POLITECNICO DI TORINO

Master's Degree Programme in
Data Science and Engineering

Winter Session Assignment 2020

Sentiment analysis task with TripAdvisor's reviews

277046 Giovanni Cioffi

Academic Year 2019-2020

Summary

1. Data exploration

The development and evaluation sets are loaded as pandas DataFrames through *read_csv* function. 28754 reviews are given, no missing values have been detected and every TripAdvisor's review is assigned a label correctly, depending on its positive or negative sentiment. An important thing to point out is that the development dataset is not labelled uniformly. In fact, as shown in figure 1, 67.9% of reviews is labelled with class 'pos' whereas the remaining part is labelled 'neg'. This information will significantly affect the classification task as discussed in *Tuning and validation* section.

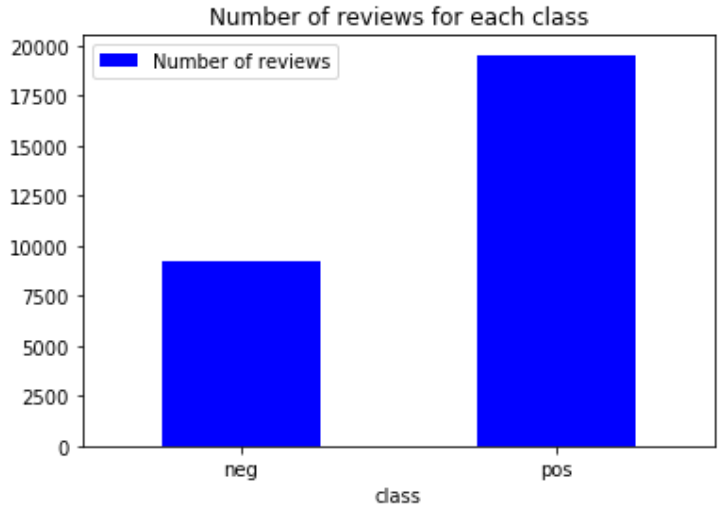

Figure 1 - Number of reviews for each class in develompent.csv

A further analysis was made in order to compute the average length of reviews in terms of number of words. Figure 2 shows how negative reviews are a slightly longer on average with respect to the positive ones; since the difference is just about 40 words on average, it won't be considered in the next steps.

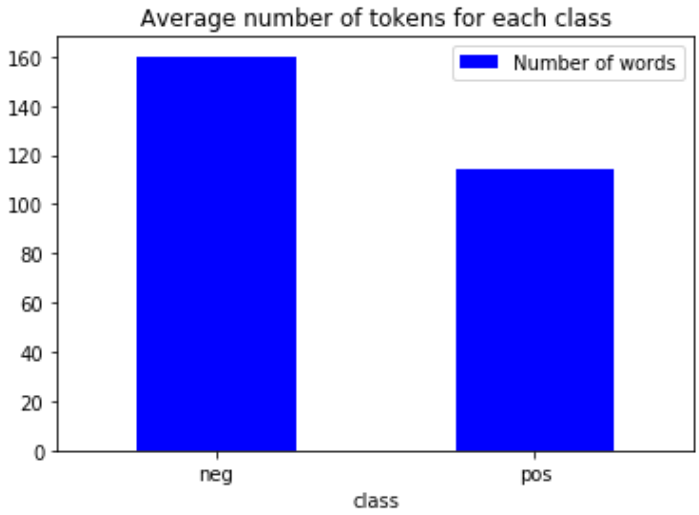

Figure 2- Average number of tokens for each class in development.csv

The following assumption was made: reviews belonging to the same class share the same 'sentiment', satisfaction for the positives and disappointment for the negatives (or a levered combination of the two). Thus, they are likely to be written in a similar way, using common set of words. In order to highlight most discriminative words, wordclouds have been realized separately for positive and

negative classes. It could be easily foreseen that the most recurring items would have been part-of-speech terms such as: punctuations, articles, prepositions, demonstrative and possessive adjectives, pronouns, common verbs and adverbs. Therefore, a list of them provided by *nltk* have been filtered out for this data exploration part. The results are the following:

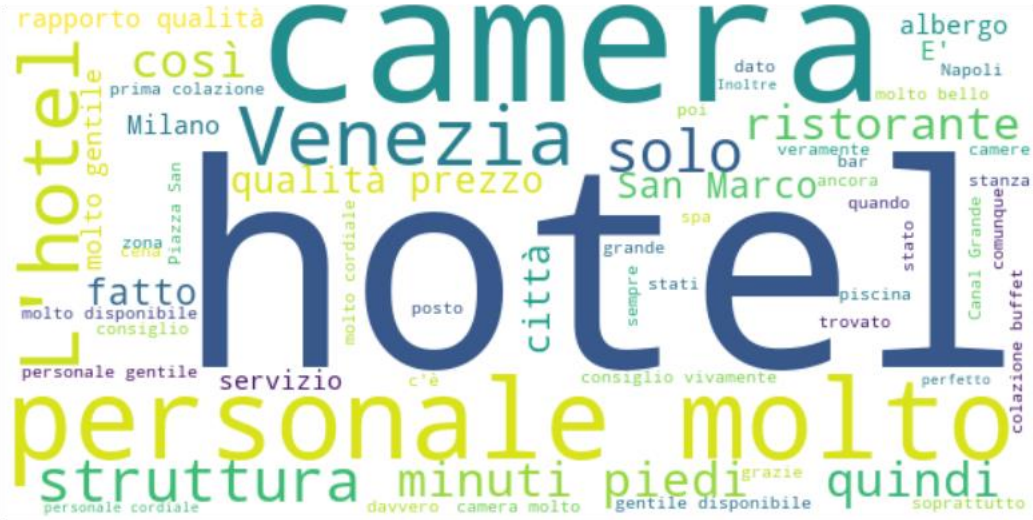

Figure 3 - Positive reviews wordcloud before preprocessing

Figure 4 - Negative reviews wordcloud before preprocessing

Even though some stopwords were removed, it was still hard to distinguish class-characterizing terms, especially for the negative class. This is due to the following reasons:

- Discriminative terms are hidden by most frequent words related to hotel context such as “camera”, “hotel”, “colazione”, “posizione”.
- Different inflections of the same lemma appear as frequent different items, for example “stato”, “stati”, “era”, “c’è”, “è” should simply result as “essere”. Same for “hotel”, “l’hotel”, ”dell’hotel” or “camera”, “camere”.

Data exploration stated that tokenization, lemmatization and a further filtering process were necessary to highlights the most important features for the classification. These steps are covered in the next *Preprocessing* section.

2. Preprocessing

2.1 Tokenization and first filtering

Each review has been tokenized applying ntlk word_tokenize function, which split every item of the text. Each token is turned into lower case, then the following are filtered out:

- Tokens consisting of less than three characters (prepositions, pronouns or simply typing errors)
- Digits and tokens containing numbers (such as hours in hh:mm format or again typing errors)
- Punctuations
- *ntlk* Italian stopwords (except for token ‘non’, which will turn out to be crucial for most important bigram features)

2.2 Lemmatization and second filtering

Lemmatization process has been carried out with *Spacy lemmatizer*. It has been a crucial step for grouping together the inflected forms of a word in a way that they can be analysed as a single item, identified by the word's lemma. This process reduces the general dimension of the problem embedding different inflections of the same lemma; the result is a single feature that preserves the semantic context. Each lemma is then filtered again removing other stopwords which are detected with a second wordcloud analysis. In this case, words with a high frequency of occurrence in both classes are removed because they can’t help in discriminating classes. Thus, the following are filtered out:

- Words related to hotel and travelling context (already mentioned in the *Data exploration* section)
- Common verbs and adverbs

The wordcloud for the positive class is shown below:

Figure 5 - Positive review wordcloud after lemmatization

2.3 Vectorization

Textual data needs to be processed in order to obtain a numerical representation of each review. This is achieved via the application of TF-IDF weighting schema thanks to *TfifdVectorizer* class. Its *transform* method provides an $m \times n$ sparse matrix (*X_train*) where m is the total number of reviews in the development set and n is the number of features (all the possible tokens, stopwords excluded). Each element of this matrix is the term frequency-inverse document frequency of that term in that review. TF-IDF appeared as the most suitable weighting schema because it

- gives less importance to terms occurring virtually in all reviews
- ensures that the matching of reviews of the same class is more influenced by discriminative words which have relatively low frequencies in the entire collection
- takes care of normalization of reviews of different sizes

The vectorizer learns the vocabulary fitting on the development test, then the evaluation set is transformed in order to match the same number of features.

3. Algorithm choice

Given the vector representation of the development and test set, the classifier model had to be built exploiting one of the classification algorithms. In order to choose the probable best performing one, *Principal Component Analysis* was applied by means of the *Singular Value Decomposition* (SVD) for reducing the feature space in 2 and having a visual perspective on labelled reviews. In this way, a 2D representation of labelled points was obtained using *matplotlib*.

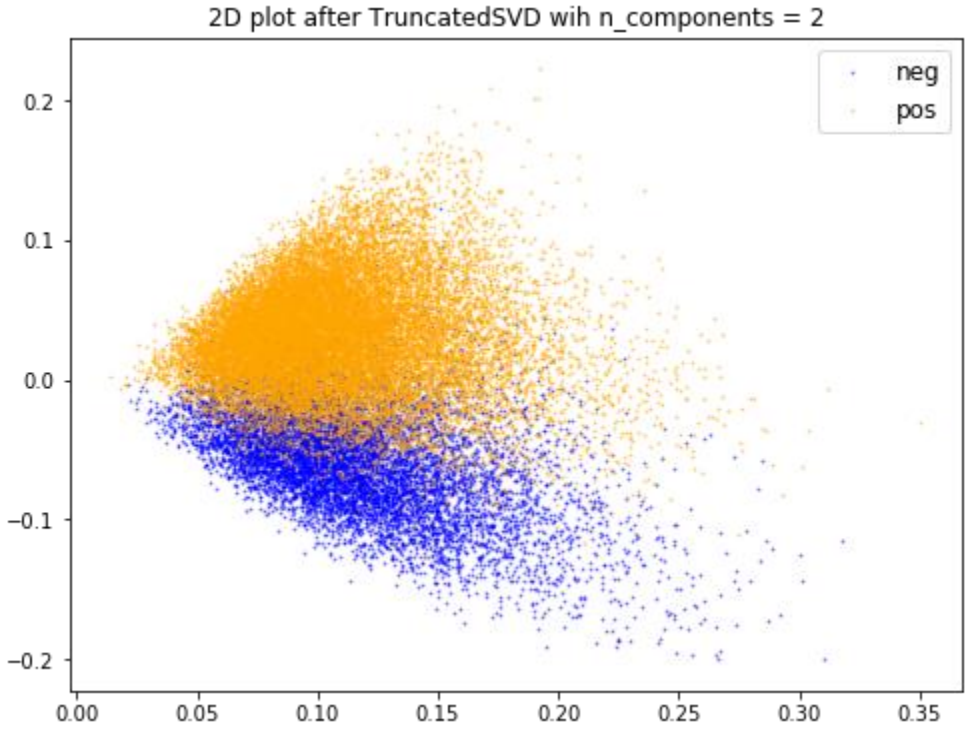

Figure 6 - 2D plot after TruncatedSVD with n_components = 2

As shown in figure 6, the positive and negative classes seem to be well-distinguishable and separable (by a line in 2D space, a plane in 3D space and so on and so forth), even though the components

extracted by the decomposition express less than 1% of the explained variance (which is a reasonable number considering that initial features space was reduced from more than 180000 dimensions to 2). Nevertheless, the hint that came from this plot was that one of the linear classifiers could be the most appropriate for their ability to work on binary classification problems (using linear combinations of the features) and their speed in high-dimensional spaces. Hence, the following algorithms were taken into consideration: *LinearSVC* and *LogisticRegression* classifier.

LinearSVC algorithm belongs to *Support Vector Machine* supervised learning methods. It represents each data item (in this case each review) as a point in n-dimensional space (where n is number of features, in our case all the possible unigrams and bigrams); the value of each feature could be the tf-idf of that word in that review. The classification is performed by finding the hyper-plane that differentiate the two classes maximizing the margin between the decision hyperplane. Therefore, all points on one side of the hyperplane are classified as "pos", while the others are classified as "neg".

Logistic regression was considered too, it is a probability-based classification algorithm used to assign elements to a discrete set of classes, in our case assigning a positive or negative sentiment to reviews. Logistic regression transforms its output using the sigmoid function (also called logistic function) to return a probability value which can then be mapped to one of the two classes using a decision boundary.

Both *LinearSVC* and *LogisticRegression* were implemented for the classification task; they both resulted in high f1-score as it will be discussed in the next *Tuning and validation* section.

4. Tuning and validation

4.1 Vectorizer parameters

Vectorizer parameters turned out to be very influent in terms of dimension of the classification problem and final f1-score. In particular, *n_grams* was set to (1,2) in order to keep unigrams and bigrams, which means features composed of one or two words. This was a critical step because bigrams allow to take into consideration very important features that otherwise would be lost. Even worse, some of them could become misleading. For example, all the negations of verbs such as ‘non tornare’, ‘non consigliare’ (that surely express a negative sentiment) would lose the negation and would result in a feature of totally contrary sentiment. For the same reason, the word ‘non’ has been removed from the stopword list. Minimum document frequency set to *min_df* = 2; this choice causes a significant increase of the number of features, but both *LinearSVC* and *LogisticRegression* do not have computational problems in high dimensional spaces. On the contrary, they achieved the best results in terms of f1-score with higher number of features.

4.2 Classifier parameters’ tuning and classification report

In order to set parameters of both *LinearSVC* and *LogisticRegression,* a classification model is built exclusively exploiting the development set using *GridSearchCV* with K-fold (with K=10) cross validation in order to avoid overfitting. Most interesting parameters set are *class_weight='balanced'* for both classifiers, that automatically tries to fix the unbalancing of the classes already mentioned in *Data exploration* section and the regularization parameters, set in order to prevent overfitting (default C=1 for *LinearSVC* and C=100 for *LogisticRegression)*.

Main classification metrics were calculated with *classification_report*, the model was trained and tested using *train_test_split* with *shuffle=True* and the parameters chosen with the previous tuning. In the figure below classification report and confusion matrix is shown for *LinearSVC*:

	precision	recall	f1-score	support
neg	0.9492	0.9459	0.9476	2827
pos	0.9737	0.9753	0.9745	5800
accuracy			0.9657	8627
macro avg	0.9615	0.9606	0.9610	8627
weighted avg	0.9657	0.9657	0.9657	8627

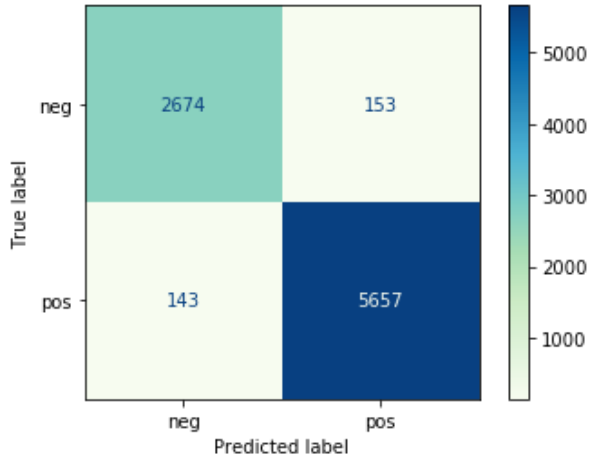

Figure 7 - Classification report and confusion matrix for LinearSVC

It is possible to notice that precision and recall of negative class are slightly lower than the positive. This seems reasonable because the model was trained on an unbalanced set of data. Better results would have surely been achieved by balancing the development set adding an external set of negative reviews coherent to the context. Nevertheless, both algorithms resulted in f1-score higher than 96%.

4.3 Most important features analysis

Combining *coeff_* classifier's attribute with vectorizer's *get_feature_names* the most characterizing features for both classes can be determined. For *LogisticRegression* classifier the 10 most discriminative ones for positive class are shown below:

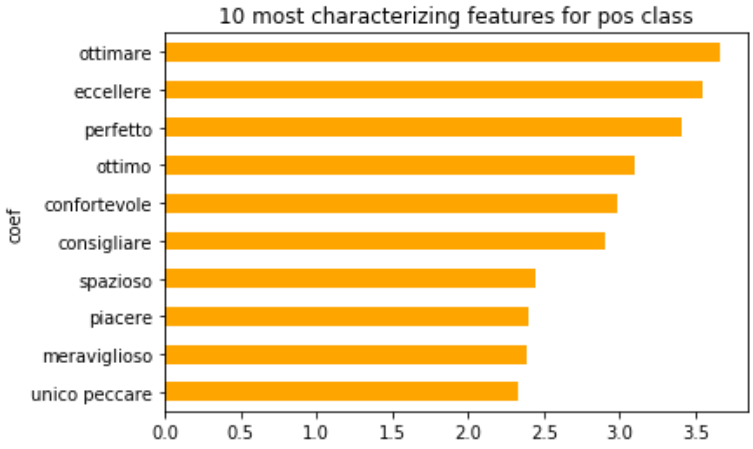

Figure 8 - 10 most characterizing features for pos class using LogisticRegression