1 Problem 4 write-up

1.1 Problem 1

Implementation-wise, Problem 1 was straightforward. The tricky part was figuring out features to implement. The $\text{npchunk}_{\text{features}}$ given by the nltk textbook keeps track of things like the current pos tag, the previous word's pos tag, and the subsequent word's pos tag. My bright idea was to also keep track of the pos tag for the word after the subsequent word (the nextnext word). This actually slightly increased the score on all counts. Slightly more creatively, I also thought to keep track of the length of the current word and the previous word. My thinking here was that nouns, especially proper nouns, and adjectives are typically longer then other parts of speech (I dont know if this actually true, but intuitively it seems to make sense), and adjectives generally precede nouns. Given that keeping track of the length of the subsequent word actually decreased precision scores, while the other features increased it, my hypothesis may be in part correct.

1.2 Problem 2

This problem was tricky for me because I worked on it when we no longer had access to $\text{get}_{\text{entity}}$, and I had to extract named entities myself. However, I quickly realized that any word that was not a NE, was paired with the letter 'O,' so extracting named entities was simply a matter of collating all words not paired with O. The other difficult bit was figuring out how to calculate the precision and recall of the nltk chunker. Initially, it seemed like all I had to do was walk through the list of NE for the golden corpus and the list of NEs for the nltk chunker and compare elements one-by-one. However, I noticed the length of the golden corpus list was much larger than the nltk list, so clearly the nltk chunker missed a lot. Rather than check the golden corpus NE list at each index, I decided to iterate by a subset, in order to account for desynchronization. The size of this subset reamined constant, but in hindsight i should have increased the size as a function of the current index, as the list were getting more and more out of synch.

1.3 Problem 3

In general, this problem was pretty straightforward, as most of the code was already given. The difficult problem for me was part C. My immediate thought was that I had to modify the structure of the three trees given in order to eliminate the problem. I noticed that for whatever reason, some verbs were nested in two verb-phrases. I deleted one of the verb-phrases and that seemed to fix the problem. I wish I understood why this worked.