**Dieu My Nguyen**
CSCI 5832
Prof. Martin
Due Date: Oct 01, 2018

# Assignment 2

*Part of speech tagging*

*Deliverables: You will need to turn in your code, along with a short report describing what you did and all the choices you made. You should include in your report how you developed and assessed your system (training/dev) and how well you believe it should work on unseen test data.*

**Train & test set split:**
I shuffled the sentences from the berp-POS-training.txt file, and split into the usual ratio of 80% train and 20% test.

**Baseline system:**
For the baseline POS tagger, using the train set created above, I created a sort of look-up table that link a given token, its assigned POS tag(s), and the frequency of each tag given to that token. On the test set, I match an input token to its most frequent tag. With unseen words, I gave them the tag that has the highest frequency across the look-up table. That tag is the period. The baseline system's accuracy is approximately 90%.

**Viterbi system:**
The class "UNK" was created to account for words not in the train set. I obtained the set of tokens from the entire given dataset (berp-POS-training.txt), counted for those not in the train set. I then calculated the transition and emission probability matrices with the train set, also taking into account the UNK counts. Laplacian add-one smoothing was applied to these probabilities.

I then implemented the Viterbi algorithm based on a bigram assumption that the probability of a tag is dependent on the previous tag. Thus, the equation I followed to build a bigram-based system that will output the most probable tag sequence is:

$$\hat{t}_1^n \approx_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

The viterbi algorithm includes a table (in matrix form in code) where columns are the input token sequence and rows are the possible hidden tags. I swept through the table left to right and fill it out using the transition and emission probability matrices. I also kept a backtrace matrix and calculated the best path pointer as a starting point for the backtracing to obtain the most probable sequence of tags.

I assessed the accuracies of my baseline and Viterbi systems by training them on the train set, and then used the 20% test set with ground-truth labels to compare my systems' output sequence of tags. The Viterbi system achieved approximately 95% accuracy, higher than the baseline. I think my algorithm might overall do ok, if I debug it further to make sure. However, sadly, the accuracy on tokens not seen in the training set is as low as could be: 0%. Just as

the baseline system assigns the tag "." to unseen tokens, the Viterbi system also tagged all of the unseens with a period. I might not have implemented the smoothing and UNK counting correctly.

As with most things concerning data files, a big chunk of my time and effort was dedicated to processing the data. So, apologies for giving up on making elegant code.