COMP9318 Project Report

Yuhan He (z5224997)
Zihao Xu (z5184152)

Implementation details of Q1

After reading *State_File* we got the data information as following:

1. The number of states, N
2. The matrix, N_{ij}
3. An array, N_i

Every element n_{ij} in N_{ij} is the number of frequency of seeing state sequence i followed by j.
Every element n_i in N_i is the frequency of seeing state i.
Then calculate the matrix A which is the transition probability matrix.

After reading *Symbol_File* we got the data information as following:

1. The number of symbols, M
2. The matrix, N_{ij}
3. An array, N_i

Every element n_{ij} in N_{ij} is the number of frequency of seeing state i transit to symbol j.
Every element n_i in N_i is the total number of times state i has been seen.
Emission matrix B is calculated according to the information above.

As N, A, M, B has been calculated, then we implement the Viterbi Algorithm for each query.

In the emission matrix, we add one column at the end intends to store the probability of transfer from state i to unknown tokens (tokens not in the *State_file*).

Firstly, we translate the tokens in a query to symbol IDs (if the token is unknown, we treat its id as M).
Viterbi algorithm is essentially a dynamic programming problem.
Secondly, initialize the first column of the dynamic programming matrix. Each value is the logarithm probability of state *BEGIN* transferring to state i where i is the line denotes the i-th state.
Then, calculating values in the dynamic programming matrix by column and row. Each value is the maximum value of comparing all possible previous states transfer to the current i-th state. Each value to be compared is calculated by simply adding the log probability of each state transfer to i in A, and adding the log probability of seeing symbol j from state i in emission probability matrix B.
At the same time, we store the states sequence of the maximum value we got so far.
When the above computing finished, we add the log probability of transferring state i to *END* state to every value we got and select the maximum one, adding by *END* state, we got the most possible state sequence of seeing the query tokens.

Extend Viterbi algorithm in Q1 to top-k Viterbi algorithm

Now we need to extend the dynamic programming matrix to store top k-sequences. Currently, at each step(computing the i-th row and j-th column value in the dynamic programming matrix), we only store one parent. We extend that to a list in order to store the top k predecessors. So each step corresponds not only to a most likely value and the node which it transitioned from but a list of the top k (at most k, maybe less than) nodes it could have transitioned from and their values in sorted order. At each step, we also sorted the result according to the ties and just store the top-k values, if there are less than k selections, we keep them all.

The last step is to add the log probability of transferring state i to END state to every value we got and select the top-k maximum sequences as the final result.

Approach for advanced decoding

In order to reduce the *Total Number of Incorrect Labels* in Q1, we choose Absolute Discounting to smooth the emission probabilities.

Now, we declare the meaning of symbols we used in the calculation:
M : The total number of symbols appear in the Symbol_File

F : A list with length of the number of states, $F[i]$ is the number of different symbols transmitted from the i-th state.

N : A matrix in the shape of *(total states number) * (total symbol number + 1)*. We added one more symbol: *UNK* to denote all tokens that have not to appear in the *Symbol_File*. The symbol ID of *UNK* is the total symbol number M.

$N[i][j]$: The frequency for each symbol i.

S : A list with length of the number of states, $S[i]$ is the total frequency number of all possible symbols for the i-th state.

We choose an arbitrary value $a[i] = \frac{1}{(F[i]+S[i])}$

The formula we used to calculate the probability that from state i to symbol j as follows:

$$if\ N[i][j] = 0,\ B[i,j] = \frac{F[i]*a[i]}{M - F[i] + 1}$$
$$else,\ B[i,j] = \frac{N[i][j]}{S[i]} - a[i]$$

Absolute discounting is a serendipitously discovered estimator, it corrects empirical frequencies by subtracting a constant from observed categories, which it then redistributes among the unobserved. It outperforms classical estimators empirically and has been used extensively in natural language modelling.