To run script, cd into the root directory of the project & type the following command:

`./script.sh`

Alternatively (or if using a non-unix system), in the root directory of the project, run the following:

```
python3 hmm_part_1.py
python3 hmm_part_2.py
python3 hmm_part_3.py
python3 hmm_part_4.py
python3 hmm_part_4b.py

# run on test data
python3 hmm_part_4.py -t train -i test.in

python3 EvalScript/evalResult.py RU/dev.p1.out RU/dev.out > RU/1.out
python3 EvalScript/evalResult.py ES/dev.p1.out ES/dev.out > ES/1.out

# repeat for all the output files
```

Part 2 & 3

In our transition step, instead of only keeping track of the best sequence found, we keep track of the all possible sequences occuring after the previous step, and we truncate each step to the top `i` values (`i = 1` in part 2 and `i = 5` in part 3).

We use log likelihood to avoid floating point precision issues.

To handle the cases where no path can be found with a score of more than 0 (which happens for example when a word we have only observed as `I-positive` occurs after a word only observed as `O`, since $t(\texttt{I-positive}|\texttt{O}) = 0$), we set $\log(0)$ to be a large negative number (-1000000) so that in such cases, we consider the "least impossible value" (likelihood scores where we have to multiply by the least number of zeroes, followed by those with the same number of zeroes where we would have a higher score if we did not multiply by those zeroes).

In practice our log score (when there is a feasible solution) is in the range $[-10^3, 0]$, so this should not affect the cases where a feasible solution can be found.

Part 4

Now, based on the training and development set, think of a better design for developing an im- proved sentiment analysis system for tweets using any model you like. Please explain clearly the model/method that you used for

designing the new system. We will check your code and may call you for an interview if we have questions about your code. Please run your system on the development set RU/dev.in and ES/dev.in. Write your outputs to RU/dev.p4.out and ES/dev.p4.out. Report the precision, recall and F scores of your new systems for these two languages. (10 points)

A disadvantage of the HMM is that it only remembers the previous state. In real life, when we interpret the sentiment of a word we use much more of the surrounding context to make our judgements.

Hence we extend the HMM by hypothesizing that each new state does not solely depend on the previous state, but also the state before it. In this model, instead of having transmission parameters $t(y_n|y_{n-1})$ we have a new transmission parameter $t(y_n|y_{n-1}, y_{n-2})$.

To find the tag sequence that gives us the maximum score, we extend Viterbi's algorithm to keep track of the maximum score so far given the current tag and the previous tag. Letting this be $b_{i,y_i,y_{i-1}}$, we have the recurrence

$$b_{i,y_i,y_{i-1}} = \max_{y_{i-2}} b_{i-1,y_{i-1},y_{i-2}} e(x_i|y_i) t(y_i|y_{i-1}, y_{i-2})$$

In addition, we no longer try to handle unknown words in the same way as before. This is because there are a lot more values tagged as `O` than there are for instance `I-negative`, and because we estimate the emission parameters for unknown words as $e(\texttt{\#UNK\#}|Y) = \frac{1}{\text{Count}(Y)}$ the parameter $e(\texttt{\#UNK\#}|\texttt{O})$ is a lot less than the parameter $e(\texttt{\#UNK\#}|\texttt{I-negative})$. In practice there is not much reason for the assignment of these scores, and by handling each of them as "impossible" cases (as explained in part 2/3) we are effectively assigning them the same emission parameters for each possible tag. We have found that our F values increase quite dramatically due to this change, as realistically we should expect that most unknown values should be tagged as `O` (rather than as the other tags, which are much rarer).

Our Sentiment F scores improve on part 2 by about 0.2 in their evaluation of both the ES and RU development sets.

```
# ES

#Entity in gold data: 255
#Entity in prediction: 215

#Correct Entity : 139
Entity  precision: 0.6465
Entity  recall: 0.5451
Entity  F: 0.5915

#Correct Sentiment : 112
```

```
Sentiment  precision: 0.5209
Sentiment  recall: 0.4392
Sentiment  F: 0.4766

# RU

#Entity in gold data: 461
#Entity in prediction: 353

#Correct Entity : 239
Entity  precision: 0.6771
Entity  recall: 0.5184
Entity  F: 0.5872

#Correct Sentiment : 163
Sentiment  precision: 0.4618
Sentiment  recall: 0.3536
Sentiment  F: 0.4005
```

We have attempted to extend this model such that our transitions depend on the previous 3 states instead (as seen in the files labeled part 4b), but we have found that this does not meaningfully affect our F scores.