NLP Assignment 4, HMM

- *Shreyash Arya (2015097)*

* Data is read from the trainset provided. The words and tags are tab separated (‘\t’) and * sentence ends at full stop (.). Start of the sentence is at the new line (‘\n’).
* Log probabilities are used to calculate all the probabilities to tackle the underflow problem.

1. Extract the required counts from the training data for the various probability estimates that are needed.

- Count of the tags and words are calculated from the training data.
- Bigrams of the tag-tag and word-tags are created.
- Transition probabilities are calculated using the formula: (count(tag1, tag2)+1) / (count(tags1)+|V|) where tag1 is tag from previous state, tag2 is from the current state and |V| is the number of unique POS tags.
- Emission probabilities are calculated using the formula: (count(word, tag)+1) / (count(tag)+|V|) where the word is the observation from the current state, the tag is the hidden state from the current state and |V| is the number of unique POS tags.
- Start probabilities are calculated for each of the POS tags using the formula: (count(tag)+1) / (count(total_tags) + |V|) where |V| is the number of unique POS tags.

2. Deal will unknown words in some systematic way.

The unknown words are dealt with using the <UNK> tag. In training set, the words with the frequency less than 2 are made UNKs and at the time of test if the word is not present in the training set, it is converted into the <UNK> tag and accordingly, probabilities are calculated.

3. Do some form of smoothing (for the bigram transition probabilities).

Laplace smoothing is used for calculating the bigram probabilities of tag-tag and word-tag. +1 is done in the numerator and divided by the number of unique POS tags i.e. 34.
It helps to tackle the problem of 0 probabilities if some tag-tag or word-tag pair is not present in the training set.

4. Implement the Viterbi algorithm.

Viterbi is implemented following the pseudo-code presented in the Jurafsky’s Speech and Language Processing book (3rd edition) on page 153.

Same 3 step procedure is followed:

1. Initializing step: For the first stage, the start probabilities for all the hidden states are calculated as mentioned in part 1.
2. Recursion step: For each stage, the probabilities for all the hidden states are calculated using the transition, emission and previously hidden state probabilities and max the all hidden states is stored. Also, the last state from which the maximum came is stored for backtrack.
3. Termination step: At the final stage, the maximum of all the hidden states and path is stored.

Inferences:

The train accuracy is for the trained model is **86.3818643097%** i.e. the correct number of tags predicted for the words that are stored in the training dataset divided by the total number of tags in the training dataset.

This shows that the model is not perfectly trained as the ideal train accuracy should be 100%. For a given sentence, out of 6 tags, 4-5 tags are predicted correctly.

Output:

Output is presented in the same format as asked in the assignment. The output is printed in the format (word tag) for each sentence separated by a newline. Also, the output is stored in a file with the same format as the train data.

References:

https://medium.freecodecamp.org/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24
https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf
https://github.com/WuLC/ViterbiAlgorithm
https://towardsdatascience.com/part-of-speech-tagging-with-hidden-markov-chain-models-e9fccc835c0e