Younjee Kang
1005978621

CSC384 - A4: HMM Part-of-Speech Tagging Heuristics

— the interesting heuristics or techniques used to enhance the performance of our HMM! —

From Viterbi:

To get the 'most likely' tagging for each word, we have found that using the Viterbi algorithm will let us find the corresponding tags to the observations, which are given words. However, just running the ordinary Viterbi algorithm did not seem to work as well as we have expected. Thus, we have made a few modifications to it.

There are two main objectives of all of the listed modifications below on the ordinary Viterbi algorithm;

(1) To achieve better accuracy, and

(2) To simplify the computations as much as possible so that it is efficient in time and space.

Using Logarithmic Functions:

To get a probability of each timestamp of each state, there are two difficulties we have found to be very challenging.

As we learned, the likelihood of a path is the product of the transition probabilities along path X the probabilities of the observations at each state. However, given that there are 76 hidden states(possible tags) and a possibly very large number of observations we have to infer, it is inevitable to do this continuous multiplication a lot of times. These may cause:

1. Very long running time. We questioned each time if this computation is inevitable at all. As soon as we can find this resulting probability is never used as the maximum probability at the end, we stop the computation, moving on to the next computation.

   But for those inevitable multiplications, we had to find another way. By using a log function, we could use a log rule, that is using an addition instead of the multiplications, which showed an improvement in the runtime, especially with very big test cases.

2. Very small numbers in decimals, so the program won't be very good at comparing or even storing those values. The normalization would definitely help. Though, normalizing is basically a division that may take another long time, when considering each of the calculations.

   Therefore, we have decided to choose a python built-in log function so that we can avoid very very small decimal points, and as we hoped, avoid the need for normalization, as well.

Younjee Kang
1005978621

Space:

When computing the probability, by the HMM model's property, in terms of the timesteps, we only care about the last probability of a hidden state. Thus, we do not need to carry all the former probabilities excluding the very last one. So when we can get to compute the next timestep's probability, we can deallocate unnecessary space to make a more efficient operation.

Obvious Tags

As a preprocessing space, what if we have found a very trivial, or very obvious word that as a human, we can tag the word right away? We know it might be not ideal to do let the algorithm know all the background knowledge we humans have, but for tags like punctuations and for those we use already in a program to split the sentence, we can let the AI know about this information so that it can avoid computing all the hidden states' probability for that observation. For example, if 'the' is encountered, just tag 'AJ0' right away!

Splitting the Sentence

As we know the POS tagging matters within a sentence, we have splitted and run the Viterbi for each sentence to reduce time.

Unknown Probabilities

For unknown emission or transition probabilities, we have used an epsilon value, which we define to be a very small number. This is just to handle a case where we have encountered a totally new observation or a totally new transition.

Besides this very naive implementation, for an emission probability specifically, we have used smoothings and checked if any variations of the word are observed before. Then, we assign an appropriate value for each case.