Data Science Lab: Process and methods

Politecnico di Torino

Project report

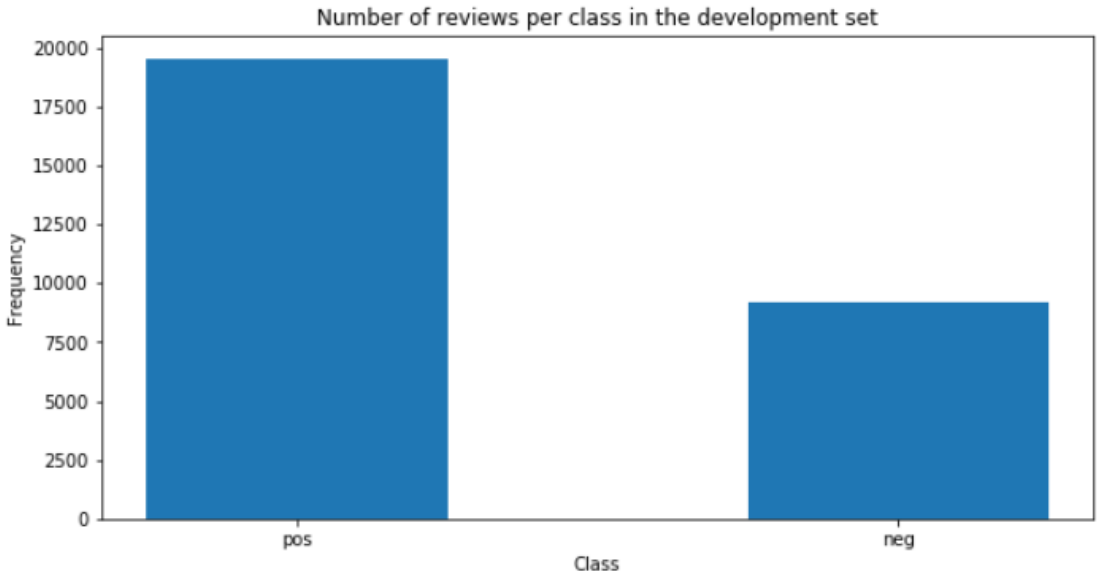
Bianca Iacomussi Student ID: s277857

Exam session: Winter 2020

1. Data exploration (max. 400 words)

A dataset of 41077 textual reviews written in the Italian language was given from the "tripadvisor.it" Italian web site. Those reviews have to be classified through a sentiment analysis to define them into positive and negative. The two files "development.csv" and "evaluation.csv", which contain the training and the test set for the classification, have been read and saved in two DataFrames using the Pandas' function read_csv (with encoding "UTF-8"). In the development DataFrame there are 28754 reviews each one labelled with its membership class "pos" or "neg", to indicate a positive or negative sentiment. So, development DataFrame has a shape (28754, 2): the first column corresponds to the review text and the second column corresponds to the class.

As shown in the picture below, in the development set, the positive and negative classes are not balanced. In fact, there are much more positive reviews (67.93%) than negative (32.07%). This may have an impact on the classification task. We would like to have the same size among the two classes, since differently, by splitting the original label distribution into the training and validation set, the result of the classification gets less trustable.



The evaluation DataFrame has only one column for the review text and it contains 12323 reviews. Although in both datasets there are not missing values, some reviews are not helpful since look like having no meaning. This may due to a poor automatic translation.
Some words, which you can read in these reviews, such as “ottimo”, “pessimo” and so on, allows almost directly to clearly distinguish the opinion expressed. So, these ones will be considered keywords. While other words (e.g. “non”, “mai”) show an explicit sentiment meaning only by associating them with other words and still depending on the context.
The sentiment detection may be further improved using some elements which are not words. For example, exclamation marks and suspension points, which emphasize an opinion in both directions, can make label assignments easier. Furthermore, the presence of happy or sad emoticons in some reviews can be helpful in order to detect positive or negative impressions.

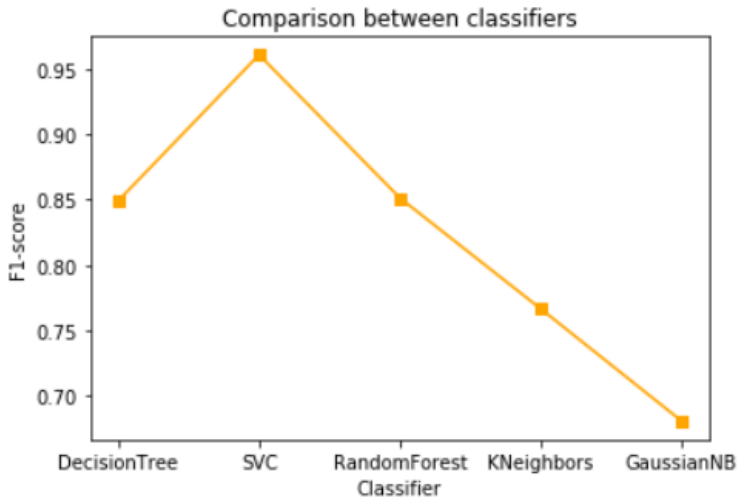
2. Preprocessing (max. 400 words)

In the pre-processing phase, text data has been cleaned from noise and a TF-IDF sparse matrix has been constructed from all reviews.
In order to clean data, a class StemTokenizer has been constructed. Using its method, it was possible to generate cleaned reviews. A combination of tokenization, stemming, punctuation removal and filtering has been adopted.
Some special tools have been used: 1) SnowballStemmer from the library nltk, which is one of the stemmers available in the Italian language and transforms inflected words into their root by cutting the ending 2) TweetTokenizer from nltk in order to not separate emoticons and to keep them in the analysis. Tokens after stemming where filtered and converted to lower case, so in the end cleaned text only contains lowercase alphabetical words with more than 3 letters[1], emoticons, exclamation marks and suspension points for the reasons explained before.
Then, the cleaned text has been used to construct the term frequency–inverse document frequency (TF-IDF) sparse matrix by a TfidfVectorizer. In this duty, stop words, which were not stemmed in previous passages, were removed. Furthermore, terms that appear in less than 5 documents were ignored (min_df=5). This has shown to be effective, because very rare terms can be seen as outliers and wrongly impact on the detection of sentiment.
The generation of a TF-IDF is adopted since this method allows to evaluate how much a word is important to a review in a collection of reviews. TF-IDF is even a way to perform feature extraction, so that the classifier will be able to make class assignations.
Since the classification is required only on the evaluation set from the “evaluation.csv” file and not on other possible datasets, the TD-IDF matrix was computed on the whole dataset, integrating the development and the evaluation too.
Then, the TF-IDF has been divided into training and test set, which both have a high number of features (9949 features). To better deal with this issue, a dimension reduction was performed by means of the Singular Value Decomposition (SVD) using TruncatedSVD. To achieve a good compromise between quality and lower computational time for the classification task, SVD was made taking the minimum number of components which could explain the 80% of total variance.

[1] In the final implementation no upper bounds for words length were used, because the longest words, which were mainly superlatives, seemed helpful for sentiment analysis.

3. Algorithm choice (max. 400 words)

In order to choose the best algorithm for the classification task, hold-out technique was used, by splitting the pre-processed development set into training and validation set. In this operation the training set has got a size of 80% of the initial set while the remaining 20% corresponds to the validation set. To overcome the problem of unbalanced labels in the development set, a stratification has been adopted, so that it is possible to preserve the original label distribution.
After that, some trials were made with different classifiers[2]. To evaluate the goodness of such classifiers the value of weighted F1-score has been adopted as a measure. As shown in the graph below, the Support Vector Classification (SVC) obtained the best score, which is quite better than other results. For this reason, SVC has been considered as the classification algorithm for the task.



To confirm the results just got and to decide which ones are the best hyperparameters for the SVC, a grid search was performed. After that, it was possible to select the two best configurations for the algorithm, which were used in the two final implementations. Both configurations use a radial basis function kernel (RBF) and gamma="scale". They differ, instead, for the C value, which is 50 for the first one and 1 for the second one.
The kernel RBF function can be expressed by the formula $K(x, x') = exp(-\gamma\|x - x'\|^2)$, in which we can detect the squared Euclidean distance between the two feature vectors. Gamma is computed as $\gamma = \frac{1}{n\sigma^2}$ if its value is "scale", where σ^2 is the variance of the training set and n is the number of features. Thanks to this function it is possible to transform data into an infinite dimensional space and then the SVM can perform classification by finding the decision hyper-plane that differentiates the two classes best.
The hyperparameter C which can be interpreted as a regularization parameter, balances correct classification and maximization of the decision function's margin.

[2] Classifiers were compared making use of their default configurations.

4. Tuning and validation (max. 400 words)

Tuning was the most trivial aspect of the task. A relevant role was assumed by the TfidfVectorizer hyperparameters: min_df, max_df. Min_df and max_df denote the minimum and maximum document frequency respectively, which means that terms which appear in less than min_df documents or appear in more than max_df documents are ignored. Different values for min_df were tried, such as 0.05, 0.01, 0.001, which represent percentages of documents, and 5 which is an absolute number of documents. The best result was obtained with min_df=5, which demonstrates that is useful to cut unusual terms (outliers) but it is not a good idea to cut too much. For max_df some values (0.95, 0.99, 0.999) were tried in combination with min_df, but it was observed that better results were given without any threshold on the maximum document frequency, that means max_df=1.0.

Dimensionality reduction has been performed both with PCA and TrucatedSVD, varying the number of components. PCA with 11 components (as suggested by the graph of the explained variance of each component) gave out the highest weighted F1-score among the ones performed with PCA. In fact, even other values were tuned.

All results obtained with PCA, except for speed, were worse than the ones performed with TruncatedSVD, so SVD was preferred.

Also different numbers of components were tried for TruncatedSVD (30, 200, 800, 3000…). Low numbers of them did not allow to perform good results because they could not explain enough variance, but too many components made the classification process very slow. So, a trade-off between quality and lower computational time was made taking the minimum number of components which could explain the 80% of total variance (n_components=1828).

To select the best configuration for SVC, the grid search had been used exploiting 3-fold cross validation. Grid parameters which were combined were kernel (rbf and linear), gamma (“scale”, 1.0, 10.0) and C (1, 50, 100). The picture below shows the result.

```
Best parameters set found on development set:

{'C': 50, 'gamma': 'scale', 'kernel': 'rbf'}

Grid scores on development set:

0.958 (+/-0.001) for {'C': 1, 'gamma': 'scale', 'kernel': 'linear'}
0.958 (+/-0.001) for {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
0.958 (+/-0.001) for {'C': 1, 'gamma': 1.0, 'kernel': 'linear'}
0.958 (+/-0.002) for {'C': 1, 'gamma': 1.0, 'kernel': 'rbf'}
0.958 (+/-0.001) for {'C': 1, 'gamma': 10.0, 'kernel': 'linear'}
0.550 (+/-0.000) for {'C': 1, 'gamma': 10.0, 'kernel': 'rbf'}
0.937 (+/-0.003) for {'C': 50, 'gamma': 'scale', 'kernel': 'linear'}
0.959 (+/-0.002) for {'C': 50, 'gamma': 'scale', 'kernel': 'rbf'}
0.937 (+/-0.003) for {'C': 50, 'gamma': 1.0, 'kernel': 'linear'}
0.958 (+/-0.001) for {'C': 50, 'gamma': 1.0, 'kernel': 'rbf'}
0.937 (+/-0.003) for {'C': 50, 'gamma': 10.0, 'kernel': 'linear'}
0.550 (+/-0.000) for {'C': 50, 'gamma': 10.0, 'kernel': 'rbf'}
0.933 (+/-0.004) for {'C': 100, 'gamma': 'scale', 'kernel': 'linear'}
0.959 (+/-0.002) for {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
0.933 (+/-0.004) for {'C': 100, 'gamma': 1.0, 'kernel': 'linear'}
0.958 (+/-0.001) for {'C': 100, 'gamma': 1.0, 'kernel': 'rbf'}
0.933 (+/-0.004) for {'C': 100, 'gamma': 10.0, 'kernel': 'linear'}
0.550 (+/-0.000) for {'C': 100, 'gamma': 10.0, 'kernel': 'rbf'}
```

It can be seen that some good configurations are quite similar in terms of weighted F1-score. Actually, the two which performed best (with score 0.959) gave the exact same classification, so it was decided to keep just one of them (kernel = “rbf”, gamma = ”scale”, C = 50) in final solution.
Then, for the second solution, among the configurations with a score of 0.958, the default configuration for SVC was chosen (kernel = “rbf”, gamma = “scale”, C = 1).