Emojify: Sentiment classification with emojis

Alberto Bernardi
alberto.bernardi.6@studenti.unipd.it

Gianmarco Cracco
gianmarco.cracco@studenti.unipd.it

Abstract

The aim of this project is to compare different models that can correctly perform sentiment analysis based on emoijs. We applied several algorithms, like Logistic Regression, k-NN, Linear Discriminant Analysis, Support Vector Machine and Deep Neural networks. We also investigate the role of standardization of our dataset. The main issue encountered was the limited dimensions of the dataset (only $\sim$130 instances), which played a central role in the performance evaluation.

1. Introduction

The relevance of sentiment analysis is clear and highlighted by the huge interest it has awakened in the last years not just inside of the scientific community. In particular, the connection of this project with emojis makes its importance even more concrete: instant messaging platforms and social networks, indeed, represent the most widespread communication means, and being able of correctly predicting the emoji to be associated with a certain phrase may really increase the immediacy, effectiveness and comprehensibility of our message. To be able of confidently predict emojis, we applied different Machine Learning techniques and also a deep neural network. For each model indiscriminately, we performed a grid search with cross-validation, which was particularly well suited and computational affordable for a small dataset like ours.

2. Dataset

Our dataset consists of 188 phrases, split into a training set of 100 instances, a validation set of 32 and a test set of 56. Each one of these phrases was processed using the Global Vector for Word Representation model (GloVe, ed.): every word was converted into a 25-dimensional vector in such a way that distance between vectors is related to the semantic similarity of the corresponding words. Once words are converted into vectors, we concatenated those composing a phrase and we padded with vectors of zero those with a length smaller than the maximum one (10 words). In the end, we obtained each phrase encoded in a $10 \cdot 25$ dimensional vector.

Our target variable, the emojis, consists of 5 different classes: each one of them has been associated to a number between 0 and 4 as reported in Table 1, while from Fig. 1 we can observe how training data are quite unbalanced, especially for class 4, but not in an exaggerate way.

Class	0	1	2	3	4
Emoji	❤️	⚾	😄	😞	🍴

Table 1.

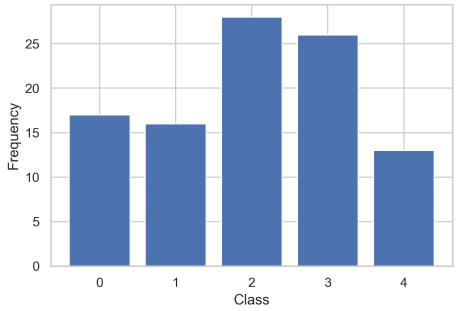

Figure 1. Distribution of the labels in the training set

In order to be able of visualizing our data, we tried to reduce the dimensionality of our 250-dimensional vectors. To do so we applied Principal Component Analysis (PCA) considering only two principal components. We also considered the respective Voronoi polygons as shown in Fig. 2 for the original dataset and in Fig. 3 for the standardized one (we will go deeper in this later): this visualization will come in handy when we will analyze the $k-$NN model in the next section.

3. Method

Due to the small size of our dataset we decided to tackle the fitting of the different models applying k-fold cross validation (we set $k = 10$ everywhere except in most complex models, the deep network, where the number of parameters is quite high), to get the best out of few data: in fact, for these types of dataset it is very frequent to overfit the

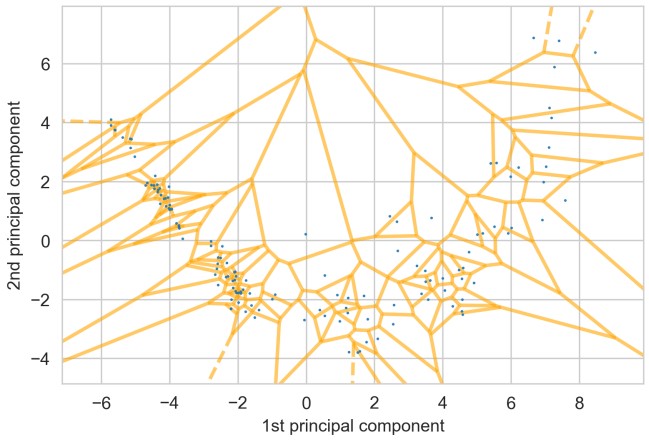

Figure 2. Voronoi diagram for the original dataset with 2 PCA components.

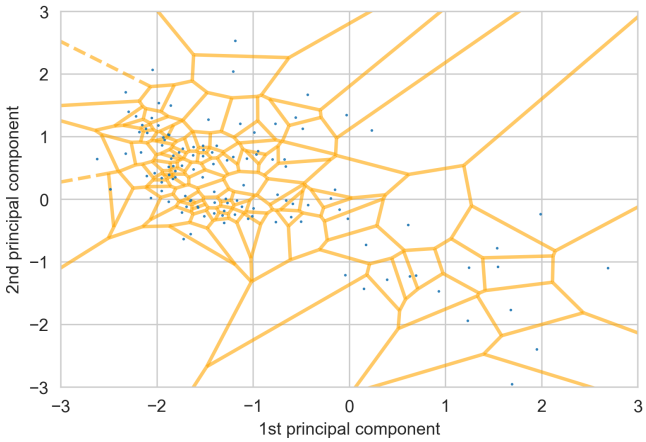

Figure 3. Voronoi diagram for the 2 PCA components standardized dataset.

training data, a risk that cross-validation tries to minimize. Applying this technique led us to merge the training and validation sets into a unique one of 132 instances.

At the same time, the small number of samples also allowed us to perform a grid search over several parameters in a reasonable amount of time, so that the quest for the best model was quite exhaustive. To properly address the overfitting issue, whenever it was possible, we tried to improve the generalization of our models by selecting multiple regularization parameters inside of the different search grids.

Another aspect we considered during our evaluation of the different models was the relevance of standardizing our data, trying to verify the benefits of this transformation as reported in [1].

In order to build robust predictors, we tried different models, both parametric and non-parametric, implementing them using the Python library `Scikit-Learn` with the exception of the deep model that we built relating on `Tensorflow`:

- *Logistic Regression (LR)*: it is one of the most classical methods for tackling a classification task. In particular, we compared which approach was the best among the *one-vs-one* and the *one-vs-all* multi-class classification. The cross validated grid search suggested the one-vs-all approach with a $\mathcal{L}^2$ regularization applied. In this case, it was observed that the standardization of the data improved the results.

- *k-NN*: another very intuitive and common approach to a multiclass classification problem is k-Nearest Neighbours. Also in this case we applied a grid search combined with a 10-fold cross validation: as grid parameters we chose values between 1 and 50 to be assigned to k, and we searched among three different distances (Euclidean, Manhattan and Chebyshev). The best model suggested $k = 1$ and the Euclidean distance for both the datasets. To visualize how k-NNs work, Voronoi diagrams come useful: with $k = 1$ and the euclidean distance, we can actually display the decision boundaries for this particular algorithm[1]. As we can see from Fig. 2 and Fig. 3 applying the standardization to our data led to a more evenly spread out points. This is reflected even in our results, that displays a big improvement in the accuracy.

- *Linear Discriminant Analysis (LDA)*: for this model we tried several regularization values, but in contrast with the previous algorithms, in this case regularization does not act on the loss function itself but, instead, it modifies the way the elements of the covariance matrix are evaluated. The parameters chosen through the grid search with a 10-fold cross validation were equal to 0.2 for the original dataset and 0.4 for the transformed one. For this model the standardization helped a lot leading to an increase of $\sim 10\%$.

- *Support Vector Machine (SVM)*: in fitting our SVM, we focused the grid search on the $\mathcal{L}^2$ regularization parameter for the objective function; the selected value for this parameter was 0.01, which is quite a strong regularization, if compared to the default value of one. This suggests a strong attempt to reduce the overfitting for the model, that seems to be, once again, the main issue for this dataset. In this case, the standardization of the data seemed not to be effective, since it just worsen the performance. One vs all

- *Neural Network (NN)*: as a final model, we built a deep neural network made up of the input layer, two hidden

d=|x1-x2| +|y1-y2|

Distanza massima lungo gli assi: max(| x_i -y_i |

[1] Clearly, the decision boundaries are just representative since the PCA transformation is likely to slightly alter the distance between the 250-dimensional data points

layers and the output layer: the last one, in particular, was 5 units wide (one for each class) and activated through the softmax function, which is the most suitable for a multi-class classification. We decided to go deep, since generally, with a comparable number of neurons, deeper networks generalize better. Despite in most of the cases it has to be avoided performing k-fold validation and grid search on a deep network because of the huge computational cost it may have, we anyway decided to perform both (limiting k to 5) leveraging the limited number of instances, which made the running time feasible. In the end, the suggested model featured 50 units in first hidden layer, 40 in the second and two $\mathcal{L}^2$ regularization parameters equal to 0.001. To avoid an exaggerate grid search, activation functions and number of epochs were manually tuned: in particular, the epochs were limited to 10, since more would have caused significant overfitting.

4. Results

The performance for the best models selected by the different grid searches are reported in Table 2: the results were not as good as we could have expected, with the Logisitc Regression and the Support Vector Machine best performing models, that achieve a modest 0.68 accuracy, while the others scored in some cases even under 50%. It is the case of k-NN and, more surprisingly of the deep model, that even in the best case achieved a low 54%. We stress, once again, how this could mainly depend on the reduced dataset: with more or less twenty training instances for each label, the models are not able to extract truly significant information from the training data, so that when it comes to testing, they are mislead in their predictions.

Despite the poor accuracies, something really interesting is the relevance of standardization, which increased the score in almost any case, sometimes even dramatically. Even if its importance might depend not just on the model, but also on the dataset considered, it is still worth a try, since, as we saw, it can bring many benefits and really improve the performance.

Algorithm	Original Dataset	Standardized Dataset
LR	0.61	**0.68**
k-NN	0.42	**0.61**
LDA	0.54	**0.64**
SVM	**0.68**	0.61
NN	0.46	**0.54**

Table 2. Accuracy evaluated on the test sets.

5. Conclusions

The achieved scores might be, at first glance, very disappointing: in particular, if compared with those achieved in [1], they are much worse, but we always have to keep in mind that those were achieved on much bigger datasets, while the reduced dimensionality of ours may have played a significant role. In many cases we achieved perfect accuracy on the training set, which may be a sign of overfitting if compared with the accuracies on the test sets; though we really focused on it, performing cross validation and always selecting many regularization parameters for our grid searches we didn't manage to avoid overfitting. Probably, with some more data we could have better addressed this issue, but still, we obtained good insights on the potential relevance of standardizing our dataset and on how Machine Learning techniques may outperform Deep networks, making the model choice for a multi-class classification task not immediate and trivial.

References

[1] Yang-Yin Lee, Hao Ke, Hen-Hsen Huang, and Hsin-Hsi Chen. Combining word embedding and lexical database for semantic relatedness measurement. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, page 73–74, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.