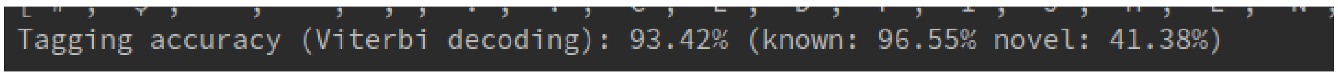
Chris White
Shawn Davidson

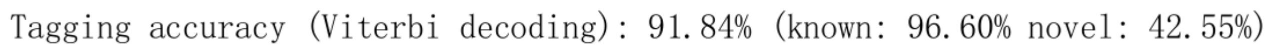
NLP Assingment 5 report
11/12/18

When we used the one-count smoothing technique our results were:



```
Tagging accuracy (Viterbi decoding): 93.42% (known: 96.55% novel: 41.38%)
```

this is a considerable improvement form the baseline tagger which is given below.



```
Tagging accuracy (Viterbi decoding): 91.84% (known: 96.60% novel: 42.55%)
```

With our program we get a higher overall accuracy and roughly the same accuracy of known percentages but suffer a small drop of novel percentages.

To make this program we started by parsing the text. In parsing we created a frequency distribution of all of the tags and using the uni-gram and bi-gram models. We also created frequency distribution of all of the word and tag pairs. (we also did not consider the first start of sentence tag when we made these counts).
After determining initial probabilities we removed the '###' symbol from the distributions to get data that was more representative of the actual text.
We then created transition and emission tables from our probabilities.

We then ran the Viterbi algorithm on our test document to find our tagging accuracy. To smooth the data we used the one-count smoothing model. To calculate the smoothed probabilities for the transition and emission tables, we created dictionaries of singletons for tags and words. We then used the following formula to calculate smoothed probabilities:

$$p_{\mathrm{tt}}(t_i \mid t_{i-1}) = \frac{c(t_{i-1}, t_i) + \lambda \cdot p_{\text{tt-backoff}}(t_i \mid t_{i-1})}{c(t_{i-1}) + \lambda} \text{ where } \lambda = 1 + sing_{\mathrm{tt}}(\cdot \mid t_{i-1})$$

$$p_{\mathrm{tw}}(w_i \mid t_i) = \frac{c(t_i, w_i) + \lambda \cdot p_{\text{tw-backoff}}(w_i \mid t_i)}{c(t_i) + \lambda} \text{ where } \lambda = 1 + sing_{\mathrm{tw}}(\cdot \mid t_i)$$