Report : Assignment - 2

Probabilistic Tagger

When I ran the Probabilistic tagger on the training set, I got an accuracy of 0.93787 (after it ran on 152119 words provided in the corpus). Hence its accuracy was around 93.8%.

For unknown word tagging, I employed the following procedure. If a word occurred just once as a particular tag in our corpus, I changed the "word,tag" pair in the dictionary to "unk,tag", i.e. replaced the "word" by "unk" where "unk" stands for unknown, and then populated the emission probabilities in the usual fashion. Whenever I encounter an unseen word, I labeled it with that tag which is the most frequent for the 'unk' words from our training set. It turns out that this tag is "NN".

POS Tagger using Viterbi and HMM (Bigram Model)

When I ran the Viterbi Tagger (after training it on the entire training set), I got an accuracy of 0.95671 (after it ran on 151789 words provided in the corpus). Hence its accuracy on the training set after testing on the same was around 95.67%.

On separating a part of the training set (first 12000 sentences), and training on it, I tested on the remaining 3865 sentences (the dev set), and got an accuracy of 61.25% initially. It turns out this drop in accuracy was due to the fact that I did not empower the program with the ability to tag words it has not seen before (i.e. words not in the training set), the 'unseen' words.

For unseen word tagging, I checked if a word occurred only once in the training set. If it did, I added it to a dictionary with the word as the key and the tag as the value. Then I counted the number of times a particular tag occurred, and recorded the most frequent of those tags. This gives me the tag which has the highest probability for such a word (i.e. one which occurs only once in the training set). When an unseen word occurs in the training set, the idea is to return (or predict) this tag for that word.

For smoothing out '0' probabilities, I implemented Add-k smoothing, and investigated the accuracy achieved for different values of k (0<k<1).

When I used unseen word tagging and Add-k smoothing, after training on a part of the training set (first 12000 sentences), and testing on the remaining 3865 sentences, I got the following accuracies:

For k = 0.8, the accuracy was 90.07%

For k = 0.6, the accuracy was 90.36%

For k = 0.4, the accuracy was 90.59%

For k = 0.2, the accuracy was 90.77%

For k = 0.1, the accuracy was 90.88%

For k = 0.05, the accuracy was 90.89%

Hence I set the value of k to 0.05.

Thus, my prediction is that the accuracy on an unseen test set should be around 91%.