CS114 (Spring 2020) Programing Assignment 4

Part-of-speech Tagging with Hidden Markov Models

Heyuan (Henry) Gao

Program Instruction

Training function

In the train() function, I used *word_dict* and *pos_dict* to store words and tags but the value of dictionary is index. This is convenient to create related matrix. And I also created a *reversed_pos_dict* applied to viterbi function. To save memory space, I generated sparsed matrix for emission B matrix, so there was an extra procedure in viterbi function adding k-smooth method on B matrix.

Viterbi function

This function directly deployed viterbi algorithm for a single sentence noted that the input sentence should be a list of word index. The most complex part is recursion step where the algorithm formula is $viterbi[s,t] = \max_{s'=1}^{N} viterbi[s', t-1] * a_{s',s} * b_s(O_t)$. However, with broadcasting feature of numpy arrays, the application of this formula in my code is to mutiply viterbi[:, t-1] with shape (s, 1), transition A matrix with shape (s, s) and B(O_s) with shape (1, s). The result would be a (s, s) matrix where max value of each row is what we need for viterbi[s, t].

K selection and model evaluation

In this case, I continued using grid search to find the best k of smoothing method. Bellowed is grid searching result for k from 0.1 to 1 with step size 0.1 where we can estimate the best k should be less than 0.1.

	k	accuracy
0	0.1	0.911255
1	0.2	0.904911
2	0.3	0.899261
3	0.4	0.893974
4	0.5	0.888927
5	0.6	0.884573
6	0.7	0.880848
7	0.8	0.87737
8	0.9	0.874198
9	1	0.870431

Therefore, I applied a more precise k list from 0.01 to 0.1:

	k	accuracy
0	0.01	0.906489
1	0.02	0.909653
2	0.03	0.911387
3	0.04	0.912007
4	0.05	0.911677
5	0.06	0.912098
6	0.07	0.911784
7	0.08	0.912123
8	0.09	0.91161
9	0.1	0.911255

The accuracy did not change a lot in different ks, so I chose the best k as 0.08. And the final model accuracy is 0.912123.