# Daimler technical challenge – Twitter sentiment analysis

This report details my solution to tackle the ‘Twitter sentiment analysis’ challenge.

The corresponding code, in the Jupyter notebook format, is also provided.

## Data cleaning

Tweets are usually written using casual language and extensive use of non-alphanumeric characters, meaning that some data cleaning is necessary:

* Remove usernames and hashtags (Note: this information may be relevant for a deeper analysis, but it will be ignored in this context);
* Remove special characters, numbers, punctuations;
* Convert all characters to lowercase, thus improving data quality by reducing the size of the feature space;
* Remove short words, thus improving data quality by reducing the size of the feature space (Note: this is not the best approach, as it indiscriminately removes potentially useful words (ex: verbs such as ‘be’ and ‘do’));
* Remove stopwords, thus improving data quality by reducing the size of the feature space in a less destructive way (Note: this still removes some relevant words (ex: not), meaning that a more careful analysis is required);
* Perform lemmatization (instead of stemming), as it group together the inflected forms of a word instead of just operating on a single word without knowledge of the context;

Removing or correcting misspelled words (ex: desparately), slang (ex: cuz, u), and others (ex: juuuuuust) would also be a very interesting way to further improving data quality by reducing the size of the feature space, but there is no time for this analysis in this context.

## Data visualisation

No data visualisation analysis was performed, as it is only useful for a more detailed approach, for which there is no time in this context.

## Feature extraction

Several different features were computed:

* term frequencies from a Bag-of-Words (BoW) (Note: this generates a very large feature space, which must be kept small artificially);
* tf-idf frequencies, which improve on simple term frequencies by reflecting how important a word is to a document in a collection (Note: this also generates a very large feature space, which must be kept small artificially);
* hashed frequencies, which perform feature hashing ('hashing trick') (Note: this allows for controlling the size of the feature space in a more natural way);
* one-hot feature encoding, as a preparation for computing word embeddings used for training Neural Networks;

## Model training and evaluation

The original training set was split into training and validation sets, following a 70/30 ratio. This was done to detect possible overfitting during model training and provide a simple criteria (i.e., performance on the validation set) to chose the algorithm that performs best.

Four traditional classifiers (Logistic Regression (LR), Support Vector Machines (SVM), Decision Trees (DT), Naïve Bayes (NB)) and two classifiers based on Neural Networks (Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM)) were considered. Logistic Regression and Support Vector Machines were chosen for being the most likely to obtain the best classification results. Decision Trees and Naïve Bayes were chosen mainly as sanity checks, in order to have a lower bound for classification performance. The Multi-Layer Perceptron was chosen for being one the simplest Neural Network architectures, in an attempt to cope with the small amount of available training data. The Long Short-Term Memory network was chosen for being the most adequate for modelling the relationships between different words.

The following results in terms of F-measure were obtained on the validation set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **term frequencies** | **tf-idf frequencies** | **hashed frequencies** | **one-hot encoding** |
| **LR** | 44.98% | **46.38%** | 43.00% | - |
| **SVM** | 42.92% | 41.51% | 39.51% | - |
| **DT** | 29.43% | 31.79% | 24.47% | - |
| **NB** | 17.88% | 18.33% | 15.50% | - |
| **MLP** | 28.89% | - | - | - |
| **LSTM** | - | - | - | 41.93% |

The best results were obtained using Logistic Regression. The label prediction of testing set provided by the Logistic Regression model trained with tf-idf frequencies obtained F-measure=45.12% (according to the submission website).

## Discussion of the results

Logistic Regression performs well (when compared with many other classifiers) with diagonal feature decision boundaries, which occur naturally in this context due to the relationships between different words.

Similarly to Logistic Regression, Support Vector Machines also perform well with diagonal feature decision boundaries, thus having similar performance even when a linear kernel is used. However, SVM require much longer training times in this context, mainly due to the large amount of features.

Decision Trees perform poorly, as decisions at each node are based on an individual features, thus overlooking the relationships between different words. Nevertheless, this effect is minimized by the fact that different decision paths along the tree represent some of these relationships.

As expected, the Naïve Bayes classifier performs very poorly, as it assumes feature independence, when in fact the exact opposite is true (in any tweet, most words are related to each other).

Multi-Layer Perceptrons perform worse than expected, probably due to the available data being insufficient to properly train their dense interlayer connections (underfitting).

Long Short-Term Memory networks are a specific type of Recurrent Neural Network capable of learning the relationships between elements in an input sequence (in our case, word sequences in tweets), therefore performing adequately even when very small amounts of data are available.

The best result (obtained using a Logistic Regression model trained with tf-idf frequencies) is a poor result, leaving much room for improvement. The main issue with all classifiers (mainly the ones based on Neural Networks) is probably the small amount of available data with high variability, especially given that some data cleaning actions were not taken. Particularly for the LSTM classifier, the use of pre-trained word embeddings from large corpora would probably lead to significant classification performance improvements, as larger networks could be trained.

Finally, here are some useful considerations regarding parameter grid search and tuning, and why they were not thoroughly performed during this challenge. Either of these actions takes a very long time to perform adequately, especially since their complexity grows exponentially with the number of existing parameters (each choice of different parameters requires executing a different model training and evaluation process). Moreover, although parameter tuning can make a significant difference in classification performance, in this context it is still more likely that having more data available will have a bigger impact in terms of classification performance. Therefore, only a limited parameter search was performed, which nevertheless lead to an average 5% increase in terms of F-measure (when compared with the results obtained with the default parameters).