A simple probabilistic model has been implemented in Python using a *sklearn* decision tree classifier*,* and the NPL library *NLTK,* to restore casing in uppercase text.

Training and testing of the classifier was done according to the following main steps:

* selection of a corpus from NLTK for training and testing
* extraction of relevant features for each word in a sentence (input definition)
* labelling of each word as *True* if it is capitalised, or *False* if it is lowercase (output definition)
* evaluating model performances on test set, and qualitatively on a “synthetic” upper case text produces from the training corpus.

More details are described in the file *model.py*:

The underlying idea of the training approach is to firstly convert uppercase text into lower case to exploit better POS tagging, and secondly to extract information about a word itself and its relation with neighbors. In particular each word *w* in a sentence *s* is POS tagged before obtaining the following main features: *w-1, w, w+1, POS-1, POS, POS+1, boolean(first w in s), boolean(w punctuated).* Including additional features (i.e. NP chunking, and suffix/prefix) is expected to further improve the model performances.

The model classification report for a 25% test set is shown below.

precision recall f1-score support

False 0.95 0.98 0.96 21752

True 0.82 0.63 0.71 3140

avg/total 0.93 0.94 0.93 24892

As expected the labeling is unbalanced, since the number of lower-case words in a mixed case sentence is dominating over the number upper-case words. Therefore, the relevant metrics that should be targeted in this case is the recall on the *True* class (i.e. capitalized words). Performance improvement is expected if the training is done in more balanced sentences.

In general this algorithm is strongly dependent on the training corpus, and it is expected to perform poorly in a completely “unrelated” upper-case text. The additional step that needs to be done is re-training the classifier on a relevant corpus for real life applications, such as news websites, twitter, and facebook. Furthermore, testing of different classifiers (Naive Bayesian and Random Forest for instance) and tuning of parameters would be beneficial to improve accuracy. More complex feature definition based on chunking and node tree connectivity can also improve the performances.

Classification of acronyms (*USA, NHS*, etc.) and words with capital letters inside (*McDonald’s*) is still missing and should be added to complete the task. This requires the definition of additional features and classes, or to combine different classifiers.

Finally, extraction of the most important features using tree based classifier would be beneficial to understand what are the most important rules that determine uppercase text recovering.