Data Science Lab: Process and methods
Politecnico di Torino

Project report
Student ID: -

Exam session: Winter 2020

1 Data exploration (max. 400 words)

The assigned task is to perform a sentiment analysis of textual reviews and build a classification model to predict whether the sentiment contained in a text is positive or negative. Loading the dataset, we verify that there are 41077 reviews, of which 28754 are labeled and 12323 are unlabeled. The labeled documents, i.e. the development set, will be used to train a model that is able to classify the reviews in the evaluation set. Each line in the development set consists of two fields: `text` contains the review written by the user, and `class` contains a label that specifies the sentiment of the review. A first analysis of the data shows that neither set contains missing values or empty strings. The `class` field only contains one of two values, `pos` or `neg`, specifying the sentiment of the review. The dataset is not well balanced, as most of the reviews show a positive sentiment. We show the distribution of the classes for the development set in Figure 1. The dataset has been scraped from the TripAdvisor Italian website, therefore we expect to find hotel or restaurant reviews. Further exploration of the reviews tells us they refer to hotel stays. Most of the reviews are written in Italian, but we can find English and Spanish reviews, as well as Chinese characters. This suggests that there are reviews which were originally written in another language and subsequently translated to Italian. A number of reviews contain spelling errors and words with repeated characters. The text contains emojis that express sentiment, which we shall consider in the analysis [1]. Furthermore, special Unicode characters are present and should be removed.

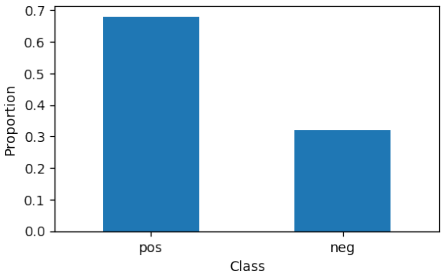

Figure 1: Class distribution

2 Preprocessing (max. 400 words)

The dataset contains textual data from which we must extract a set of features. We carry out a data preprocessing procedure that comprises tokenization, stopword removal, basic spell correction and stemming, and use a weighting schema to transform the collection of reviews into numerical feature vectors. We process every review with our custom tokenizer and the aid of the *Natural Language Toolkit* library (`nltk`), and we apply the term frequency-inverse document frequency (TFIDF) schema with the vectorizer provided by the `scikit-learn` [5] library. The custom tokenizer begins by dividing the document into substrings, or tokens, on any non-alphanumeric character it encounters when scanning the text. It handles emojis with the help of the library `emoji` [7], creating a token for every emoji present in the text. It discards punctuation tokens unless it finds an exclamation mark, a question mark or a currency symbol, as we speculate they carry additional meaning and may be useful to the classification algorithm. It removes any non-alphabetic character and does not consider tokens of length greater than w_{upper} or less than w_{lower}. It applies a basic spell checking algorithm to the accepted tokens, which corrects characters that consecutively repeat more than twice. It deletes words that appear in the stopword list provided by the `nltk` library which we slightly modify: we add the word "hotel" which appears frequently in hotel reviews and remove the word "non" which expresses negativity and therefore may be useful for a sentiment analysis task. As a final step, we use the *Snowball* stemmer [6] which provides an algorithm for the Italian language. We use a stemmer instead of a lemmatizer or POS tagging algorithm as it runs significantly faster, can be easily implemented for languages other than English and delivers satisfying results. We override the tokenizer provided by `scikit-learn`'s TFIDF vectorizer with our own custom tokenizer and build the term-document matrix. We convert all characters to lowercase and we extract n-grams from the documents. We ignore tokens that have a document frequency strictly lower than df_{min} and set a maximum number of features f_{max}. We show two word clouds in Figures 2 and 3 representing the extracted term's frequencies for positive and negative reviews.

Figure 2: Word cloud for class `pos`

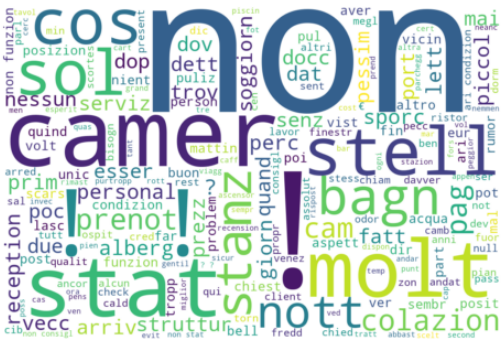

Figure 3: Word cloud for class `neg`

3 Algorithm choice (max. 400 words)

The development set is a labeled training dataset. Therefore, we shall use a supervised learning algorithm to train a classifier able to identify positive and negative reviews correctly. Many algorithms are available to perform text classification. Naive Bayes and support vector machines (SVMs) are very effective in high-dimensional spaces not unlike ours, and both methods are highly regarded in academic literature for sentiment analysis tasks [2] [3] [4]. For these reasons we take both into consideration. The `scikit-learn` library provides several versions of the naive Bayes classifier, including multinomial naive Bayes and complement naive Bayes. The latter is an adaptation of the standard multinomial algorithm that is particularly suited for imbalanced datasets such as ours. The library also provides the C-support vector classification (SVC) algorithm, which is a support vector machine algorithm that employes a regularization parameter. We compare the complement naive Bayes classifier to support vector machines, and in particular to SVC. While the former is faster in training and prediction, the latter performs consistently better, therefore we use SVMs for our classification problem. For two-class, separable training data sets, the SVM searches for a linear separator between the classes. It looks for a decision hyperplane that is maximally far away from any data point in the training set. Our development set is a two-class training set, but it does not seem to be sufficiently separable by a linear model, therefore we explore other nonlinear SVMs that map the original feature space to some higher-dimensional feature space where our training set is better separable. We find that the best suited kernel for our data is the radial basis function (rbf) kernel and we choose a regularization parameter C_0.

4 Tuning and validation (max. 400 words)

First, we tune the parameters of the tokenizer. We set $w_{upper} = 20$ and consider different values for w_{lower}. We get the best results by using $w_{lower} = 3$. We try to retain both alphabetic and numeric characters, but this results in worse performance than only keeping alphabetic characters. We initially use the default set of stopwords for the Italian language provided by the `nltk` library, and then we modify this set by adding a domain-specific stopword and removing "non", mostly used in negative reviews. This yields better results. We try to delete all punctuation tokens, including exclamation marks and question marks, however this yields worse results. After tuning the tokenizer, we change the parameters of the TFIDF vectorizer. We set $df_{min} = 2$ and reduce the number of features. We attempt to extract n-grams from our tokenized documents. We start by only using bigrams, but this gives bad results. Then we attempt to use both unigrams and bigrams, which results in a better score. However, we get a very large number of features (around 200000, almost ten times greater than the number of samples), so we set an upper bound of $f_{max} = 15000$. The classifier now works very well, reaching the highest `f1_weighted` score we obtained. We use $f_{max} = 20000$ for our second entry. We attempt to further reduce the dimensionality of the dataset using `TruncatedSVD`. Using 8000 components, we retain around 90 percent of the explained variance. Nonetheless, this does not translate to a better result. To test all these changes, we split the development set into a training set and a validation set with the `train_test_split` utility, using `stratify` to preserve the label distribution and evaluating the `f1_weighted` score on the predictions. We use 80 percent of the development set for training and the remaining 20 percent for validation. We train the final model on the whole development set and test results on the portal. Finally, we tune the parameters of the SVC using `GridSearchCV`, which performs a cross-validated grid search over a parameter grid. We use `f1_weighted` as evaluation metric. We find that the best combination of parameters for the classifier is $C_0 = 10$ with a rbf kernel. These parameters are very similar to the default ones, therefore the grid search did not improve the model by much. We show a confusion matrix computed on a test split of the development set in Figure 4.

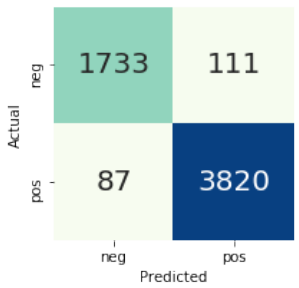

Figure 4: Confusion matrix

References

[1] Alexander Hogenboom, Daniella Bal, Flavius Frasincar, Malissa Bal, Franciska de Jong, and Uzay Kaymak. Exploiting emoticons in sentiment analysis. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, page 703, Coimbra, Portugal, 2013. ACM Press.

[2] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. Text Classification Algorithms: A Survey. *Information*, 10(4):150, April 2019. arXiv: 1904.08067.

[3] Akrivi Krouska, Christos Troussas, and Maria Virvou. Comparative Evaluation of Algorithms for Sentiment Analysis over Social Networking Services. page 14, 2017.

[4] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, New York, 2008. OCLC: ocn190786122.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Matrin Porter. Snowball stemmer algorithm. https://snowballstem.org/.

[7] Kim Taehoon and Kevin Wurster. Emoji library. https://pypi.org/project/emoji/.