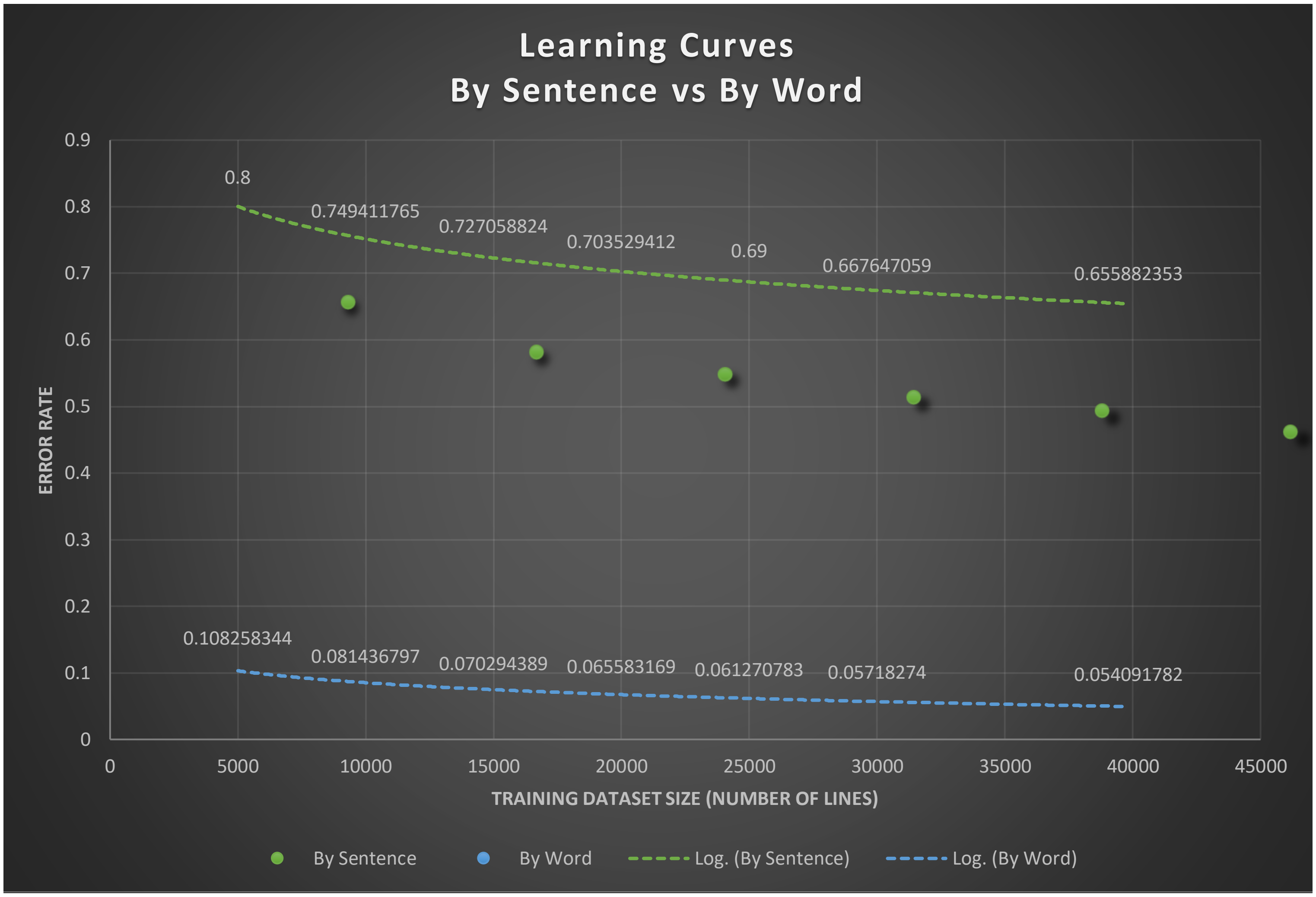
HW 2: POS Tagging

Kaustubh Deshpande

Task 1: I created the plot using Microsoft Excel. I generated various subsets of the 2-21 training dataset. Each of these was of varying lengths. These files can be found in my submission and they include the first x thousand number of lines from ptb2-21. For example, ptb2-21-25.txt contains the text for the first 25,000 lines and ptb2-21-25.tgs contains the first 25,000 tags respectively.



Looking at the curve, I understand that more training data produces lower error rates, which means better results. However, the results level off very quickly. The reason for this is that because HMMs are simple models and the performance levels off soon despite increase in data. If we were using a large neural network, we would not see leveling off, even if our data set was small.

Error can possibly occur because of two reasons:

i). The dataset is trained on small data set. In this scenario the model will encounter more words out of vocabulary words. This will cause reduction in its performance and high error rate during evaluation.

ii). High error rate can also be caused because of poor model. I think this is more important as HMM is a pretty simple model.

In conclusion, the amount of data to train the model is important for model's accuracy. More data and more words in vocabulary used for analysis causes reduction in the error of facing OOV_Words.

Task 2:

I tried multiple methods to improve the baseline model. I tried to implement trigram and different combinations of smoothing etc. I also tried to use the hint provided by the instructor "*replaced every tag by its first letter only, the tags would still be meaningful, only more coarse*". However, for any of these changes, it required me to create a new Viterbi file in python as the perl Viterbi file had too many errors and could not handle my new model. I was also required to create a post-processing script that runs after the viterbi.pl. Since I am not familiar with perl I could not create a Viterbi python file on my own. So, I decided to make a simple modification to the baseline model. Eventually, I performed additive smoothing. During this process I only modified the transition probabilities. I did not change the emission probabilities. I did this because some transitions could possibly have more zeros. So additive smoothing helps performance. The baseline model provided by the professor had an error rate by word of 0.0540917815389984 and an error rate by sentence of 0.655882352941176. However, after performing additive smoothing my model achieved an error rate by word of 0.054017000274198 and an error rate by sentence of 0.655294117647059.

In the folder that I have submitted the name of the modified model is 'modified_train_hmm .py'. I ran the script on my **modified model** to obtain **my.hmm**. I then ran viterbi on pb.23.txt using the **my.hmm** file obtained in the previous step. The output file is called 'my.out' which may be used to run tag_acc.pl along with the pb.23.tgs.

Task 3:

	Baseline HMM	Modified HMM
Japanese	Error by Word: 0.062861145	Error by Word: 0.063211347
	Error by sentence: 0.136812412	Error by sentence: 0.136812412
Bulgarian	Error by Word: 0.115942029	Error by Word: 0.115773509
	Error by sentence: 0.751256281	Error by sentence: 0.748743719

The table above shows the error rate I obtained after running the necessary tests on Japanese and Bulgarian training sets. For the Japanese training sets, the baseline model given to us performs superior than the modified HM model I created for task 2. My model shows a higher error rate by word, but the error rate is the same for sentence. On the other hand, for the Bulgarian language data sets, my modified HMM model performs superior and produced a lower error rate for words and sentences. This can be explained by the fact that the new model smoothens the transition probabilities.

Japanese is very different from English and as a result has different transitions. The Japanese transitions for different POS and words are vastly different compared to English. As a result, smoothening the transition probabilities for Japanese made the model performs poorly. Additionally, I think that another reason for this poor performance can be over fitting of the tags. However, Bulgarian is a Slavic language. I referred to the following link to get this information.

https://www.quora.com/How-similar-are-Bulgarian-and-English

The link showed me that Bulgarian is a Germanic/Romance language and thus much similar to English than compared to Japanese. Therefore, the additive smoothing model performs better as the transition between words and POS are similar between English and Bulgarian. Since English is the language used to validate, it makes sense that the model works well for Bulgarian.