CMPUT 501 Assignment 3

Afia Anjum and **Shehroze Khan**
Department of Computing Science
University of Alberta

1 Parts of Speech Tagger

In order to train and test the given files with different Parts of Speech taggers such as the Stanford POS tagger, Hidden Markov Model(HMM) tagger or the Brill's Tagger, we pre-process our training and testing files as our first step.

As per the documentation of Stanford POS Tagger, we created property file specifying values of different properties in order to train our file. One such property is the 'tag separator' which we specify as an underscore. As a part of pre-processing, we replace any spaces between a word and a tag in our train and test files with an underscore as well, while implementing this tagger. This is done so to follow the 'tag separator' property of our generated property file.

We use the following two commands to train and test our given files using the Stanford POS Tagger. For training we use the comand "java -classpath stanford-postagger.jar edu.stanford.nlp.tagger.maxent.MaxentTagger -prop proertiesFile -model modelFile -trainFile trainFile". And for the testing we use the command "java -classpath stanford-postagger.jar edu.stanford.nlp.tagger.maxent.MaxentTagger -prop propertiesFile -model modelFile -testFile testFile".

We pass our generated properties file mentioned above, path for the generation of our model file and our training file through the command line with our first command. This command generates an output properties file and an output model file, which we pass in the command line along with our test file for testing like the second command. The testing provides us accuracy in the format (Total sentences right, Total tags right and Unknown words right).

During testing, we ensure that we remove the "trainFile" property from the properties file generated from training and then execute the command.

The pre-processing in case of HMM and the Brill's Tagger is done in the similar manner. We create list of sentences for both our train and test files. Each of the words of each of the sentences are represented as a tuple of a word and a tag separated by a comma.

We then use the train_supervised() method from the HiddenMarkovModelTrainer() class in order to train a model and evaluate it's performance with the evaluate() function of the same class.

For the Brill's tagger, we can use the Stanford POS Tagger or the HMM Tagger as it's baseline and then compute accuracy training with the Brill's tagger. In our implementation, we have used to train() function and evaluate() function from the BrillTaggerTrainer class to train and evaluate our models with the Brill's Tagger.

2 Parameter Tuning

The baseline accuracy score for the HMM Tagger were very low, in the range of about 26-30%. This is because NLTK's HiddenMarkovModelTrainer class by default uses the Maximum Likelihood Estimate to compute the probability scores for a given sentence and the PoS tags of its constituent words.

In order to fix this issue, we utilized the Lidstone Probability Distribution [1] as the estimator for our HMM tagger. This probability distribution is parameterized by the the real number $\gamma \in (0, 1)$. This was the parameter that we tuned to achieve higher accuracy scores. The value of $\gamma = 0.1$ gets us good results. We tried to increase this value up to 0.9, but stuck with 0.1 because of diminishing accuracy scores.

The Brill's tagger contains a train() function parameterized with the the property 'max_score', which can be tuned. While training, it produces at most max_rules transformations to reduce net

[1] https://www.nltk.org/api/nltk.htmlnltk.probability.LidstoneProbDist

number of errors in the training set. So, we can change this max_score values for faster computation. However, the value should not be too low.

3 Error Analysis

As a part of error analysis, we have extracted accuracy scores for each of the taggers trained on the given training files(Domain1train and Domain2train) and test on the given testing files(Domain1test and Domain2test). The opposite tests are also done and accuracy scores are noted.

Using the Stanford POS Tagger we find the following result:

TrainFile	TestFile	Accuracy
Domain1Train	Domain1Test	87.17%
Domain2Train	Domain2Test	88.23%
Domain1Train	Domain2Test	86.26%
Domain2Train	Domain1Test	87.05%

Using the HMM Tagger we find the following result:

TrainFile	TestFile	Accuracy
Domain1Train	Domain1Test	84.84%
Domain2Train	Domain2Test	85.58%
Domain1Train	Domain2Test	80.74%
Domain2Train	Domain1Test	79.79%

Using the Brill's Tagger with HMM as the baseline tagger we find the following result:

TrainFile	TestFile	Accuracy
Domain1Train	Domain1Test	85.37%
Domain2Train	Domain2Test	86.10%
Domain1Train	Domain2Test	81.46%
Domain2Train	Domain1Test	80.47%

Using the Brill's Tagger with Stanford as the baseline tagger we find the following result:

TrainFile	TestFile	Accuracy
Domain1Train	Domain1Test	88.45%
Domain2Train	Domain2Test	90.8%
Domain1Train	Domain2Test	86.12%
Domain2Train	Domain1Test	91.12%

There are about 7000 incorrectly classified tokens per output file for the 14 output files of the HMM and Brill's Taggers. After having a general, high level look at these files, we conjecture as a result of our observations that most of these words are out-of-vocabulary words. This is because the misclassifications do not have any preference for a specific PoS tag; these taggers tend to misclassify OOV words with different kinds of PoS tags. This phenomenon explains the reported accuracy scores, which tend to drop for taggers trained in a different domain to the one they are tested on.

4 Learner English

In this part, we compute accuracy for one of the given test file from the English Learner Language with all our models.

Using the Stanford tagger Model to test the ELLTest file we find the following result:

TrainFile	TestFile	Accuracy
Domain1Train	ELLTest	81.62%
Domain2Train	ELLTest	81.86%

Using the HMM tagger to test the ELLTest file we find the following result:

TrainFile	TestFile	Accuracy
Domain1Train	ELLTest	83.41%
Domain2Train	ELLTest	82.82%

Using the Brill's tagger(with HMM as baseline) to test the ELLTest file we find the following result:

TrainFile	TestFile	Accuracy
Domain1Train	ELLTest	83.79%
Domain2Train	ELLTest	83.46%

And in the next part, we generate accuracy by training a model with an English Learner Language train file and testing it with a file from the same domain.

Using the Stanford POS tagger we compute accuracy for the the ELL files as follows:

TrainFile	TestFile	Accuracy
ELLTrain	ELLTest	52.57%

Using the HMM tagger we compute accuracy for the the ELL files as follows:

TrainFile	TestFile	Accuracy
ELLTrain	ELLTest	92.21%

Using the Brill's tagger(with HMM as baseline) we compute accuracy for the the ELL files as follows:

TrainFile	TestFile	Accuracy
ELLTrain	ELLTest	92.93%

The provided ELL train and test files contain non-standard English sentences. Only the Stanford POS tagger is able to determine that it's a non standard English with its low accuracy for the ELL test data, since it is trained on Standard English sentences.