Name: Soonho An
Andrew ID: Soonhoa

Algorithms for NLP

Homework 1

1. Statement of Assurance

"I certify that all of the material that I submit is original work that was done only by me."
<Soonho An>, <Mar 10, 2020>

2. Files

The files I zipped and uploaded are stated below.
<writeup.pdf, tagger.py, ptb.23.tgs, train_hmm.py, learning_curve.py, my_tag_acc.py, eval.txt>

3. Task 1: Generating Learning Curves

3.1 Generating and evaluating a bigram HMM model

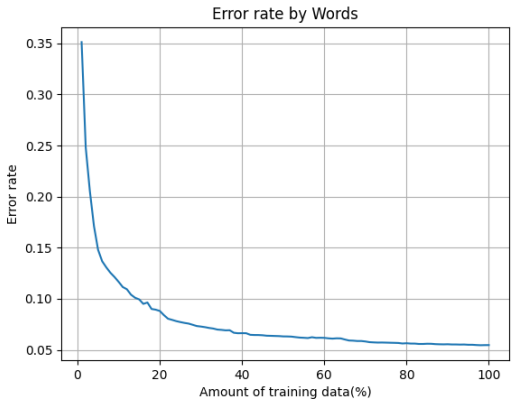
I made a python file "learning_curve.py" which utilizes train_hmm.py, viterbi.py, and tag_acc.py to get the results and plot the curves I wanted to draw.

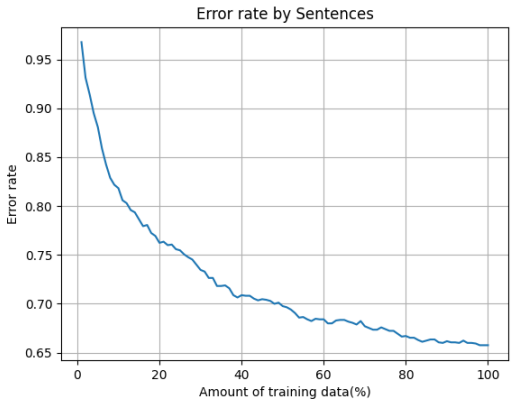
3.2 The Curve

1) I first pooled the training set by various portion. I mixed the order of the lines from training set and divided into 100 samples. With this method, I pooled the training set from 1% to 100% gradually.

2) Then I created HMM models for each pooled training set, resulting in 100 different models.

3) Finally I accumulated all the results(error rates) and plotted it.





3.3 Analysis

1) How does the size of the dataset impact the performance of the model/system?

From the learning curve I've generated, I can see that there's a specific relationship between the size of the dataset and the performance of the model. In case of the "Error rate by Words", as the portion of the training set increases, the error rate drops really fast and seems almost saturated at the end. On the other hand, "Error rate by Sentences" seems a little bit more fluctuating than "Error rate by Words". Also "Error rate by Sentences" seems like it still has potential to be dropped more if we had more data than 100%.

2) How do you think the learning curve will change for datasets of different languages? There is no single 'correct' answer to this question, but it has to make sense!

I'm not sure if this question means just for "datasets of languages other than English" or "mixed datasets of different languages", I'll just take it as the former one because the latter one is actually not predictable for me. I think if the datasets are of other languages, the learning curve can be similar to the previous one or not, depending on what kind of language is it. In case of English, the learning curve shows pretty stable decreasing curve, and I think that's because of the "projectivity" which is explained in Parsing 3 lecture. In the lecture slides, English shows 0.0% of non-projective trees which means English is very strict and sensitive in terms of the position of the words. Thus if the language I'm utilizing is giving a low non-projective trees percentages like English, Chinese or Spanish, the learning curve will be pretty much the same as English. On the other hand if the language I'm utilizing is giving a high non-projective trees percentages like Dutch or German, I can't guarantee the stable decreasing curve for this case, because some of the dataset in those languages must not be helpful at all.

4. Task 2: Building a Better System

4.1 Subtask 1: Your own improved HMM tagger

Actually, I totally failed to modify viterbi.py. Anytime I gave some change in the code like bigram to trigram, smoothing the probability estimates, the error rate goes to be 100% and never dropped. However at the last trial, I gave some change in train_hmm.py, the result got slightly better than before.

4.2 Subtask 2: Analysis

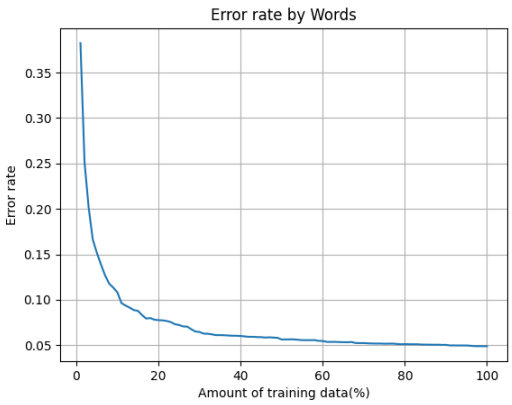
1. What modifications did you make?

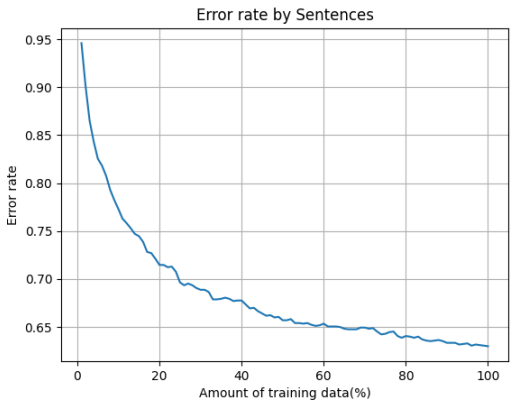
I simply just changed the OOV handling part, by adding the “One time shown words” to the emission matrix. In the original code, if the word is shown only once in the training set, then HMM model just take that word as an OOV case, and doesn’t remember that word itself. The only change I made is just moving the OOV checking line to be few lines below and changed the line “token = OOV_WORD” to be “emissions[tag][OOV_WORD] += 1” so that the probability of OOV is still calculated and the words appear only once are also considered.

2. How much improvement did your new model deliver on section 22 (ptb.22.txt)?

	Original	**My own**	**Difference**
By Words	0.05409	0.04898	**- 0.00511**
By Sentences	0.65588	0.63000	**- 0.02588**

Below is the learning curve I made for my own tagger.





3. Why do you think that these modifications improved the accuracy of this labelling task?

The reason I came up with this method, is because the original code seems like wasting the important information in the training set. Even it’s the words just appear only once in the whole training set, there still is a possibility to those words to be shown in the validation set or the test set. I think that is the advantage of my own HMM model.

5. Bonus Task: Better Evaluation

I made a new evaluation script that calculates the differences between the two different outputs. I just modified the tag_acc.py to get one more argument, and to calculate the differences of error rate and to save it as eval.txt.