COMP9318 Project Report

Jialun Li

April 30, 2019

1 Implementation of Q1

1.1 Processing files

• By reading StateFile, get the number of total States and the main content of StateFile (appearance of transition).
• By reading SymbolFile, get the number of total Symbols, create a dict for the Symbol associated with ID and get the main content of SymbolFile (appearance of emission).

1.2 Computing the probability

• For transition prob, create a matrix in size (Statenum, Statenum).By scanning the StateFile line by line, count the respective ni and nij with smoothing. Then do log(nij/ ni) for each slot in the matrix.
• Do the same thing for the emission prob matrxi which is in size (Statenum, Symbolnum). The difference is if the symbol were UNK, instead of getting prob from matrix, just simply consider it as log(1/ni(with smoothing)).

1.3 Processing query

• By reading Queryfile, each Query for one line, write a helper function called parseingquery(query), which is able to break the string into substrings with the seperator of "$*, ()/-\&*$" and 'spaces', now thew query is ready to be labelled.

1.4 Label

• By using viterbi algorithm, create a viterbi list in size (Statenum, len(query)).The row means states and the column means symbols. Compute the first column of the list which is the initial prob of each state emission the first symobl in the query. $P(state|begin) * P(symbol|state)$
xThen compute the mean part of the list using the dynamic programming. For each prob only store the max prob caculated from the previous column by using the formula $P(New) = P(Previous) * P(transition) * P(emission)$.
• At the same time of doing viterbi list, we create a same size matrix called

postionlist, where store the path of each max prob.
• Finally and the P(end—state) (log number) to the last column.

1.5 Output the result

• Check out the last column og viterbi matrix, finding the max value which is going to be the max probability of one query and the index of that value. Similarly, then use the index to find the path in the last column of positon matrix and the ID of *Begin* and *End*. Append the max prob into that path.
• Return the result properly.

2 Modification for Q2

2.1 Overall

• In Q1 implementation, for each slot in viterbi matrix, only the max prob and path was added. For Q2, the algorithm is designed for top-K problem and the key is adding top-K numbers of prob and paths into the matrix.

2.2 Cases discussion

• Say for N (numbers of states without specials) and L (length of query), the maximum value for K is N^L.
• Case 1 if $N^{L-1} \leq K \leq N^L$ In this case its essential to compute all the N^L result and sort them by prob to choose top K.
• Case 2 if $K = 1$ Same to Q1.
• Case 3 if $1 < K < N^{L-1}$ For each column, add as much as possible prob and path into the matrix such as N^0 for first column, N^1 for second column and so on until to the certain column where $N^{j-1} < K \leq N^{j-1}$ where j is the is of column. From this certain column, only add K numbers of prob and path into the matrix to the end.
Finally sort all the $N * K$ results by prob associate with path to choose top K.
• Return the result properly.

3 Q3