Data Science Lab: Process and methods

Politecnico di Torino

Project report

Student ID: s276525

Exam session: Winter 2020

1. Data exploration

The exploration of the data is based on the use of Pandas library. Both csv files are read and explored with apposite functions obtained from this library.
The first feature considered on this exploration is the description of both datasets, in order to have an idea on which kind of data are present inside.
From this initial analysis is visible that in the first given dataset, the "Development" one, are present two columns composed by 28754 elements. The "text" column contains 28754 unique reviews while the "class" one is based only on two different values: "pos" and "neg" that enucleate the sentiment of each review.
Unlikely from the "Development" database, in the "Evaluation" dataset is present only the "text" column, composed by 12323 unique reviews.
After the description of each database, is important to check the completeness of them by using a specific function in order to control the possible presence of null values that could alter the final result. In both cases, with no presence of missing values, there is no need to intervene and modify the dataset.
Another necessary exploration, useful to have a better idea of the "Development" database, is characterized by the count of the elements "pos"-" neg" contained inside the "class" column obtained through a function applied to the fancy-indexing of that given column . This gives an idea on how the first dataset is distributed and, thanks to a related matplotlib function, it is visible in the graph below.

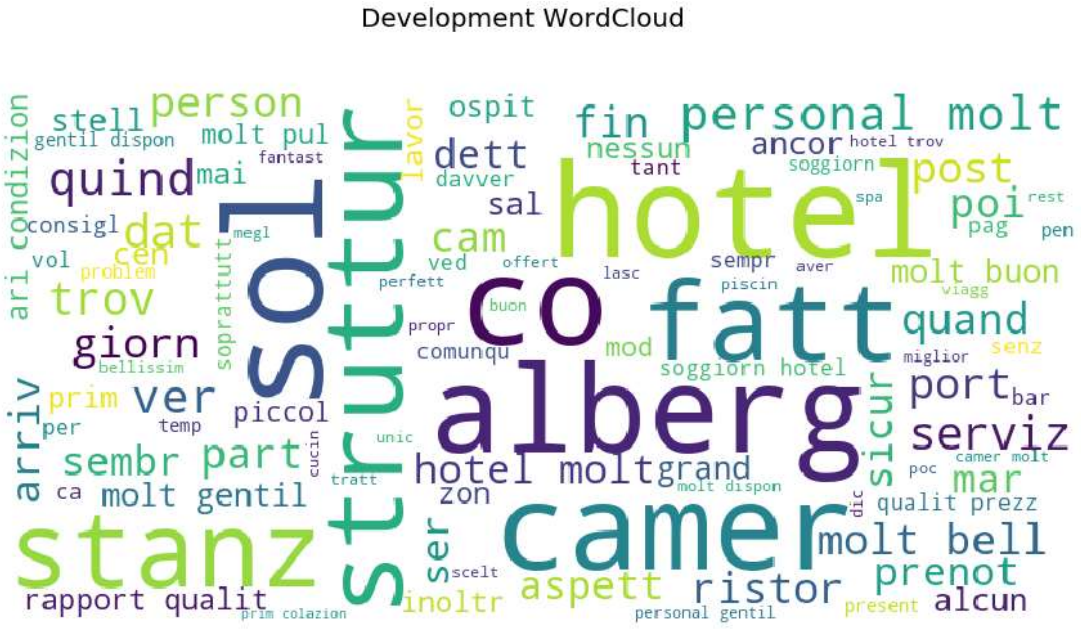
According to the result obtained it is visible a preponderance of positive reviews inside the database, more than double of the negative ones.
At this point the decision taken was to not balance the dataset in order to have the most possible number of words useful to understand better the sentiment analysis.
The last key point of the Data Exploration part is represented by the WordCloud visualization of the initial datasets, still without any kind of preprocessing, just to have an idea of the starting point.

Development WordCloud

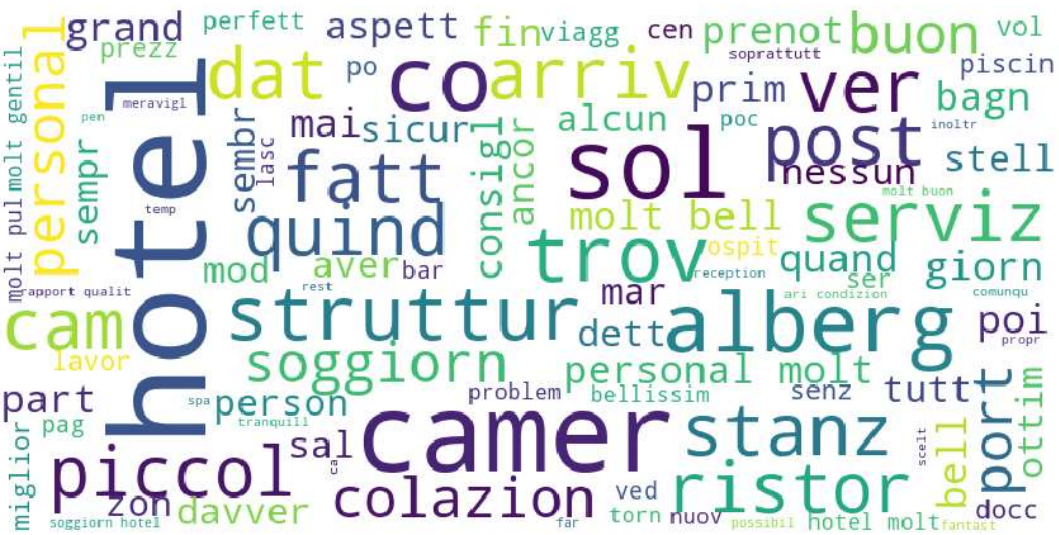
Evaluation WordCloud

2. Preprocessing

From the paragraph before it is already known that no null values are present in each database. Thanks to this knowledge, it is possible to concentrate the pre-processing phase on the treatment of the Italian words contained inside the reviews, in order to create a base model useful for the sentiment analysis of the “Evaluation” database.
All the functions used in this section are used on both the “text” columns of the given datasets, in order to improve a more accurate method of classification.
The first pre-processing step, the case normalization one, is based on a function that exploits the regular expression library “re” for the substitution of all not alphanumeric elements (with spaces in some cases, without in others) and for moving all words to the lower case.
The second approach is dedicated to the removal of all the Italian stop words(plus some not useful words added manually) , obtained from the “nltk.corpus” package, in order to have a more specific set of words through which establish the sentiment of each review. This because the most used words are usually present in both positive and negative reviews and so can’t be considered useful to the purpose in question.
For the successive step was necessary a choice between stemming and lemmatization; considering the higher speed and also a more accurate language availability, the first technique was chosen.
To adopt the stemming was useful the usage of a package never introduced in lesson but very adapt to this exam purpose: the Snowball Stemmer [1]. In fact, this package guarantees the reduction of a word to its root form according to the Italian language.
The results obtained are visible in the following WordCloud visualizations, obtained with a specific function created inside the code.



Evaluation WordCloud



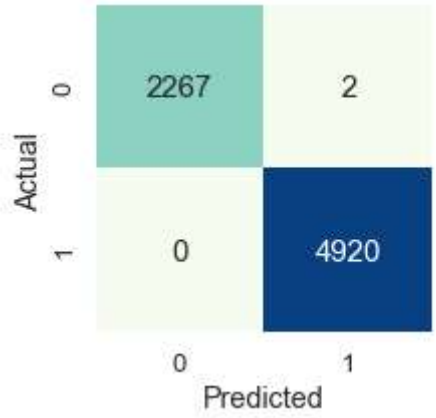
The last step is based on the extraction of the features from the reviews. This is done by applying a Term frequency inverse document frequency (tf-idf) algorithm to the set in use. The tf-idf algorithm is implemented by the sklearn library and has argument specified: the Italian stop words list, indicating the common terms to be ignored; the n_gram(1,2) to extract both unigrams and bigrams.

3. Algorithm choice

The choice was based on the research of the most accurate classification algorithm for the sentiment analysis problem.
Each review obtained from pre-processing is a sparse vector with a slot for every unique n-gram in the corpus. On datasets represented in a sparse way like in this case Linear Classifiers typically perform better than other algorithms.
As a first step, through the use of the ”train_test_split” function of the sklearn library, the “Development” database has been split in training and test set. In order to have more than the 2/3 of the database inside the training set, its size has been put equal to 0,75.
The classification algorithms tried to solve this problem have been: Decision Tree Classifier, K-Nearest Neighbors, and finally the Linear Support Vector Classification [2].
The Decision Tree Classifier is a Supervised Machine Learning model where the data is continuously split according to a certain parameter.
K-Nearest Neighbors is a simple algorithm that classifies a data point based on how its neighbours are classified.
The Linear Support Vector Classification constructs a maximum-margin separating hyperplane between data classes in an n-dimensional space (n=2 in this case). The goal of this separating hyperplane is to place all reviews (or as many as possible, given some tolerance) of positive class on one side of the hyperplane and then all reviews of negative class in the other side of the hyperplane.
The decision on the algorithm to use was based on the best evaluation metric obtained from the weighted f1_score function.
As a result, the final algorithm selected and applied to the “Evaluation” database was the Linear Support Vector Classification, obtained from Support Vector Machine package, thanks to a f1_score always higher than the others for at least 9%. In fact, the Linear Support Vector Classification gave a f1_score near to 96,5%, while the others didn’t even overcome the 87%.
The sentiment prediction obtained with the Linear Support Vector Classification, after a little phase of tuning and validation, has been submitted to the final csv file.

4. Tuning and validation

The tuning of Linear Support Vector Classification, the algorithm chosen to solve the problem, is characterized by the research of the best f1_score based on the variation of the regularization parameter C.
The for cycle created to evaluate the best conditions has declared in the majority of trials the C=1 as the most accurate parameter to use in this given case.
The accuracy of the final algorithm chosen has been evaluated also true the following confusion matrix obtained with the Seaborn library.



According to this confusion matrix, the obtained result can be considered quite optimal because only the True Positive and the True Negative cells contain an important quantity of elements, while the cells that should contain the errors are practically empty.
The last phase of the code, the final validation, is characterized by the creation of the "Result" csv file that has been evaluated for the competition.
The first validation step is characterized by the reading of all the "text" rows contained inside the "Evaluation" dataset. All the reviews are named with a progressive number, indicating their id.
Finally, in the last step, through the use of a dedicated function, the final result containing a column for the Id and a second column for the predicted values (obtained previously by the code) has been written on the "Result" csv file evaluated in the competition.

5. References

- [1] Snowball Stemmer URL: https://www.nltk.org/api/nltk.stem.html
- [2] Linear Support Vector Classification URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html