Assignment 3: Sequence Labeling

Natasa Farmaki - DS3517018

Dimitris Georgiou - DS3517004

Stratos Gounidellis - DS3517005

Course: Text Engineering & Analytics

Professor: Ion Androutsopoulos

Athens, March 2018

INTRODUCTION

In natural language processing, a common task is to assign a label to each word. In this assignment, the labels we assign are part of speech tags. More specifically, given a sentence we assign a part-of-speech (POS) tag (noun, verb, etc.) to each word. This task becomes complicated due to the fact that some words are ambiguous: for example, short can be an adjective (short vowel), a noun (direct a short), an adverb (to throw a ball short) or a verb (to short an appliance). So, figuring out which POS is the correct one depends on the context, including the POS tags of the neighboring words. In order to classify each word to a tag we will use a baseline classifier, Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs).

The python code is available in the following link: https://thinkingtea.github.io/repoTEA/

DATA DESCRIPTION

The data used to train and test the classifiers are in the form of treebanks. In linguistics, a treebank is a parsed text corpus that annotates syntactic or semantic sentence structure. For example, a sentence in a tree bank is presented as follows:

```
# text = What if Google Morphed Into GoogleOS?
1   What    what    PRON    WP  PronType=Int    0   root    0:root  _
2   if  if  SCONJ   IN  _   4   mark    4:mark  _
3   Google  Google  PROPN   NNP Number=Sing 4   nsubj   4:nsubj _
4   Morphed morph   VERB    VBD Mood=Ind|Tense=Past|VerbForm=Fin    1   advcl   1:advcl _
5   Into    into    ADP IN  _   6   case    6:case  _
6   GoogleOS    GoogleOS    PROPN   NNP Number=Sing 4   obl 4:obl   SpaceAfter=No
7   ?   ?   PUNCT   .   _   4   punct   4:punct _
```

As tags we consider the fourth "column" (i.e. the xpostag) of the assumed "table" formed above.

DATA PRE-PROSSESING

In order to use the aforementioned classifiers, we need to transform the data in an appropriate form. Firstly, we read the file which includes the tree banks (.conllu files) using the conllu library. We split the file using the new line character. We remove lines which start with the pattern '# sent_id' and '# newdoc id'. The line which starts with the pattern '# text' includes the whole sentence which we use as key in the dictionary. During parsing the files with the conllu library we bump into issues with specific tokens. To cope with this problem, we read the tokens manually and characterize them with their POS tag. The above issue appears in 30 tokens in both training and test set.

After parsing the files, we construct two dictionaries, one for the test set and the other for the training set. The dictionaries have the whole sentence as key and a list of tuples as a value. On the first position of each tuple there is a token and on the second its POS tag.

```
('What if Google Morphed Into GoogleOS?',
 [('What', 'WP'),
  ('if', 'IN'),
  ('Google', 'NNP'),
  ('Morphed', 'VBD'),
  ('Into', 'IN'),
  ('GoogleOS', 'NNP'),
  ('?', '.')])
```

POS TAGGERS

As mentioned above we will use 3 classifiers to find the corresponding tags for each word.

Baseline approach
The baseline algorithm is the simplest approach to classify tokens to tags. More specifically we find for each token the most frequent tag and store it in a dictionary (token is the key, tag is the value). Therefore, for each token in our test set we find the corresponding tag from that dictionary and classify it accordingly. We also find the most frequent tag of the training set and assign it to tokens that we have no information about as they do not appear in our training set. In our case the most frequent POS tag is “NN”.

HMMs
The Hidden Markov Model (HMM) is a powerful statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence. HMMs have found application in many areas interested in signal processing, and in particular speech processing, but have also been applied with success to low level NLP tasks such as part-of-speech tagging, phrase chunking, and extracting target information from documents. The state is directly visible to the observer, and therefore the state transition probabilities are the only parameters, while in the hidden Markov model, the state is not directly visible, but the output (in the form of data or "token" in the following), dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states, this is also known as pattern theory, a topic of grammar introduction.

CRFs
Conditional random fields (CRFs) are a probabilistic framework for labeling and segmenting structured data, such as sequences, trees and lattices. The underlying idea is that of defining a conditional probability distribution over label sequences given a particular observation sequence, rather than a joint distribution over both label and observation sequences. The primary advantage of CRFs over Hidden Markov models is their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference. According to research results, CRFs outperform HMMs on several real-world tasks in many fields, including bioinformatics, computational linguistics and speech recognition. Another advantage of CRFs over HMMs is that they use as input multiple features instead of just a token. In our case we extracted information about word identity, word suffix and word shape and formed a feature vector.

EVALUATION METRICS

In order to evaluate the results of the multiple classifiers and therefore compare their performance we need to implement appropriate measures. The metrics that can holistically evaluate performance are accuracy, recall, precision and F1-score (produced by both recall and precision). In this problem except from comparing the classifiers we are interested in deciding whether collecting more training instances will improve performance of the classifiers. For that reason, we also construct learning curves based on accuracy and on F1-score.

Accuracy is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage.

Precision is the number of True Positives divided by the number of True Positives and False Positives. Precision can be thought of as a measure of a classifier’s exactness. A low precision can also indicate a large number of False Positives.

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.

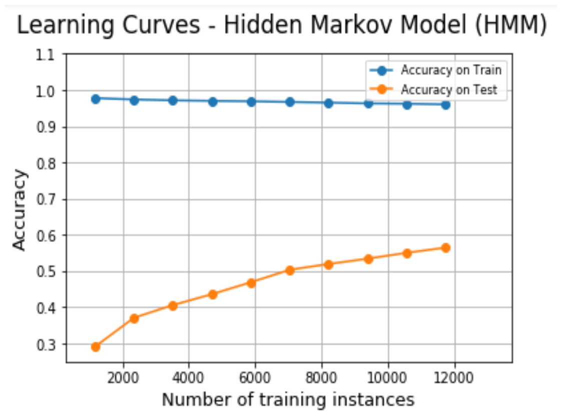
F1-score is the 2*((precision*recall) / (precision + recall)). Put another way, the F1 score conveys the balance between the precision and the recall.

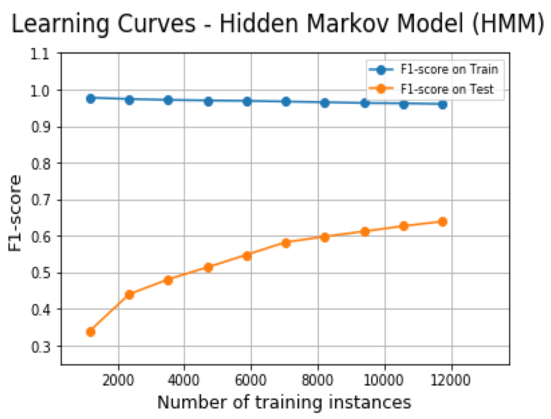
Learning Curve is the representation in graph form of the rate of learning something over time or repeated experiences. In a text classification problem, a learning curve shows whether collecting more training instances will improve the performance of the classifier both in training and test set. It is important to note that learning curve is not useful for model assessment.

EXPERIMENTAL RESULTS

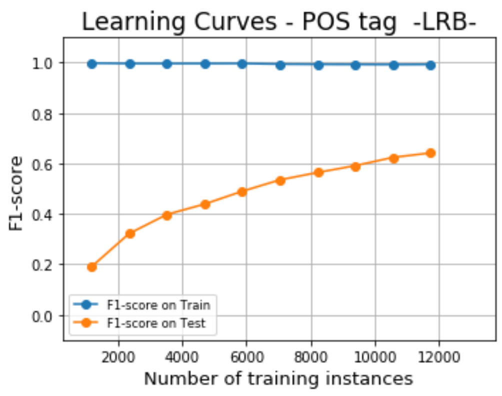
HMMs

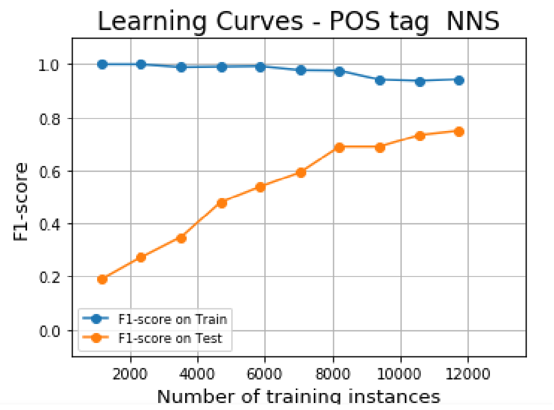
The following graphs show the learning curves with the average F1-score and accuracy for the HMMs POS tagger.



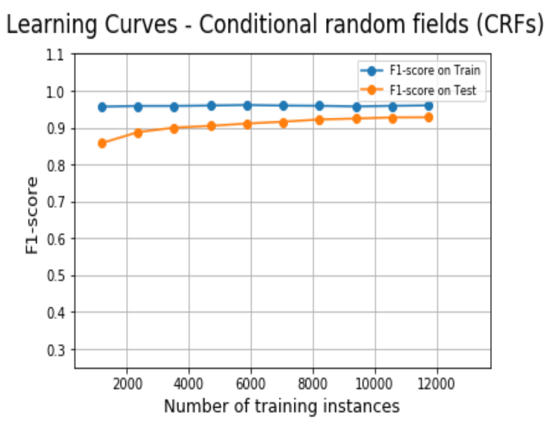


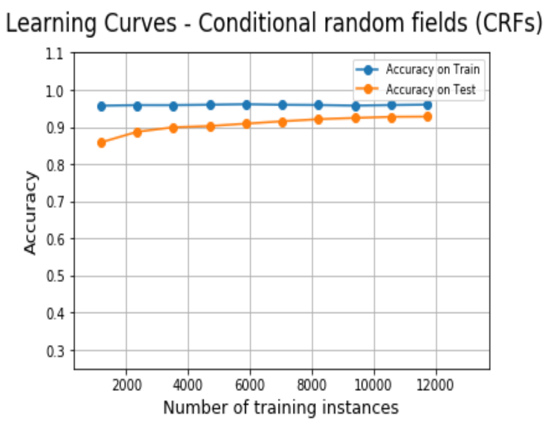
Then, we indicatively present two learning curves from the total forty-nine learning curves, which depict F1-score separately per POS. The rest learning curves are available in the source code.



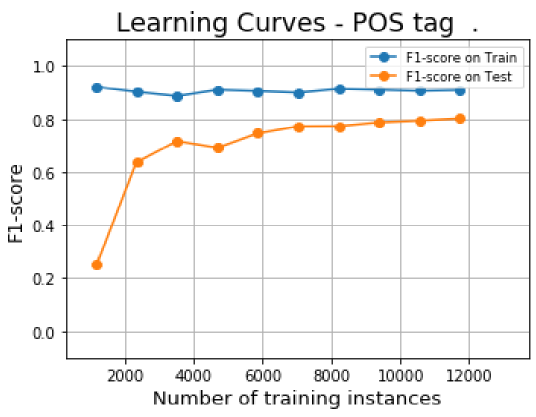


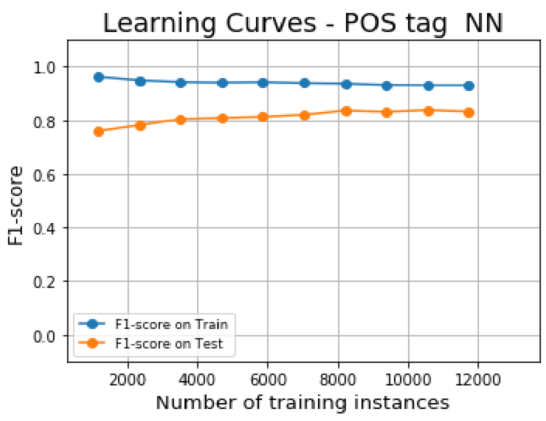
CRFs
The following graphs show the learning curves with the average F1-score and accuracy for the CRFs POS tagger.



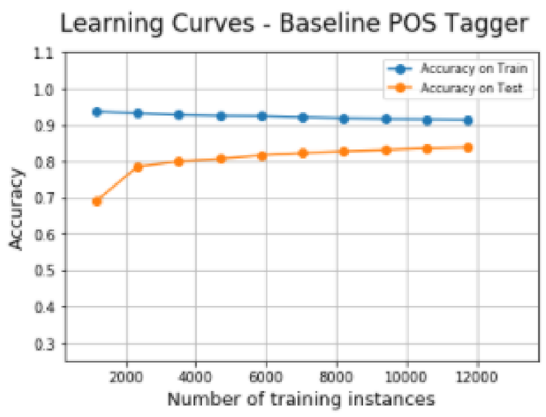


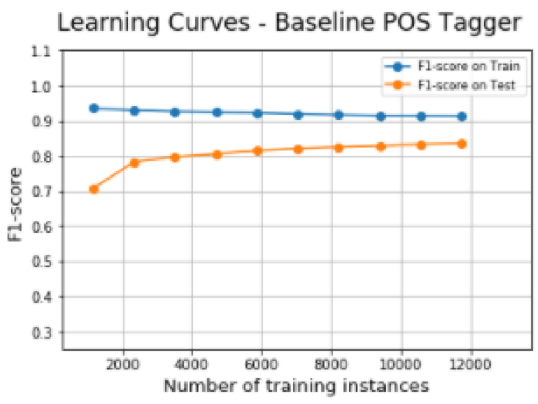
Then, we indicatively present two learning curves from the total forty-nine learning curves, which depict F1-score separately per POS. The rest learning curves are available in the source code.



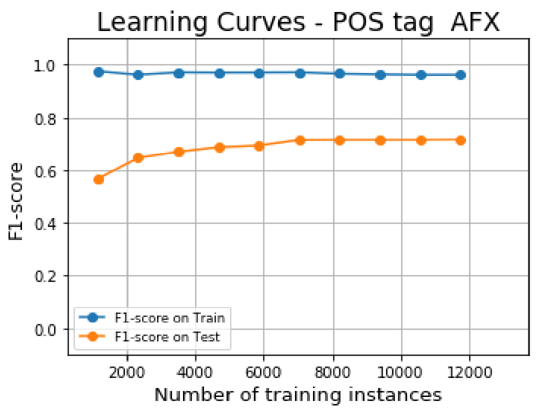


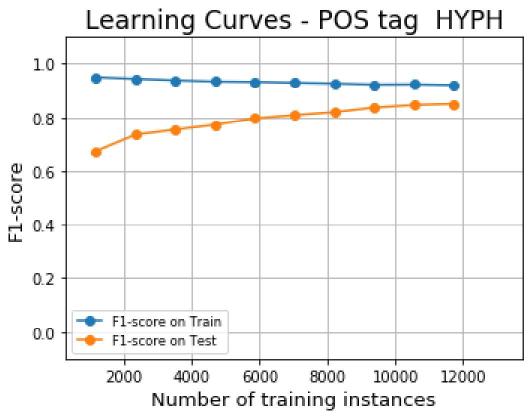
Baseline
The following graphs show the learning curves with the average F1-score and accuracy for the baseline POS tagger.





Then, we indicatively present two learning curves from the total forty-nine learning curves, which depict F1-score separately per POS. The rest learning curves are available in the source code.





CONCLUSIONS

While comparing the baseline approach with HMM and CRFs we observe that HMM does not perform better. On the contrary, CRFs perform significantly well because as mentioned in the theory that approach takes into account previous words and uses features instead of just tokens.

More analytically, for the **Baseline tagger** the performance on the test set is:
Average Accuracy: 0.83835
Average F1-score: 0.83534

For the **HMM tagger** the performance on the test set is:
Average Accuracy: 0.56447
Average F1-score: 0.63918

For the **CRF tagger** the performance on the test set is:
Average Accuracy: 0.92763
Average F1-score: 0.92764