COMP9318

Project1

Group Name: Twosome
Group Member: GUANQUN ZHOU, z5174741 KAIWEN LUO, z5100899

1. Implementation details of Q1.

- Parse query with *, ( ) / - & * by using regex(re.split() and re.sub()).
- Find state lists and observation lists
- Use State_file to generate hidden state, initial state matrix and transition matrix.
- Use Symbol_file to generate emission matrix.
- Apply add-1 smoothing to smooth transition matrix and emission matrix.
- In function viterbi(obser_li, state_li, init_pro, trans_pro, emission_pro), we generate two matrixes. one stores max probability, another one stores the path to get this max probability.
- Just like the way mentioned in lecture, in the loop, for every current observed symbol, we find the max possibility after computation and update the corresponding path.
- Finally, according to what we have gotten as the max possibility for last observed symbol, we can use these two matrixes to find exact hidden state and its log probability.

2. Details on how you extended the Viterbi algorithm (Q1) to return the top-k state sequences (for Q2).

- In Q1, we only find max probability and the path to get it. However, in Q2, our task is to find best k probability and its step.
- The difference is that we should store first topK possibility for every corresponding state for current observed symbol in a list, then get it sorted by the probability. Specifically, we get the first topK possibility from all the states, then get it sorted by the probability including its state and topK(as a tuple)
- In the end, we can finally backtrack to find the topK corresponding hidden states and their log probability.

3. Your approach for the advanced decoding (Q3).

- Our team observed a fact that there are a lot of symbols of which emission frequency is significantly huge. However, the emission frequency of some others is only 1 or 2.
- As above fact shows, the distribution of symbols has limitation, which is called data sparseness. Even if the corpus was expanded to a large scale, these extremely small probabilities would also exist, and it cannot provide reliable probability evaluation.
- In terms of this situation, we apply several smoothing methods to optimize data sparseness().
- For add-1 smoothing, the result of incorrect labels is 134.
- For add-k(0<k<1) smoothing, the best result of incorrect labels is 131.
- For Kneser-Ney smoothing, the value of d (discounting) usually is empiric value 0.75 and the result of incorrect labels is the best, which is 116.

- The formula of Kneser-Ney smoothing is like this:

$$P_{KN}(w_i|w_{i-1}) = \frac{\max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{continuation}(w_i)$$

  where

$$\lambda(w_{i-1}) = \frac{d}{\mathrm{C}(w_{i-1})} \cdot |\{w: C(w_{i-1}, w) > 0\}|$$

$$P_{continuation} = \frac{N(.w_i)}{N(..)}$$

  and

$$N(.w_i) = |\{w_{i-1}|c(w_{i-1}, w_i) > 0\}|$$

$$N(..) = |\{(w_{i-1}, w_i)|c(w_{i-1}, w_i) > 0\}|$$

  For UNK, we set $N(.w_i) = 1$, which is the best guess.
- In conclusion, the best result among these smoothing algorithms is Kneser-Ney smoothing, which is the combination of several smoothing methods. Therefore, it has good effect on optimizing sparse data and decreasing the rate of incorrect labels.