# 189G_NLP_HW2 REPORT

Sairamvinay Vijayaraghavan

May 9, 2019

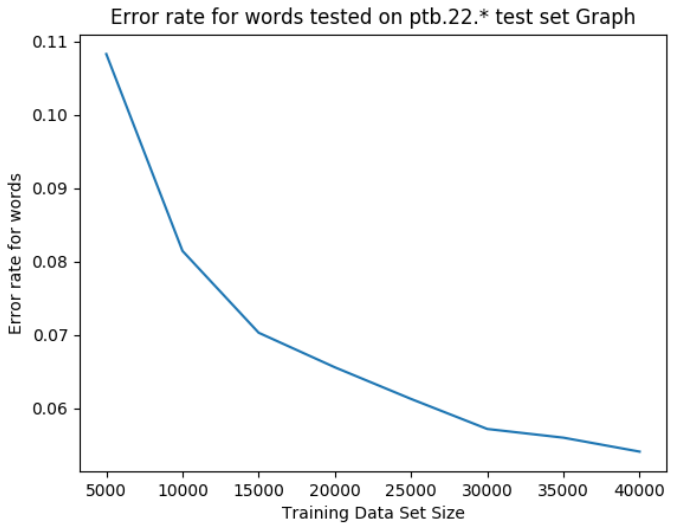
## 1 Task 1

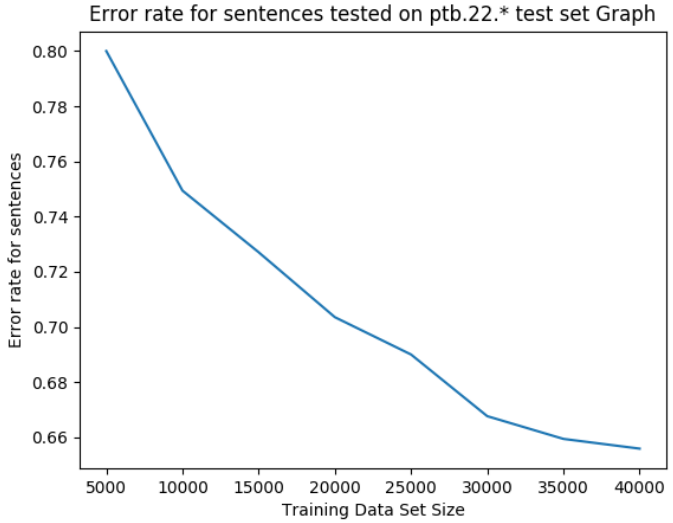
In this task, I had to train the bigram HMM model based on the training data set from the Penn Tree Bank. I was only supposed to use the ptb.2-21.txt and ptb.2-21.tgs files. In order to check the effect of the amount of training data on the performance, I had devised a quick method to implement the same.

First, I had picked certain fixed numbers X as the length of the training data set, such as 5000,10000,15000,20000,25000,30000,35000,40000. Now, before I trained my HMM, I had made some changes into the train_hmm.py where I had trained the data set only on the first X lines of data which was used to train the data set. Here X belongs to the fixed numbers which are 5000,10000,15000,20000,25000,30000,35000,40000.

Now, I had plotted two graphs with the increasing size of the training data set with the sentence error rate and the word error rate of the test data set. The learning curves are plotted as follows:

I got the following graphs as the results for the various sizes:






As we notice, the graphs represent the fact that with increasing size of the training data sets, we would get more POS tagged data to learn from and hence the predictions improve which can be observed through lesser error rates per sentence and word. Hence, with more data to learn from, the algorithm learns more and hence it would be able to predict better.

So by having a larger corpus to learn from, the system would improve much more with a bigger data set to learn from.

## 2 Task 3

The bigram HMM model yielded low error rates per word and sentence. However, in order to improve our model from the data set, I had re-developed a new HMM model. I had used a trigram HMM model with add-1 smoothing, in which the tags are trained more like a trigram HMM model. In this case, the current tag (state of the word) depends on the previous 2 tags of the sentence.

So, I had re-written the entire train_hmm.py and now here, I stored my transition Probabilities as a dictionary (that was the only major change done, emit probabilities are not changed at all). Here, the keys of the dictionary was a tuple containing the previous two tags and the current tag. The transitions are recorded as two dictionaries. First, the trigrams are all recorded with their counts while the previous two tags were recorded in another dictionary as the key which recorded the count of the bigrams which was later used in calculating the probability. The formula used for recording the trigram probabilities was as follows:

$$P(Tag_i|Tag_i{-}1, Tag_i{-}2) = [Count(Tagi, Tag_i{-}1, Tag_i{-}2){+}1]/[Count(Tag_i{-}1, Tag_i{-}2){+}V]$$

Where V is the number of unique bigrams of tags present in the data set.

Now I had then changed my viterbi algorithm accordingly, to accommodate the changes to the HMM model. In this case, the trigram model was now used to record the transmission probabilities from the previous 2 tags to another tag. So I used this model and now ran the viterbi algorithm on the ptb22.* data set which recorded results as follows:

error rate by word: 0.0515492185357828 (2068 errors out of 40117) error rate by sentence: 0.634117647058824 (1078 errors out of 1700)

```
campus-015-196:hw2_189 sairamvinayvijayaraghavan$ ./tag_acc.pl ptb.22.tgs my3.out
error rate by word:      0.0515492185357828 (2068 errors out of 40117)
error rate by sentence:  0.634117647058824 (1078 errors out of 1700)
campus-015-196:hw2_189 sairamvinayvijayaraghavan$
```

Clearly, this model outperformed the baseline bigram model and hence, it did a better job. The add 1 smoothing was done because, without the smoothing, my trigram model was being biased towards certain tags in particular and was highly uneven. Hence, in order to smoothen the model, the Laplace add 1 smoothing was performed which did a good prediction.

My results on the ptb23.* data set is included

# 3 Task 4

I had now run my implementation on the other languages development set.

## 3.1 The results for the Bulgarian data set for my model and the base line

The results for the baseline bigram model on the Bulgarian data set:

```
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$ ./viterbi.pl btb_base.hmm < btb.test.txt > btb_base.out
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$ ./tag_acc.pl btb.test.tgs btb_base.out
error rate by word:      0.115942028985507 (688 errors out of 5934)
error rate by sentence:  0.751256281407035 (299 errors out of 398)
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$
```

The results for my trigram model on the Bulgarian data set:

```
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$ ./tag_acc.pl btb.test.tgs btb_mymodel.out
error rate by word:      0.113077182339063 (671 errors out of 5934)
error rate by sentence:  0.751256281407035 (299 errors out of 398)
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$
```

## 3.2 The results for the Japanese data set for my model and the base line

The results for the baseline bigram model on the Japanese data set:

```
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$ ./tag_acc.pl jv.test.tgs jv_base.out
error rate by word:      0.0628611451584661 (359 errors out of 5711)
error rate by sentence:  0.136812411847673 (97 errors out of 709)
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$
```

The results for my trigram model on the Japanese data set:

```
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$ ./tag_acc.pl jv.test.tgs jv_mymodel.out
error rate by word:      0.0460514796007704 (263 errors out of 5711)
error rate by sentence:  0.208744710860367 (148 errors out of 709)
sairamv@ad3.ucdavis.edu@pc41:~/Desktop/hw2_189$
```

## 3.3 Explanation about the difference in performance with respect to models

### 3.3.1 Bulgarian data set

Overall, the results of both these models are not very different on this data set. In the bulgarian dataset, my trigram add one smoothed model works slightly better than the baseline model. However the error rates are indeed quite high in both the models. We can analyze this performance with respect to the vast difference in tokenizing English words and Bulgarian words as Bulgarian alphabets are not the same as the English alphabets and hence there is a high error rate.

### 3.3.2 Japanese data set

The results in the Japanese seem to be much better than the Bulgarian data set. My trigram model works much better in this case with respect to the baseline model. My model does better because the add 1 smoothed trigram offers much more accuracy in predictions as the probabilities are now more accurate with respect to the transition in tags (it is dependent on the previous two tags) while the baseline implements a bigram model in which probability of current tag depends solely on the previous tag and hence there may be lesser accuracy in this case.

## 3.4 Explanation about the difference in performance with respect to languages

English performs the best, followed by Japanese and then Bulgarian for both the models used. This is because the HMM models, trained in Python, use mainly ASCII characters in a string. Since Japanese uses more of the English alphabets than the more foreign Bulgarian, Japanese does better. Although my model predicts a lower error by word rate for the Japanase dataset, the error by sentence is very high because, it fails to find the correct tag sequence overall

because of the different kinds of non-ASCII characters used in the training data set.

Bulgarian language uses more foreign characters (non-ASCII) which would have been difficult to represent and hence the performance was worse on both the models.

Japanese language uses more ASCII characters which makes it easier to train and hence predict as python strings only uses ASCII Characters. Therefore both the models do better than Bulgarian but however, since English is most familiar language to Python in terms of character usage and alphabet design, it performs the best on both the models

Hence Japan