MATH637 – Mathematical Techniques for Data Science

Project 1 Report: Sentiment Analysis on TripAdvisor Reviews

Giuliamaria Menara, 702524795

Data exploration

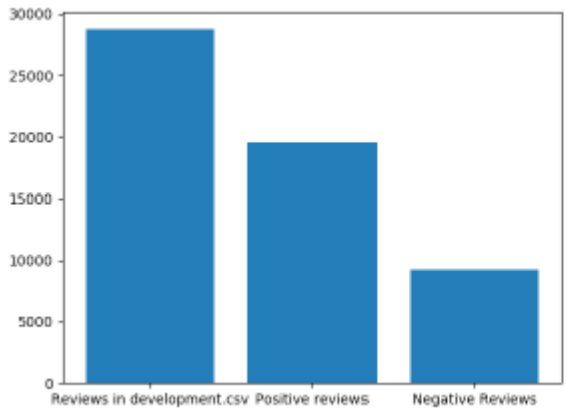
This report describes the ides used to carry out a text sentiment analysis on reviews collected in the Italian TripAdvisor website.
The first step of the analysis consisted of importing data from `development.csv` and `evaluation.csv` file.

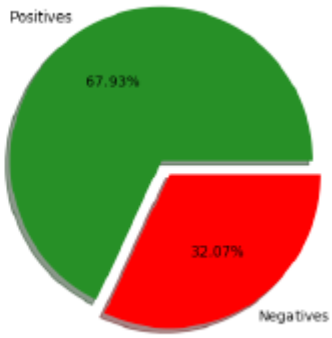
By reading the files, it was possible to notice that the **41077** textual reviews contained into the dataset are split as follows:

- **28754** in `development.csv`
- **12323** in `evaluation.csv`

In order to build a good classification model, the focus was directed toward the content of `development.csv`. Analyzing this file better, it was possible to extract other important statistics like:

- The number of positive reviews: **19532** (67.93%)
- The number of negative reviews: **9222** (32.07%)





As shown in the plots, classes are not represented equally, so this dataset is quite imbalanced, but in this preliminary phase neither *undersample* to majority class nor *oversample* technique to minority class have been applied.

Some other statistics have been computed by processing the content of each review in `development.csv`: it was observed that the overall number of words is over **20** million and **153276** of them are different.

From these results we can see that the starting amount of data is huge, and so it is necessary to apply an efficient preprocessing phase before proceeding to evaluate a classification model.

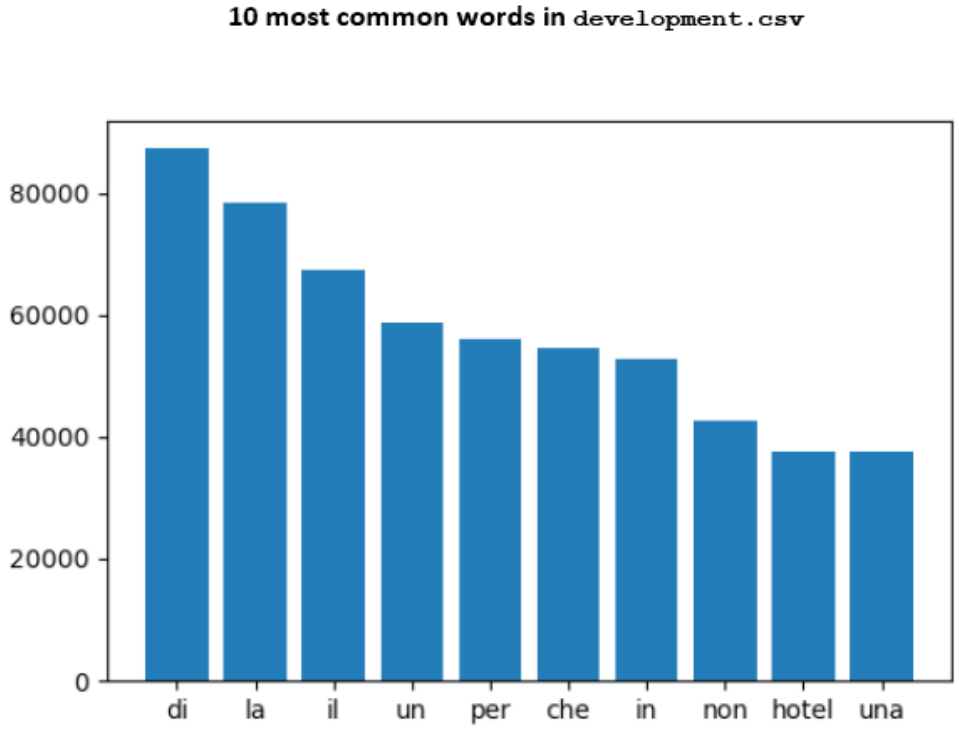
Preprocessing

Document processing

This step consisted of sequential operations that have allowed us to generate a structured data representation.
Firstly, the following operations are performed on each textual review:

- Removal of all numbers;
- Removal of all the foreign characters and words;
- Removal of all the special characters;
- Removal of punctuation and newlines;
- Removal of all single characters;
- Removal of whitespaces;
- Conversion of all words to lowercase (Case normalization).

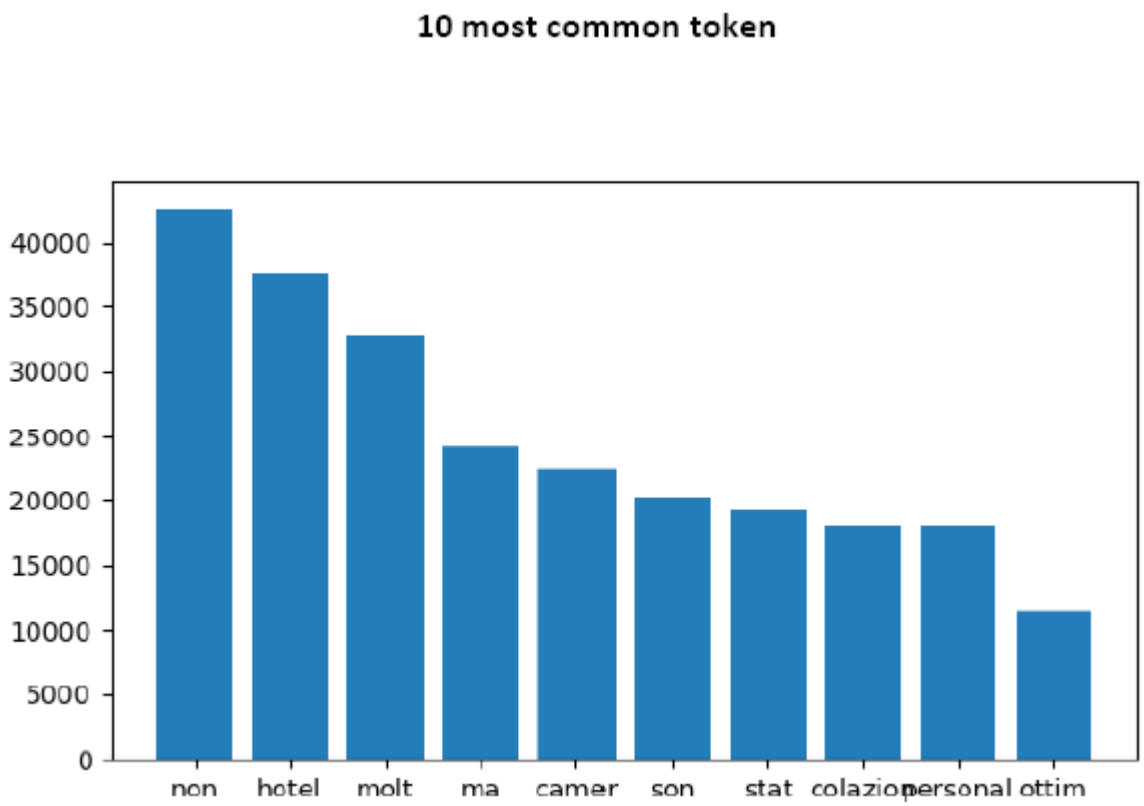
After these operations, the number of different words in `development.csv` was **52890** (about a third of the original set).



As it can be seen from graph, there are still useless terms (articles, prepositions and the word “hotel”). Since most common words in `development.csv` are articles and prepositions that certainly are not be able to provide useful indications about a negative or positive feeling that could direct correctly the sentiment analysis, a

preprocessing phase improvement is performed by ***document splitting***, ***tokenization*** and ***stemming***.
In order to implement these, some *Natural Language Toolkit* (*NLTK*) libraries were imported:

- **Nltk.tokenize [2]**: this suite provides the *word_tokenize* method, that returns a tokenized copy of the text using *NLTK's* recommended word tokenizer;
- **Nltk.stem.snowball [3]**: this suite provides us with the *ItalianStemmer* method, which reduces Italian words to their root form, removing prefixes, suffixes and pluralization;
- **Nltk.corpus [4]**: this library contains a method that provides us with a list of stopwords from many languages. From this default list of words, we removed some useless items:
    - "non", an adverb meaning "not" (indeed it is the 5th word most used in negative reviews);
    - "ma", a conjunction meaning "but";
    - "sono", Italian for "I am";
    - "contro", Italian for "cons".



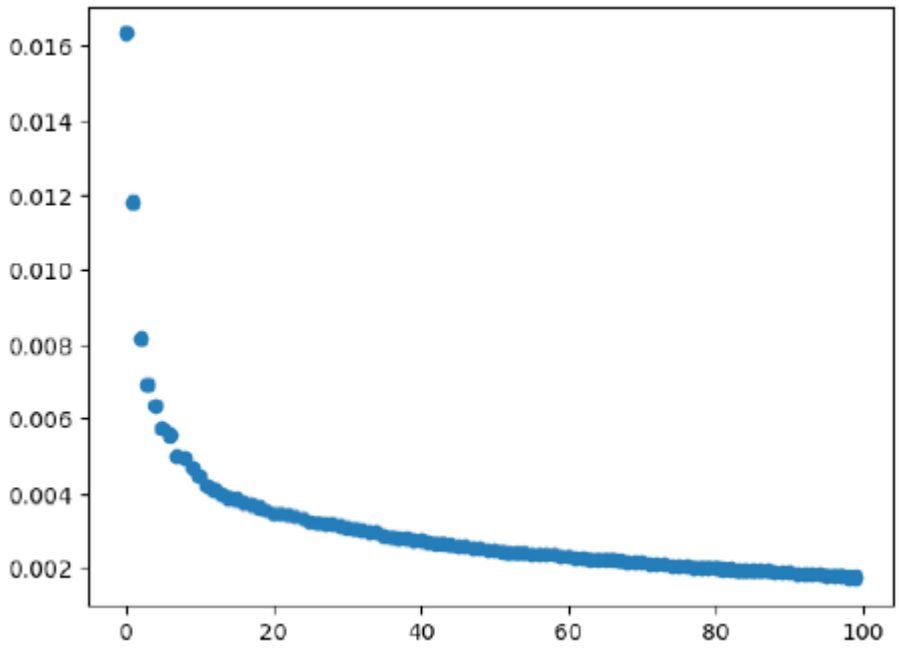
After the described operations, the total number of different tokens was equal to **27207** (less than a fifth of the original set).

Tf-idf Vectorizer

Another important step consists of calculating the ***tf-idf*** (*term frequency–inverse document frequency*) parameter through the *TfidfVectorizer* function from *sklearn* library. This weighting scheme measured the importance of each term with respect to the dataset.

Data reduction

In order to speed up the research of best Classification model it was fundamental to use *Principal Component Analysis* (***PCA***) to reduce the size of attributes. In this project I adopted the specific case of *Incremental principal component analysis* (***IPCA***), a technique used when the dataset to be decomposed is too large to fit in the memory.



From this graph, it is possible to see that the suitable number of kept components after applying ***IPCA*** is 10.

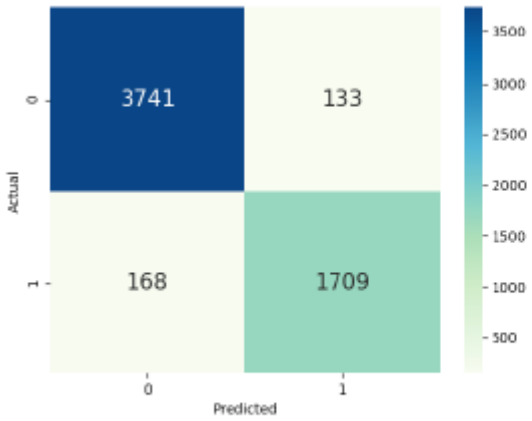
Algorithm choice

Starting from results obtained from the previous steps, it was possible to build a robust model able to predict the belonging class of data coming from an evaluation set.
In order to find a solution, we have to choose a classification techniques able of handling a linear text processing problem like this.
Basing on this premise, we consider the following classification algorithms:

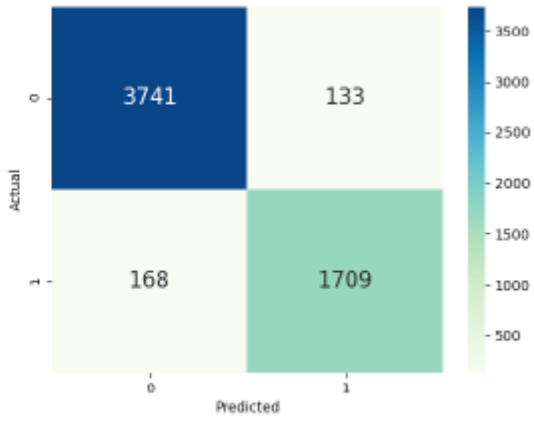
- Decision Tree Classifier
- Random Forest Classifier
- Stochastic Gradient Descent (*SGDClassifier)*
- Support Vector Machine (*LinearSVC Classifier).*

The decision of including SGDClassifier is dictated by its simple and efficient approach to solve linear text classification problems like this.

The criterion used to choose the best model is the *model accuracy*. According to the obtained results, it was possible to notice that most accurate models were ***LinearSVC*** and ***SGDClassifier***, that provided the following confusion matrices.



LinearSVC confusion matrix



SGDClassifier confusion matrix

References

[1] Dataset from tripadvisor.it Italian web site (https://www.tripadvisor.it/)
[2] NLTK.tokenizer (https://www.nltk.org/api/nltk.tokenize.html)
[3] NLTK.stem.snowball Italian Stemmer (https://www.nltk.org/api/nltk.stem.html#nltk.stem.snowball.ItalianStemmer)
[4] Nltk.corpus stopwords (https://www.nltk.org/api/nltk.corpus.html)
[5] Incremental PCA (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html)
[6] sklearn.linear_model.SGDClassifier(https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html)
[7] LinearSVC (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html)