Antonio Norelli

Natural Language Processing *a.a. 2016/2017* *prof. Roberto Navigli*

Homework 1: Supervised Morphological Segmentation

Introduction

The model used is presented in *Supervised Morphological Segmentation in a Low-Resource Learning Setting using Conditional Random Fields* (Ruokolainen Teemu, et al. 2013). The idea is treating morphological segmentation as a classification problem of a sequence of letters.

System overview

The language chosen is Python 2.7, the library used are the sklearn_crfsuite library to implement the CRF and the pickle library to export the model.

Following the model the classification problem is well defined when each character of the dataset is represented by a binary vector of features and labelled with the appropriate class among the possible six. The feature vector is encoded as a dictionary, theoretically it should be very sparse with few 1 and a large amount of 0, nevertheless the crf_suite library is able to receive as input only the present features avoiding to build and deal with huge dictionaries.

The dictionaries of each character of a word are organized in a list and all these lists are organized in another list representing the desired learning set (training, dev or test). The same procedure is performed for the labels, with a string containing the label instead of the dictionary. The described data structure coincides with the accepted input format of the CRF class of the crf_suite library.

Referring to the *main.py* file, the acquisition of the datasets from the given files is performed in the first section of the code `#COLLECT DATA AND LABELLING`. The construction of the feature dictionaries and the organization in the correct data structure is accomplished by the `prepare_data` function defined in the `#COMPUTE FEATURES` section and used in the `#DATA PREPARATION AND FIT` section. In the final `#EVALUATION` section the Precision, Recall and F1 scores are computed and the results are printed on the console.

Results and analysis

The weights of the CRF are learned with the averaged structured perceptron algorithm, this algorithm has two hyperparameters, epsilon that determines the condition of convergence, and the maximum number of iterations. These parameters and the delta are tuned with a grid search in *train_hyperparameters.py.* The search intervals are chosen after some preliminary iterations except for the delta where the [3, 9] interval is chosen accordingly to the results of the article for the English language. All the possible configurations are valued on the dev set, the best one is chosen and tested on the test set, the results are reported in the table 1. This process is repeated for different split of the training data, from 100 to 1000 instances.

With the purpose of increasing the performance some extra features are added to the standard delta based. It is exploited the great flexibility of the CRFs models that are able to work with non-independent features, this property permits to add features that only emphasize patterns, without adding new information. The results are reported in the table 2. The new features are the following:

- **Left index of the letter in the word.** e.g. the features dictionary of the letter *i* in the word *drivers* will have also the entries { 'pos_start_1' : 0 ; 'pos_start_2' : 0 ; 'pos_start_3' : 1 ; 'pos_start_4' : 0 … }.
  The position of the character is intuitively an exploitable information because of the standard length of common prefixes that coincides with morphs, like *pre-* or *anti-* .
- **A flag if the considered character is *a*, *s* or *o*.** e.g. the features dictionary of the letter *i* in the word *drivers* will have also the entries { 'a' : 0 ; 's' : 0 ; 'o' : 0} while the letter *s* will have { 'a' : 0 ; 's' : 1 ; 'o' : 0}. Notice that this information is completely redundant because in the feature vector it is present {'right_x' : 1} where *x* is the considered character. The purpose of this feature is simply stressing the fact that the

considered character is (or is not) an *a*, a *s* or an *o*. The choice of these letters, and the choice of *only* these letters, derives from the analysis of some stats about the English language available at http://scottbryce.com/cryptograms/stats.htm. Moving the attention on long morphs that are more difficult to detect than the short systematic ones, like *-ed*, *-ing* or *pre-*, since long morphs are often complete words it is possible to exploit stats on words. In particular looking at the lists of the most common first and last letter of a word and considering the general frequencies of letters in the English language it is possible to see that *o* and *a* are the second and third most common first letters, while *s* is the second most common last letter in a word. Why exclude *t*, the most common first, or *e* the most common last letter in a word? Simply performance is worse, an explanation can be that letter *t* is also the third most common last letter in a word and so it has a poor discriminative power. Concerning letter *e*, it is extremely frequent in the English language and so it has a poor discriminative power too.

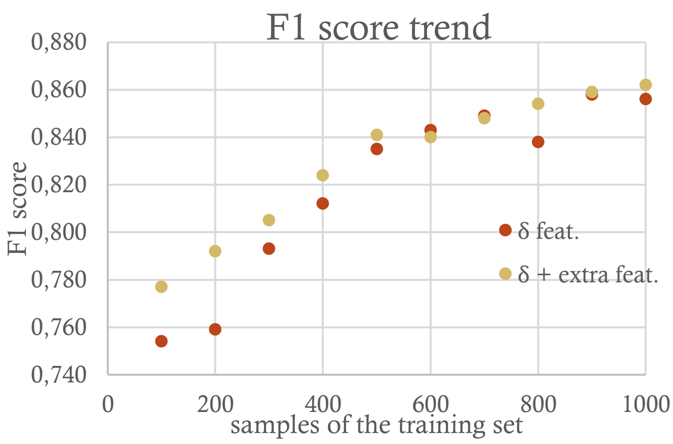
Table 1: only standard delta features

N	δ	ε	max_it.	Prec.	Rec.	F1
100	5	1e-07	160	0.832	0.689	0.754
200	3	1e-07	160	0.828	0.7	0.759
300	7	1e-05	160	0.86	0.735	0.793
400	6	0.001	120	0.862	0.767	0.812
500	9	0.001	120	0.906	0.773	0.835
600	7	1e-07	160	0.906	0.787	0.843
700	9	1e-07	120	0.889	0.812	0.849
800	9	1e-05	120	0.885	0.796	0.838
900	5	0.001	160	0.899	0.821	0.858
1000	8	1e-05	160	0.897	0.818	0.856

Table 2: standard delta features + extra features

N	δ	ε	max_it.	Prec.	Rec.	F1
100	9	1e-07	120	0.839	0.723	0.777
200	9	0.001	120	0.888	0.715	0.792
300	6	1e-05	80	0.893	0.734	0.805
400	7	1e-07	80	0.892	0.766	0.824
500	8	1e-05	80	0.905	0.786	0.841
600	9	0.001	160	0.902	0.786	0.84
700	7	1e-07	120	0.901	0.801	0.848
800	4	0.001	80	0.89	0.821	0.854
900	9	1e-05	120	0.906	0.817	0.859
1000	6	0.001	80	0.907	0.821	0.862

Results are in general satisfactory, with a F1 score near 0.80. As expected with the increasing of the training data the morphological segmentation performs better. Both the Precision and the Recall grow but it is especially the latter that benefits from the increase of the training data, intuitively this behaviour can be justified by the nature of the two scores that measure respectively wrong and missed boundaries. Notice that the Precision remains near 0.8 even considering only 50 training samples while the Recall falls down to 0.64. Not surprisingly it is hard to extrapolate a trend for the epsilon and max iteration parameters, their effect on the final score is always marginal. More surprisingly also the delta seems to have not a recognizable trend, it assumes all the possible values of the search interval in a non-regular way.



The results obtained with the extra features are slightly better, especially with few samples where improvement on the F1 score is up to a 3%. This result led to a natural question: Why choose only two extra features? Add simply more features as possible is not a good strategy, even if all are related with known useful patterns. The fact is that a new feature could move the weights of the learning algorithm from a more useful feature to a less useful one, causing worse results. For example it seems to be an immediate choice add the Right index of the letter as third extra feature, but during the development this tris of extra features performs always worse on the dev set than the presented couple, that instead correspond to the best configuration of extra features tried.

Finally are reported the results obtained with the combined training set. The F1 score is better as expected. Remembering the previous considerations about the Recall score, notice the consistent improvement with 1700 more entries. On the other hand the Precision falls a bit, this could be related with the lower quality of the crowdsourced dataset. Precision is more sensitive on the quality of the dataset, a wrong morphological segmentation in the training set cause more errors than misses.

N	δ	ε	max_it.	Prec.	Rec.	F1	features
2702	9	0.001	80	0.885	0.849	0.866	δ
2702	9	1e-05	80	0.885	0.847	0.866	δ+extra