NLP - HW 3

Guy Tevet (305257206), Gavriel Habib (304946445)

Question 2 – Most frequent tag baseline

(b) We evaluated our tagger against the development set, and the accuracy that we got was: 92.54%.

Question 3 – HMM Tagger

(b + c) Our pruning policy was to avoid all the tags that gave zero probability on the emission $e(x_i|y_i)$, and therefore we went over much less tags during the code.
We chose the values of λ_i by performing a grid search over the combinations. The best combination was: $\lambda_1 = 0.5, \lambda_2 = 0.4, \lambda_3 = 0.1$, and we got 95.46% accuracy.

Question 4 – MEMM Tagger

(c) We have made some optimizations on our code to reduce the run-time complexity. In each iteration, we saved only the best 50 probabilities of the tuples (u,v) which are the previous tag and the current tag. It means that in the next iteration we need to calculate only those 50 possibilities. The run-time complexity reduced from 50*46 probabilities in each iteration, instead of 46^2.

(d) We evaluated our tagger against the development set, and the accuracy that we got was: 95.7% with Viterbi, and 95.7% with the greedy inference.

(e) We have sampled errors from our best model, which is the MEMM tagger. Both MEMM with Viterbi, and MEMM with the greedy inference had the same accuracy (95.7%), so we chose to analyze the errors of MEMM with Viterbi.
Here are some common errors:

```
sentence_words:
['initCap', 'Advertisers', 'Try', 'UNK', 'initCap']
real_tags:
['JJ', 'NNS', 'VBP', 'JJ', 'NNS']
predicted_tags:
['NNP', 'NNPS', 'NNP', 'NNP', 'NNP']
```

We can see that our model has made mistakes in all the words in this sentence. For example, it tagged the first word (which is some rare word) as an adjective, even though it is a noun. We can tell that the category 'initCap' does not help us understand the POS of this word in the sentence. This is because it appears as the first word and therefore it is obvious that it begins with a capital letter.

```
sentence_words:
['As', 'Mr.', 'initCap', 'of', 'the', 'allCaps',
'acknowledges', ':', '``', 'Government', 'is', 'not',
'going', 'to', 'solve', 'the', 'problem', '...', '.']
real_tags:
['IN', 'NNP', 'NNP', 'IN', 'DT', 'NNP', 'VBZ', ':', '``',
'NNP', 'VBZ', 'RB', 'VBG', 'TO', 'VB', 'DT', 'NN', ':',
'.']
predicted_tags:
['IN', 'NNP', 'NNP', 'IN', 'DT', 'NNP', 'VBZ', ':', '``',
'NN', 'VBZ', 'RB', 'VBG', 'TO', 'VB', 'DT', 'NN', ':',
'.']
```

We can see that our model has made a mistake only in one word at the above sentence. The model tagged the word 'Government' as 'NN' instead of 'NNP'. We can guess it's not a dramatic mistake…