Data Science Lab: Process and methods

Politecnico di Torino

Project report

Student: Roberto Franceschi

Exam session: Winter 2020

1. Data exploration

The following report explains the pipeline used to achieve the result obtained on the competition platform. After having imported the data of both files (`development.csv` and `evaluation.csv`) into the dataframes, basic descriptive statistics was run about negative and positive class. The results showed the presence of 19532 (67.93%) positive reviews and the remaining 9222 (32.07%) negatives. Due to the unbalanced nature of the dataset, in the training phase, a cross-validation approach was performed to limit *overfitting*.

In the preliminary investigation of the dataset the presence of emojis, links (URL), special characters and foreign terms has been noticed in different reviews by visually inspecting some samples. The presence of some reviews (partially or entirely) written in non-Italian language has been better examined, using the *language detection library* [1].
However, since the number of reviews in a foreign language was limited compared to the datasets' size, these were removed from the first one. In the following table are reported the number of reviews per language found in both datasets:

Development set		***Evaluation set***	
language	text	language	text
it	28747	it	12319
en	3	en	2
nl	1	cs	1
ja	1	ca	1
fr	1		
es	1		

Table 1: *number of reviews per language*

The `.describe()` function of Pandas library was used in order to provide a descriptive statistics summarization related to the length of the reviews as showed in Table 2. Outliers were not highlighted based on the length in terms of words and characters, and the dataset analyzed showed that negative reviews were generally greater in length compared to the positive ones. In details the average number of characters were 624.57 and 864.23 for positive and negative reviews, respectively.

character		***Word***	
mean	701.43	mean	113.55
std	606.39	std	100.68
min	58.00	min	5.00
25%	325.00	25%	51.00
50%	515.00	50%	83.00
75%	850.75	75%	138.00
max	9153.00	max	1546.00

Table 2: *statistic summary about reviews length*

Lastly, it was investigated the *document frequency* aiming to define the association of a given word (unigram and bigram) to a positive or negative class. (i.e. if the word was used more frequently in positive reviews than negative, and vice versa). The top 20 unigrams and bigrams are reported in Figure 1, 2, 3.

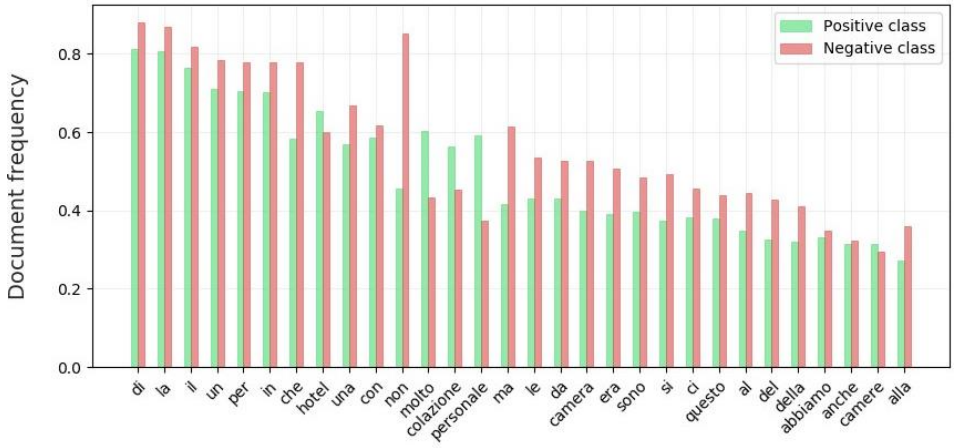

Figure 1: *Top 20 unigrams (before filtering stopwords)*

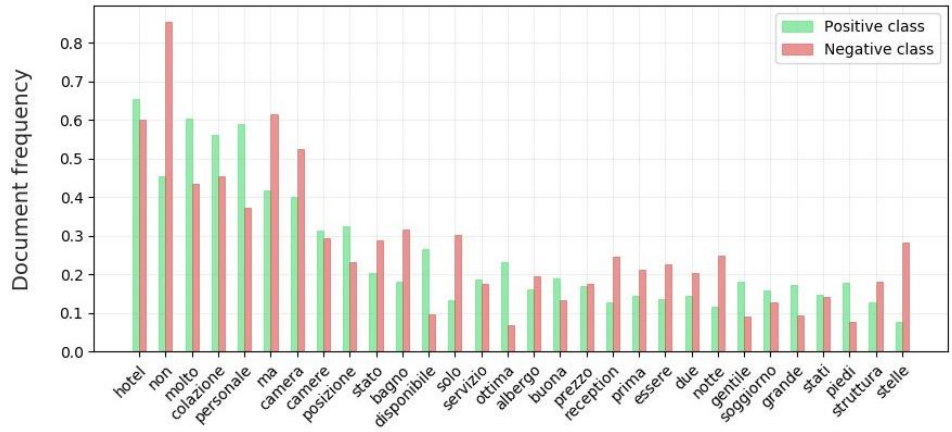

Figure 2: *Top 20 unigrams (after filtering stopwords)*

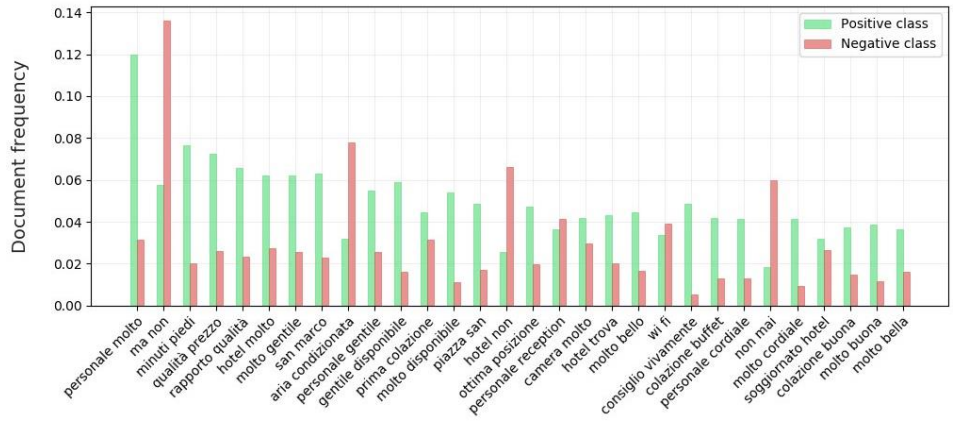

Figure 3: *Top 20 bigrams (after filtering stopwords)*

2. Preprocessing

In this section are described the preprocessing steps applied to make the data more suitable for data mining. Different strategies have been adopted to turn raw input into significant data. Following is listed a set of operations that have been carried out over the data cleaning process:

1. Removal/Exclusion of non-Italian reviews;
2. Substitution of emojis with a default string (the meaning of emojis was derived by studying the document frequency separately for positive and negative reviews);
3. Removal of URL links (e.g. http://ow.ly/oPtVn);
4. Removal of all special characters (including punctation and numbers);
5. Removal of all single characters and multiple whitespaces;
6. Conversion to lower case: all the letters in the sentences are converted into lower case;
7. Construction of n-grams: set of n-grams out of consecutive words.

The collected dataset was used to extract features needed to train the classifier. After these operations more accurate researches were run on the most frequent terms and consequently on the most important features (inspecting the dictionary returned by CountVectorizer).

Tokenization and Stemming
Before classification experiments, tokens (i.e. words) in the dataset were pre-processed using a Stemmer. For this purpose, an Italian Stemmer from `ntlk` was implemented [2]. Then a TF-IDF transformer was applied to make data suitable for classification phase.

Stopwords
In the experiments stopwords list was implemented excluding some words (like 'ma' and 'non'), due to the fact of being ‘fairly good’ indicators of negative class as suggested from the data displayed in Figure 1-2-3. The list of stopwords applied was the Italian one of the Natural Language Toolkit.

Data reduction
Data reduction phase consisted of a principal component analysis adopted in its variant IPCA [3] given the dataset’s size. In this view, the IPCA allowed the creation of an approximation for the input data using a quantity of memory that is independent of the number of features.

Normalization
During the normalization phase was implemented the MaxAbsScaler [4]. It allows to work with sparse data and consequently the absolute values were mapped in the range [0,1]. Other normalizers such as MinMax and Standard were not implemented because of the need of dense data.

3. Algorithm choice

After feature extraction a Principal Component Analysis was used in the pipeline to choose the best classifier. Indeed, PCA has the advantage of limiting the number of components and consequently a lower computational time was required (but at the same time decrease the accuracy). Once the best model was selected, the prediction was made on the evaluation dataset.

The final step in the text classification framework was to train a classifier using the features created in the previous steps. The following classifiers were implemented for this purpose:

- Random Forest
- Naïve Bayes Classifier (MultinomialNB [5], ComplementNB [6])
- Support Vector Machine (Linear SVC Classifier [7])

In order to limit the effect of the unbalanced dataset (i.e. overfitting) for the training and validation phase a cross-validation approach was used, specifically with the implementation of Scikit-learn’s `KFold` class. The approach divides all the samples in groups of subsamples and consequently the prediction function was learned using k-1 folds and the fold left out was used for test. For this project the number of folds was set to 10.

The selection of the best model was done experimentally by using the different parameters, such as the number of features, the choice of tokenization (unigrams and bigrams), the use of a normalizer etc.

The following table shows the averages of the weighted f1-score obtained on the development dataset (please note that the corresponding scores in the platform were slightly higher).

	LinearSVC	*Random Forest*	*Naïve Bayes classifier*	
			MultinomialNB	*ComplementNB*
Only preprocessing	0.9651	0.8878	**0.9334**	**0.9336**
Stemmer	0.9678	0.9012	0.9307	0.9312
No stopwords	0.9655	0.9032	0.9239	0.9242
Stemmer + No stopwords + emoji replacement	**0.9695**	**0.9135**	0.9276	0.9297

Table 3: *classifiers comparison by mean of f1-score obtained with cross-validation on the development dataset by using different pre-processing techniques*

The results obtained with different machine learning methods highlighted **Linear SVC** as the best method and that the algorithm outperforms with stemmer, stopwords and emoji filtering. The other classifiers were Multinomial (and its variant the Complement) which performed better without any kind of additional preprocessing. Random Forest instead had a behavior similar to the SMV, indeed showed better results for the configuration with both stemmer and stopwords removal. The main drawback of this last approach was the computational time with respect to the other classifier examined.

4. Tuning and validation

The introduction of stopwords removal decreased the sparseness of the data for machine learning techniques but did not improve the model (as underlined in section 3). On the contrary when a small number of significant stopwords were allowed by removing them from the original `nltk` list overall it improved.
The use of a stemmer and bigrams together with unigrams positively affect the results, but on the other hand it highly increased the time taken during the preprocessing phase and the number of final features.
The cleaning and substitution of emoticons proved as well significant changes in the quality of the prediction.

Best configuration found:

- minimum word's lenght: 1 character
- stopwords removal: yes (but some significant have been kept as explained before)
- stemming: yes
- emoji substitution: yes
- ngram_range: unigrams & bigrams
- max_df: 0.5
- min_df: 2

About the LinearSVC different parameters were tested with parameter search. The best configuration found is the following:

- Penalty: 'l2' (norm used in the penalization)
- Loss function: 'hinge'
- Dual = True (n_features > n_samples)
- Regularization parameter (C) : 2
- Tolerance (tol): 1e-06

From the confusion matrix (in figure 4), there is no particular concern about the validity of the classifier. The f1 score suggests that the classifier has no particular problems predicting the two classes on the development set.

Overall the best accuracy 0.9694 was obtained using Linear SVM with stemmer and token unigrams plus bigrams as features and emoticons replacement as pre-processing technique. The main related issue was that with SVM was not performed aggressive feature selection.

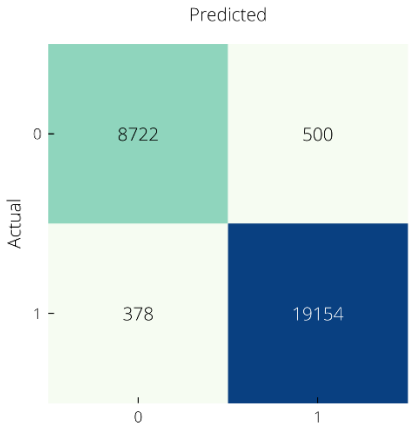

***Figure 4**: Confusion matrix*

5. References

[1] *Language Detect Library* (https://pypi.org/project/langdetect/)

[2] *NLTK Italian Stemmer* (https://www.nltk.org/api/nltk.stem.html#nltk.stem.snowball.ItalianStemmer)

[3] *IPCA - Incremental PCA* (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html)

[4] *MaxAbsScaler for Normalization* (htttps://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScaler.html)

[5] *Naïve Bayes Multinomial (*(https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html)

[6] *Naïve Bayes Complement* (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB)

[7] *Linear SVC (SVM)* (https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html)