NLP assignment

Jacob Jonsson, Diego de Rochebouet, Dinara Veshchezerova

March 30, 2018

1 HMMs, PCFGs and Dual Decomposition

1.1 POS tagger

For the POS tagger, a problem which we had to overcome was how to deal with words not appearing in the emissions file. There are multiple approaches to do this, but we chose to assume that the unknown word is a noun. This is not correct for every case, but nouns are the largest word class and it is reasonable to assume that words not recognized before are nouns.

We first tested for one string out of the dev set, and when finding the correct result, we tested the full dev set. Doing so, we got an F1-score of 86%.

1.2 Context free grammar parser

We had issues with the pcfg implementation, for which we used the design from the previous lab in the class. Our running time was slow (over 24h for all sentences). Another issue was that the most probable tree found does not usually have the 'S' root. However, when 'S' was a possible root, most of the time the returned tree matched the correct one. We decided to still return the tree with the highest probability to stay correct to the task. Further, for several of the sentences, especially the longer ones, no root was found. We don't know why and have no good solution, so we returned the tree (NO_ROOT) for those. This significantly lowers the possible score, as about 40% sentences did not yield a root.

Due to the long running time, we were unable to run the code for all dev sentences. For the ones we ran, we got an F1-score of about 60% when we chose the 'S' root, but about 45% when using the most probable tree. For our submission, we still used the most probable root according to the algorithm. We believe the scores are good considering 40% of the sentences were deemed ungrammatical by our algorithm.

1.3 Dual decomposition

We did not successfully implement the dual decomposition.

2 Named Entity Recognition

For the named entity recognition task, the code provided by the course was upgraded by adding features and then by modifying the model.

2.1 Constrained

Multiple features were designed. We first started by designing the features in the chapter 21 of Jurafsky and Martin's book:

- identity of wi
- identity of neighboring words
- part of speech of wi

- part of speech of neighboring words
- wi contains a particular prefix (from all prefixes of length less than 4)
- wi contains a particular suffix (from all suffixes of length less than 4)
- wi is all upper case
- word shape of wi
- word shape of neighboring words
- short word shape of wi
- short word shape of neighboring words
- presence of hyphen

The feature "base-phrase syntactic chunk label of wi and neighboring words" was not implemented.

The last feature mentioned in the book is the use of a gazetteer. For this, we had to search online and found one at http://download.geonames.org/, a gazetteer that was in the list of gazetteer from wikipedia. This gazetteer provided us with a list of names and aliases for places in the whole word, but also with categories for this places. The use of a gazetteer improved our score and we were surprised that the score didn't go even higher.

Other features computed where:

- the word in lowercase
- how many times the word repeated in the previous 50 words or next 50 words in the corpus. This is because analyzing the corpus we saw that the proper names were repeated often
- combination features, were the shape of the word is concatenated with the shape of the neighbor
- a feature telling if the word is the beginning or the end of the sentence

Further, we changed the number of neighbors to see the 3 last neighbors and the next three neighbors when computing all the previous features.

The adaboost classifier and the random forest classifiers were tried but both yielded worse results.

2.2 Unconstrained

For the unconstrained results, a different model was tried, the CRF from sklearn_crfsuite. This model improved our score significantly. This is because in the perceptron model, we could not find a way to tell the model that after predicting for example B-ORG, we couldn't have I-ORG. Because everything was predicted in the .predict() method of the classifier, we could not use the previous prediction as features. But the CRF has transitional probabilities and this allow for those errors to be prevented.

2.3 Conclusion

Overall we found that every feature was really important, and that using a gazetteer, even if they improve the score, are not magical solutions that solve the problem automatically. After reading many papers, many interesting methods were found, like stacking models or using more complex features. This was not done because of a time constraint but it would have been really interesting to test them.