric. The overall test accuracy is pretty high (circa 80 %), but all other metrics are disappointing. This is due to the very nature of the data used. The original dataset is extremely imbalanced, an oversampling technique can mitigate this problem but can not remove the issue completely. Furthermore the sheer size of the dataset is way too small to train a proper classifier.

The confusion matrix in figure 2 confirms the poor performance of the classifier. If it manages to perform reasonably well classify the non spoiler reviews, 1890 of 2310, but fails frequently with the spoiler ones.

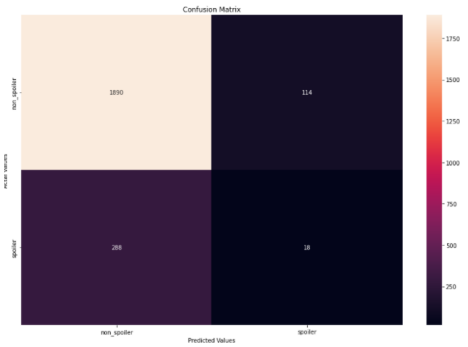

Fig. 2. confusion matrix

5. Topic Modeling

The task is addressed using the Latent Dirichlet Allocation (LDA) approach. This is an unsupervised machine learning technique, aimed at extract a list of topics with associated clusters of words from a document or a document collection. In particular the *Gensim* package allows to create a LDA model that computes both the distribution of words for each topic $K$and the distribution of topics for each document i. The number of topics is manually set to 5, the objective is to try and capture the nature of the most reviewed books in the dataset. Considering the computational requirements for this package a number of 10 passes on the corpus is selected to properly train the algorithm. An interactive visualization is provided, it includes both the inter-topic distance map and the bar chart for the 30 most salient terms and the overall frequency of each of them. In figure 3 a static visualization of said map is proposed. In particular the map shows five circles, one for each topic found, whose area is proportional to the amount of words in the entire dictionary belonging to that specific topic. In this case the topics seems to be well separated and the first three of them appear to have more words than the last two.

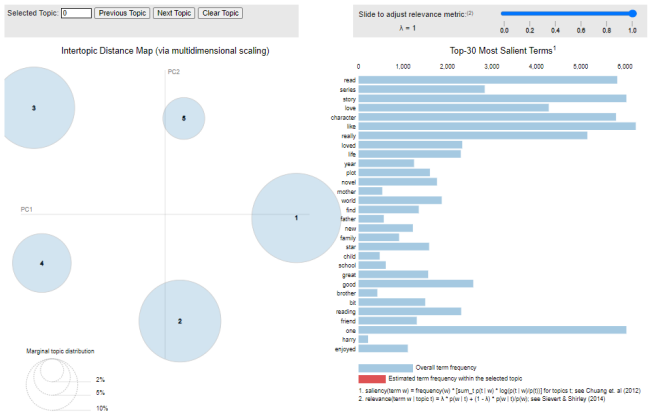

Fig. 3. topic modeling

Generally the interpretation of the topics is not that immediate, but it is more a game of suppositions. For this reason in order to properly evaluate the results two intrinsic metrics are computed:

- **Perplexity**: A measure of uncertainty, measures how well a probability model predicts a sample. Low values

are to be preferred being a sign of a better model. The perplexity value obtained is equal to -7.606.

- **Coherence**: A measure of semantic similarity between top words in a topic. It scores a single topic by measuring the degree of semantic similarity between high scoring words in that topic. Computation of this measure is based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses normalized point wise mutual information and the cosine similarity. Higher values of coherence indicate a better performance of the model. In this metric only an overall value of 0.303 is return.

As can be observed the overall performance of the model is not impressive. In particular the coherence score is very low. Generally changing the number of topics is a valid way to improve the total coherence of the model, but in this case this score keeps itself stead around the 0.28 to 0.35 threshold. This behaviour is probably due to the dissimilarity of the words in the reviews and their impossibility to be grouped efficiently in topics.

6. Conclusion

The overall results obtained by performing both the binary classification and the topic modeling are not satisfactory.
Two main reasons are behind this results: a lack of data and the nature of the data themselves. Computational limitations required a selection of a small dataset, this may hinder a proper training of most algorithms. On the other hand the extremely unbalanced nature of the data does not lend itself to a comfortable use, requiring specific sampling operation to mitigate the problem. The classification is a valid solution to create an anti-spam filter, but a proper implementation of the ideas presented in this paper requires a new and better dataset for training. Regarding topic modeling, the meaning of the topics produced by the LDA model is not clear. In this regard to the problems discussed above its added the difficulty to model short and different reviews in specific topics. The original objective to find the most discussed genres through this operation is therefore to be considered not accomplished.