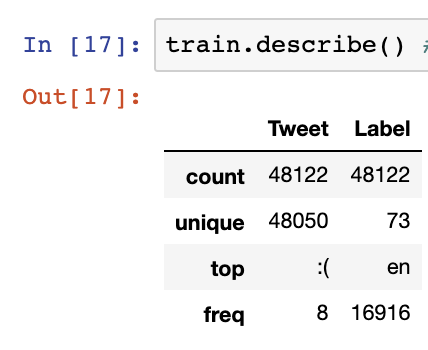
Lab report David Bielik and Debora Beuret

# Data

The data we were given was not yet it its final form. For this reason, the methodology we followed was very simple: create pandas dataframes for both datasets (the labels and the tweets) and then merge them with the IDs. With the module pandas, the process was quick and efficient.

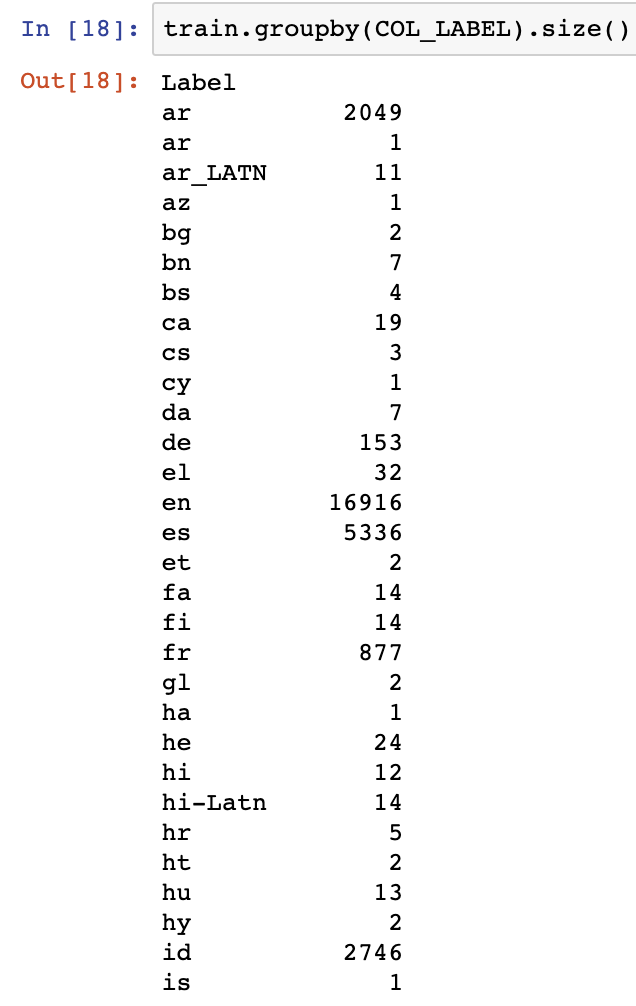
## Properties of the data



A first glimpse on the data shows that there are there are, in total, 48’122 data points on the training set. Out of those, only 48050 tweets are unique, which means some tweets must have appeared twice in the dataset. Those are presumably re-tweets.

Additionally, there are 73 unique labels, aka 73 languages in which tweets were coded. The top label was English.

Here a little outline of the labels:



We see that indeed, English is the top label (out of the showed sample). Other big contributors include Arabic and Spanish.

# Part 1 – Language identification with linear classification

## Pipelines

We considered quite a few pipelines. Obviously, **ngrams** and **tfidf** came into question and were used in our pipeline. Additionally, we tried adding the **average word length of the tweets** (see class AverageWordLengthExtractor), but it had a negative impact on the accuracy. One possible explanation is the special speech style in tweets: links, for example, have the same length in every language. There are also frequent emojis and other abbreviations that might make this particular pipeline useless.

We also considered removing stop words, but it probably wouldn’t have been a good idea as they are probably important in the classifier, allowing it to identify a given language. We, for a minute, considered lemmatizing/stemming the words, but since those modules rely on already knowing what language is used, this option soon proved impossible. For this reason, we have a limited number of features.

For all classifiers used, in the grid search, we used cv = 10, which means that 10% of the data is used as test data in each fold. This is to comply with the 90/10 split asked in the exercise.

## SGD Classifier

The parameters we tested were those described in the exercise.

* loss function: we chose to test loss with both ‘hinge’ and ‘log’ loss functions.
* regularization: we tried the options None, L1 and L2
* # of iterations: we tried 50, 100, 500, 1000.

To deal with class imbalance, ???? 🡪 the simple fact of trying many different parameters is already helping.

The best parameters were:

loss function:

regularization:

# of iterations

## Multinomial Naïve Bayes Classifier

The parameters we tested were:

* alpha: 0.2, 0.6, 0.8, 1.0
* Fit\_prior: True, False

The best parameters were:

alpha = 0.2

fit prior = False

## Multi Layer Perceptron

The parameters we tested were:

* hidden layer sizes: (4, 3) and (5, 3)
* activation functions: tanh and relu
* solver: SGD and Adam
* max iterations: 50
* momentum: 0.9

The best paramters were:

hidden layer:

activation function:

solver:

max iterations:

momentum: