COMP9318 Project Report

(1) Implementation for Q1

First, create Matrix A.

A transition probability matrix A, each $a_{i,j}$ representing the probability of moving from state i to state j satisfy $\sum_j^N a_{i,j} = 1 \; \forall \; i$

In order to perform the **add-one smoothing**, all the **entries firstly are initiated to 1**. After the input file split, we add the transition frequency to the related entries. How even there are 2 special case:

The A [-1], the ***END*** state, is set to 0 means which means from the end state to any state the probability are 0.

A[:,-2] =0, means from any states to the ***BEGIN*** states, the probability is 0.

Secondly, we create Matrix B and hash map about symbol to their ID.

A Matrix of observation likelihoods, also called emission probabilities, each expressing the probability of an observation o_t being generated from a state i to store the probability that the observations appeared in states.

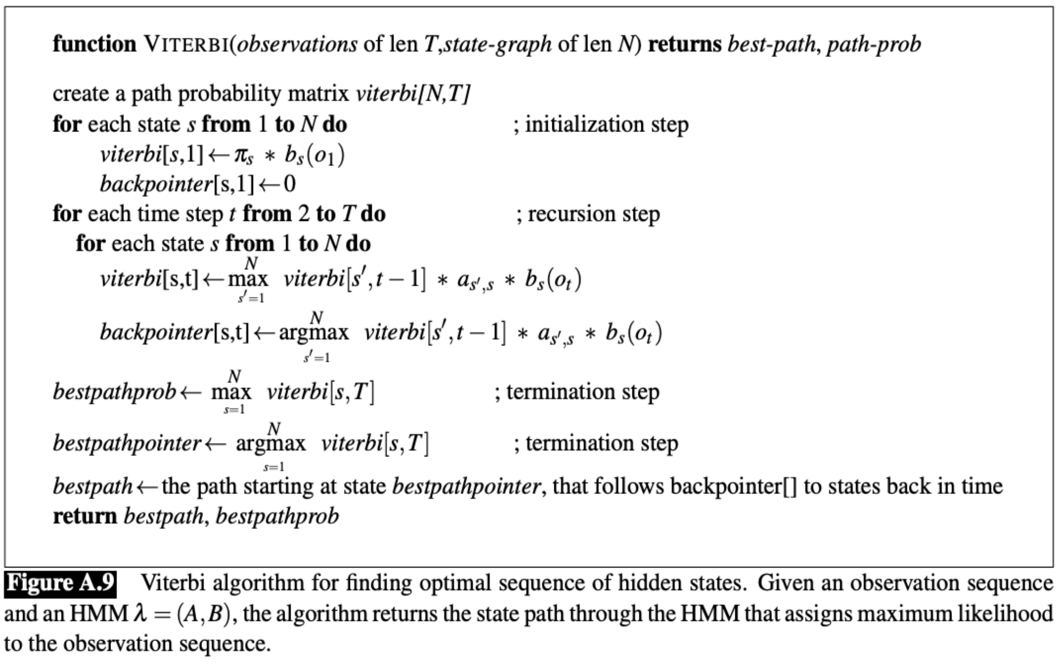
Create it with input file and perform the **add-one smoothing**, in same analogy with Matrix A

Here is another special case **UKN**, we handle it with using **defaultdict** set default value to the *given symbol length,* and that makes equivalent to store all **UKN**'s ID to a same id which is the *given symbol length.* When we build the Matrix B we adding another column in the last for it (the original index of the last column id is *given symbol length – 1,* and we adding 1 last column just let the index equal to our default set ID)

Thirdly, handle query.

We recursively token the sequences based on the symbols given in the specs. And got the observation sequence with length T

And perform the Viterbi algorithm
The pseudo code we use to implement the Viterbi algorithm is from:
https://web.stanford.edu/~jurafsky/slp3/A.pdf



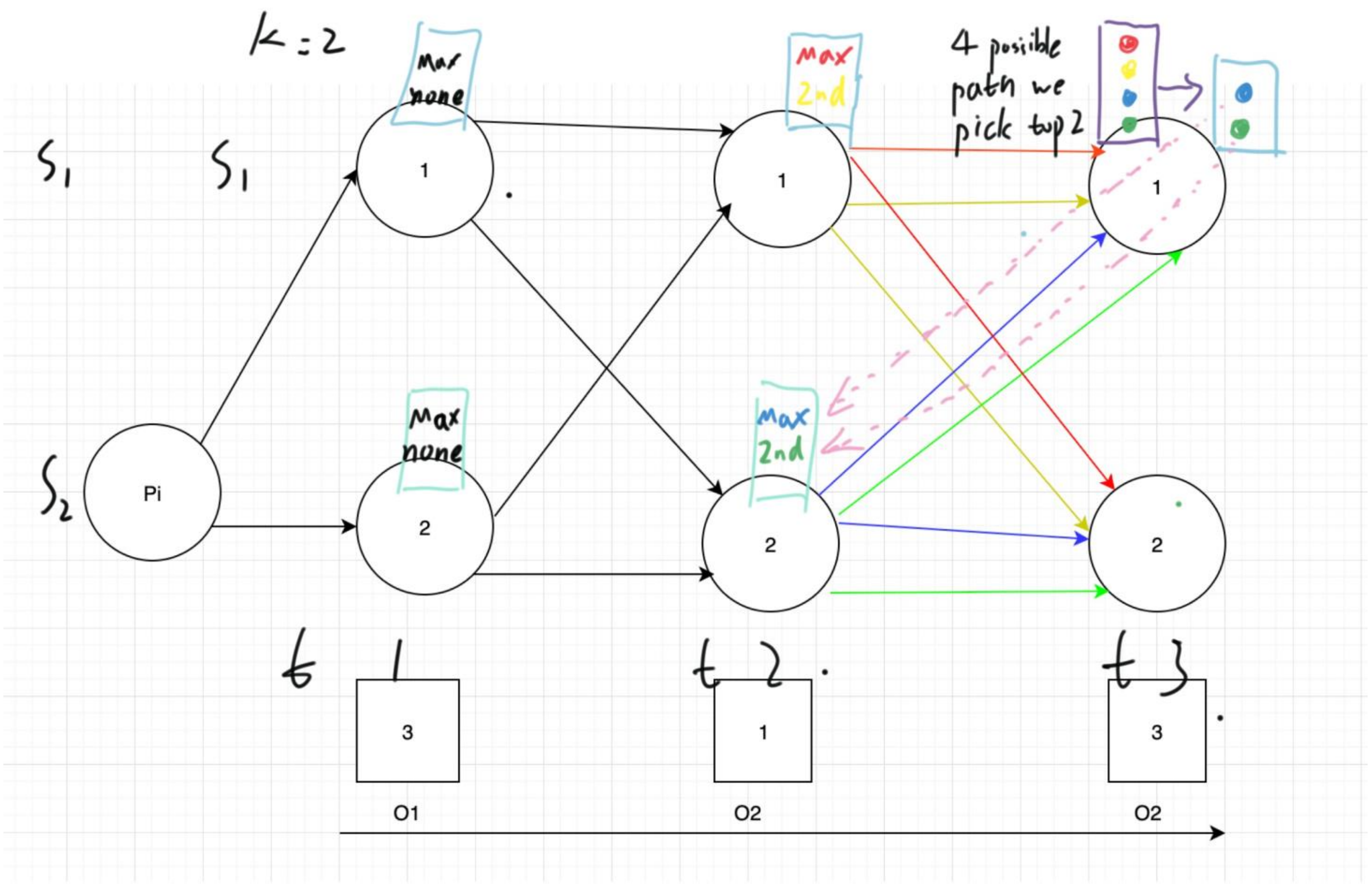
```
function VITERBI(observations of len T,state-graph of len N) returns best-path, path-prob

create a path probability matrix viterbi[N,T]
for each state s from 1 to N do                          ; initialization step
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step t from 2 to T do                      ; recursion step
  for each state s from 1 to N do
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$                ; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$          ; termination step
bestpath ← the path starting at state bestpathpointer, that follows backpointer[] to states back in time
return bestpath, bestpathprob
```

Figure A.9 Viterbi algorithm for finding optimal sequence of hidden states. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

(2) Implementation for Q2

In order to perform extend Viterbi algorithm to Top-K, what been stored in the *Viterbi and Backpointer* should represent at least Top-K path to this point. The key idea is:

1. Extend the *Viterbi* to store the Top-K probability rather the MAX probability.
2. Extend the *Backpointer* to store the related back state for each Top-K path, and in order to know where the path coming, we not only need to know which state the path comes from , but also need which rank the path is in that back state. So we build the *Backrank* to store the back path's rank in the back state.



For example k=2 T=3 N=2, in T1 because we only have 1 path from ***BEGIN*** so even we have k=2 the valuable probability in our *Viterbi* for this moment is only one , and when path goes to T2 we just have 2 choose , then we go to T3, here we can see that we have 4 different path for each state, and we store the top-2 in our *Viterbi matrix* in this moment, but we need to remember where this path coming, like in this example the top-2 paths all coming from the same back states (state 2) so we need to also remember the back rank of our top-2 probability.

The other part of the algorithm is same as the original one.

(3) Implementation for Q3

Reported in the bonus.pdf