Problem 1:

Userids: rtalwai-satvarma-spanchal

HMM Formulations

S represents a set of part of speech tags for each word in the sentence and W represents a set of observed words in the sentence

Prior Probabilities: $P(S_t = i)$ where S_0 is the POS tag for the first word and i represents the 12 possible POS tags.

For example $P(S_t = \text{noun}) = \frac{\text{Total number of sentences having POS of first word as noun in the training file}}{\text{Total number of sentences in the training file}}$

Transition Probabilities: $P(S_{t+1} = j \mid S_t = i)$, where i and j represents the POS tags and t is the index of the word in the sentence

For example $P(S_{t+1} = \text{verb} \mid S_t = \text{noun}) = \frac{\text{Total number of noun to verb transitions in the training file}}{\text{Total number of nouns in the training file}}$

Emission Probabilities: $P(W_t = w \mid S_t = i)$

For example $P(W_t = \text{poet} \mid S_t = \text{noun}) = \frac{\text{Number of instances of the word poet tagged as noun in the training file}}{\text{Total number of nouns in the training file}}$

Working of the Program

- We calculate all the above mentioned probabilities inside the train() method.

- Simplified:

  To implement this, we consider the simplified Bayes net and first calculate $P(S_t|W_t)$ by multiplying $P(S_t = i)$ and $P(W_t = w \mid S_t = i)$.
  For example, $P(S_t = \text{verb}|W_t = \text{work}) = P(S_t = \text{verb}) * P(W_t = \text{work} \mid S_t = \text{verb})$.
  Similarly, we calculate for the rest of the POS tags i.e. for $P(S_t = \text{noun}|W_t = \text{work})$, $P(S_t = \text{adv}|W_t = \text{work})$ etc. Next step is to take the maximum amongst all the values of $P(S_t|W_t)$ which would give us the most likely part of speech tag for that word in the sentence. In the end the method returns a list of the predicted POS tag for each word in the sentence.

- Variable Elimination (HMM VE):

  We calculate $P(S_t|W_t)$ by multiplying $P(S_t = i)$, $P(W_t = w \mid S_t = i)$ and $P(S_{t+1} = j \mid S_t = i)$ and taking the maximum value.
  For example (for the 2nd word in sentence), $P(S_2 = \text{verb}|W_2 = \text{work}) = P(S_1 = \text{verb}) * P(W_2 = \text{work} \mid S_1 = \text{verb}) * P(S_2 = \text{verb} \mid S_{1=} \text{noun}, \text{verb}\ldots)$. We add all the values of $P(S_t = \text{verb}|W_t = \text{work})$. This summed value is stored in the dictionary and will be used while calculating the posterior probability for the next word. Similarly, we calculate for the rest of the POS tags i.e. for $P(S_t =$

noun|W_t = work), P(S_t = adv|W_t = work) etc. We take the maximum of all these values to get P(S_t |W_t). This value gives us the most likely POS tag for that word in the sentence.

- Viterbi (HMM MAP):

  We calculate P(S_t|W_t) by multiplying P(S_t = i) , P(W_t = w | S_t = i) and P(S_{t+1} = j | S_t =i). For example, P(S_t = verb|W_t = work) = P(S_0 = verb) * P(W_t = work | S_t = verb) * P(S_{t+1} = verb | S_t= noun ,verb……). We take the maximum of all these values for P(S_t = verb | W_t = work) and store it in a list called *next_list*. This value will be used to calculate the posterior probability for the next word. Similarly, we calculate for the rest of the POS tags i.e. for P(S_t = noun | W_t = work), P(S_t = adv|W_t = work) etc. The maximum of all values for P(S_t | W_t = work) and store it in a dictionary called *backtrack*.

  The above process is repeated till all the words in sentence have been covered. Based on the maximum value of P(S_n|W_n) where n is the index of last word, the POS tag associated with it will be stored. This will be the most likely POS tag for that word. Next we access the dictionary named *backtrack* to check from which POS tag, i, in P(S_{n-1} = i|W_{n-1}) gave us the max(P(S_n|W_n)). This POS tag, i, will be the most likely for the word W_{n-1}. We keep backtracking until we reach the first word. Hence at the end we return a list consisting of the most likely sequence of the POS tags for that sentence.

Results:

We tested our model on bc.test file and go the following results

==> So far scored 2000 sentences with 29442 words.

	Words correct:	Sentences correct:
0. Ground truth:	100.00%	100.00%
1. Simplified:	91.51%	36.35%
2. HMM VE:	93.40%	44.70%
3. HMM MAP:	95.06%	54.45%

Assumptions:

1. For words which don't appear in the training file but exist in the test file, we have assigned them a low value of 0.000001.
2. For Transition Probabilities, Prior Probabilities and Emission Probabilities of the training data, if we do not have a POS tag for a word or any transition of POS, we have assigned it a low value of 0.000001.