Classification of Amazon reviews by topic

Raffaele Anselmo[1], Lorenzo Pastore[2]

Abstract
The aim of this paper is to explore the potential of text mining and machine learning techniques. Specifically, a multiclass text classification problem has been addressed. The primary objective was to develop the data processing flow in order to identify the crucial and most difficult phases in a generic implementation scenario.

Keywords
Text mining — Classification — Machine Learning — Data Science

[1] *Università degli studi di Milano-Bicocca, CdLM DataScience*
[2] *Università degli studi di Milano-Bicocca, CdLM DataScience*

Contents

Introduction

Data mining is an interdisciplinary subfield of computer science and statistics with the overall goal of extracting information from a dataset and transform it into a comprehensible structure for further use. [1] [2]

Text mining is a branch of data mining but with the substantial difference that patterns are extracted from natural language text rather than from structured databases. The potential of text mining was already highlighted by some academics. Back in 2004, Ian Witten defined it as "a burgeoning new field that attempts to glean meaningful information from natural language text". [3][4]

Although text mining immediately became the object of great interest in academia, it is only in recent years that text mining has attracted the interest of many private companies thanks to the increase in computational capabilities and the interdisciplinarity of the field. Among the many tasks related to text mining are information retrieval, text summarization, content based recommender systems, content clustering and text classification.

The aim of this paper is to explore one of the possible applications of text mining, namely, a multiclass text classification problem. The paper is divided into 6 sections:

- In the first section dataset is presented;
- The second section contains the preliminary analyses of the dataset that led to the definition of the appropriate normalization, tokenization, stopwords removal and stemming techniques;
- The third section presents the text representation models used;
- The fourth section presents the classification models and the preliminary dimensionality reduction techniques adopted;
- In the fifth section the models obtained were compared thanks to some validation techniques and measures of performance evaluation.

Lastly, the results achieved in the analysis are commented with focus on the most important aspects emerged from the survey.

1. Dataset

The analysis was conducted using the "Amazon reviews data (2018)", which is a collection of product reviews and metadata from Amazon.[5][6]

Within the collection, the data are divided by product category. Of the available categories, the following were selected to carry out this classification task:

1. **Digital Music:** 169,781 reviews
2. **All Beauty:** 5,269 reviews
3. **Software:** 12,805 reviews
4. **Gift Cards:** 2,972 reviews
5. **Luxury Beauty:** 34,278 reviews
6. **Appliances:** 2,277 reviews
7. **Magazine Subscription:** 2,375 reviews

For this task, instead of using the complete review data, a smaller subset consisting of 5-core reviews was used, so that each of the remaining users and items have 5 reviews each. Each dataset contains the following 10 variables:

1. **reviewerID:** ID of the reviewer
2. **asin:** ID of the product
3. **reviewerName:** name of the reviewer
4. **vote:** "helpful" votes of the review
5. **style:** a dictionary of the product metadata
6. **reviewText:** text of the review
7. **overall:** rating of the product
8. **summary:** summary of the review
9. **unixReviewTime:** time of the review (unix time)
10. **reviewTime:** time of the review (raw)

To avoid making the data analysis process too expensive in computational terms a sample of 2000 was used from each category.

2. Pre-processing

A new dataframe was created from the samples extracted from the different categories containing only the variable "reviewText". Therefore, henceforth we'll refer to the sample collection of the review from every category as "dataset".

Specifically, the following operations were carried out during the pre-processing phase.

2.1 Normalization

Through this process words were made uniform in terms of accents, capitalization, punctuation and typos that needed correcting. Initially everything but characters were removed from the dataset and the text was normalized to lower case. Moreover, we noticed a consistent number of reviews with repetition of the same character, i.e. "aaaaa". These repetitions could be both typos or used to add emphasis. However, given the classification problem, in order not to influence the algorithm, it was decided to remove all n-duplicate for $n>2$. At last the punctuation was removed from the text, which was ready for the next phase.

2.2 Tokenization

The goal of tokenization is to split the text into meaningful dense units. In this case to identify the tokens, a simple *Regular Expression* was used, dividing dense units via one or more white spaces.

2.3 Stopwords removal

This operation allows to obtain a cleaner dataset by removing the most common words. In this particular case, a precompiled list of English stop words from the python package *"Natural Language ToolKit"* was used.

2.4 Stemming

Stemming removes morphological affixes from words, leaving only the word stem. On one hand, the risk is to lose the precise meaning of the words; but on the other hand, there is a substantial gain in terms of memory required. The stemmer used on the dataset is the well known *Porter Stemming Algorithm*.

It is worth highlighting that stemming was conducted after the removal of the stop words. This was done to minimize the risk that some stop words would not be identified as part of the *NLTK* list using the inverse sequence of operations.

3. Text representation

In this phase, the first step was to decide between a series of models for the representation of the text in a simplified matrix form.

In the specific case of this analysis, the following two were taken into consideration:

- **Bag of words:** The bag-of-words model (BoW) is a simplified representation used in natural language processing and information retrieval. In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order. The bag-of-words model is commonly used in methods of document classification where the frequency of each word is used as a feature for training a classifier.
- **Tf-idf:** The "term frequency - inverse document frequency" model assumes that not all words in a text

describe the content with the same accuracy or informativity. As shown in figure 1, the ability of words to

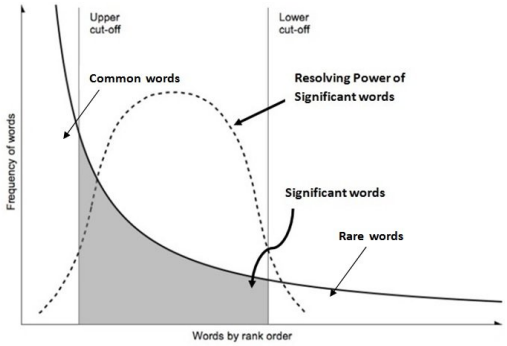

Figure 1. Luhn Analysis Schema

discriminate the content in a document is highest at the intermediate position between the two cut-off levels. The basic idea is to give weight to terms that represent a document. Based on Luhn's analysis, two factors were identified to assign weights to words:

Corpus-wise: This factor is measured by the *term frequency* heuristics and is defined as the number of times the term t occurs in document d.

Document-wise: This factor is measured by the *inverse document frequency*, based on the idea that the specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.

The two heuristics combine in the tf-idf weight defined as follow:

$$W_{t,d} = \frac{tf_{t,d}}{max_{ti}tf_{ti,d}} \times \log \frac{N}{df_t}$$

3.1 Bag of words

In this work the bag-of-words model was mainly used as an exploratory analytical tool of words frequency. Even though this model doesn't have a weighting system, it allowed us to evaluate the frequency distribution of terms in the text.

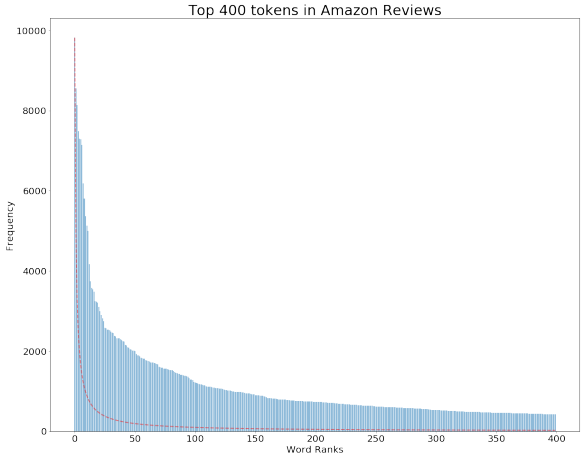

Figure 2. Amazon distribution

According to **Zip's law**, which describes the frequency of a word in a set according to its rank, the data follow a near Zipfian distribution, as shown in figure 2.

This makes explicit that "head words" are very recurrent but have no meaning, while "tail words" take the largest portion of the vocabulary but are rarely needed in documents.

Figure 3. Amazon Reviews wordcloud using BoW

3.2 Tf-idf

The tf-idf matrix was used to develop the text classification model. According to Luhn's analysis, indeed in this paper we assume that the frequency with which certain words appear in the text gives an important insight into the meaning of the words. The resulting matrix has a dimension of 14000×4852.

4. Text classification

This section presents the dataset size reduction techniques and the classification models that have been used.

4.1 Dimensionality reduction

Since the dimensions of the text representation matrix tf-idf are very large, it was decided to use two techniques to reduce the dataset dimensionality. The very first step was to operate a **feature selection** which was followed by a **features synthetization** that allowed to further reduce the numbers of features.

- **Chi-squared test:** is used to determine the relationship between the independent features and class attribute. In feature selection, we aim to select the features which are highly dependent on the response. [7]
- **Principal component analysis:** is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into

a set of values of linearly uncorrelated variables called principal components. [8]

In the development of the project pipeline, a feature selection was first made maintaining only 75% of the best features according to a chi squared test, enabling us to go from 4852 to 3639 features. Subsequently a PCA was carried out, which made it possible to further reduce the number of features from 3639 to 2000, while maintaining more than 90% of the total variance in the data.

4.2 Data modelling

Firstly, a set of Machine Learning algorithms was selected taking into account both the data structure and the objective of the analysis. Specifically, the classification models used refer to 3 distinct categories:

- **Separation models:** these attempt to map the data into a larger space, with the aim of finding the hyperplane that best separates the variables according to the variable of interest by using functions. Among these, a Support VectorMachine (SVM) was integrated using the *LinearSVC* classificator of Scikit-Learn library.
- **Regression models:** are very flexible and easy to understand as it is possible to measure the effect of the different variables in the classification thanks to the coefficients assigned by the model. In the analysis the *LogisticRegression* classifier of Scikit-Learn library was implemented.
- **Neural networks models:** they are black-box models that exploit a network of artificial neurons to stimulate complex functions. However, precisely because of their structure, they are difficult to interpret. Among these, the *MLPclassifier* by Scikit-Learn was implemented.

While the default settings of Scikit-Learn were used for the classifiers LinearSVC and LogistiRegression, for the MLPclassifier an "ad hoc" structure was defined. A neural network with **4 hidden layer** was defined. The first layer has 100 neurons while the other 3 have respectively 50 neurons each.

A **Rectified Linear Unit (ReLU)** activation function was used for all layers. The choice was due both to the efficiency in backpropagation of errors and the ability to make the network scattered limiting learning and saturation problems. The **Softmax** function was selected as the activation function for the output level. This produces as output a probability vector, in the same way as the Sigmoid, and is often used in classification problems preferably in the case of more than two classes.

The loss function to be minimized was the **Categorical Cross Entropy**, although other performance measures were considered including Accuracy, Precision, Recall and F-measure. As optimizer, we used **Adam** (RMSprop + Momentum), a gradient algorithm with adaptive learning rate, which helps to add a bias correction at different times.

5. Models evaluation

This section presents the procedures that have been carried out for the evaluation of the models, the measurements used to compare them and the results thanks to some known graphic techniques.

5.1 Hold-out and cross-validation

After pre-processing, the dataset was divided with the *hold-out* method into two partitions, equal respectively to 90% - training set - and 10% - test set - of the initial dataset. Thanks to the training set, the selected classification models were cross-validated and compared, while the best model detailed evaluation was subsequently possible through the evaluation of the performance on the test set.

A second partition was made to obtain a validation of the classifiers using an approach called K-fold cross validation. This technique divides the dataset into K-partitions of equal number and ensures that all records are used at least once in both the training set and the validation set. At each step, K-1 partitions become part of the training set, on which the classification algorithm is trained, while the remaining k-th partition is used as validation set, a block of observations of which the model predicts the value of the class attribute. It was decided to attribute the number of iterations of Cross Validation to K=5.

5.2 Performance evaluation measures

Usually the selection of evaluation measures is made starting from the structure of the problem and taking into account the objective of the specific application. Since in this study case it was not necessary to give particular penalties to false positives or false negatives, the 3 models were compared in terms of **Accuracy**.

For the best resulting model, the measures of Precision, Recall and F1-score were also calculated.

5.3 Models comparison

In order to identify the best classifier, a 5-fold cross-validation in terms of accuracy was calculated on each model. The measurements are summarized in the table in Figure 4.

	Mean Accuracy	Standard Deviation	Training time	Training costs
LinearSVC	0.8657	0.0058	46 sec	3 min 59 sec
LogisticRegression	0.8597	0.0091	1 min 5 sec	4 min 18 sec
MLPclassifier	0.8328	0.0079	3 min 36 sec	8 min 49 sec

Figure 4. Cross-validation models Accuracy

The best results are obtained by the *Linear SVC*, not only in terms of higher average accuracy value than other models, but especially of computational cost.

To obtain a more comprehensive graphical representation, the results have been displayed by means of some boxplots. Figure 5 shows the boxplots of each classifier's performance accuracy on a 5-fold crossvalidation.

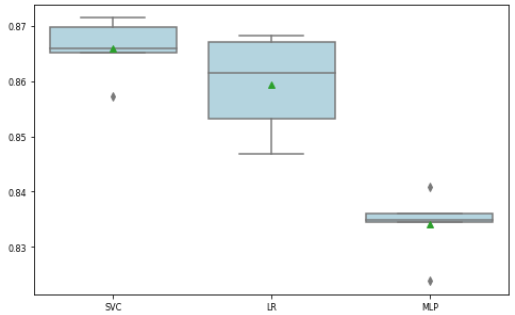

Figure 5. Box-plot of 5-fold cross-validation

In order to investigate this classifier in more detail, its performance on the test set was evaluated taking into consideration other measures derived from the confusion matrix (see Fig.6).

	Precision	Recall	F1-Score
Digital Music	0.80	0.78	0.79
All Beauty	1.00	0.97	0.98
Software	0.86	0.87	0.86
Gift Cards	0.75	0.88	0.81
Luxury Beauty	0.84	0.82	0.83
Appliances	0.92	0.88	0.90
Magazine Subscrition	0.96	0.88	0.92
Model Accuracy	0.87		
Classification cost	1 sec		

Figure 6. Linear SVC test set performance

Both accuracy and recall are based on an understanding and a measure of relevance of true and false positives, while the F1-score allows the relationship between them to be taken into account by making it the harmonic average.

6. Conclusions

The aim of this paper was to highlight the potential of text mining and machine learning techniques. Specifically, a multiclass classification problem was addressed. The primary objective was to develop the data processing flow in order to identify which could be the crucial and most difficult phases in a generic implementation process.

While the development of the classification models was relatively simple, one of the major obstacles was to handle high computational costs with respect to the definition of the most appropriate dataset dimension.

The main problem that this study attempted to address was the reduction in dimensionality to obtain better performance from the classification algorithms. In particular, a feature selection and PCA was carried out allowing to reduce the number from 4852 features to 2000 main components, maintaining more than 90% of data variability.

Another difficulty was to limit the number of observations for each category because beyond 30000 observations the computational capabilities at the ram level (local and gdrive) would be exceeded.

In conclusion, although the results may be satisfactory, computational costs remain a major hurdle for making models scalable. Possible future developments could be addressed by considering other techniques or variations of the same PCA, such as the categorical PCA.

References

[1] Christopher Clifton. Data mining, Sep 2017.

[2] Trevor Hastie, Jerome Friedman, and Robert Tisbshirani. *The Elements of statistical learning: data mining, inference, and prediction*. Springer, 2017.

[3] Marti A. Hearst. Untangling text data mining. In *University of Maryland*, pages 3–10, 1999.

[4] Ian H. Witten. Text mining. In *in a Digital Library. International Journal on Digital Libraries archive*.

[5] Amazon review data (2018).

[6] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics.

[7] Sampath Kumar Gajawada. Chi-square test for feature selection in machine learning, 2019.

[8] Wikipedia contributors. Principal component analysis — Wikipedia, the free encyclopedia.