Hidden Markove Model

Kasra Eskandari
955361005

June 22, 2020

1 Data Structure

the data structure must contain tokens and part of speech (PoS), in order each token must be in a single line, in which tokens and PoS must be separated with a space character.

2 Preprocess

this method is implemented in `HMM` class available in `pltk/HMM/hmm.py` file. this method will convert the data file into a list of tuples, un which each tuple contains the token and its PoS.

NOTE: each token will be normalized.

NOTE: The lines having nonstandard structure will be ignored

3 Train

to train the model we have to populate two matrix:

- $stateTransision_{N \times N}$: $stateTransision[i, j] = P(State_j | State_i)$ WHERE N is the number of states.
- $tokenProbability_{N \times M}$: $tokenProbability[i, j] = P(Token_j | State_i)$ WHERE M is the number of tokens.

  NOTE: To avoid underflow we will use the logarithm value of probabilities

The pseudo code of calculation is described below:

- count the tokens for each state, also number of altered states and record them in appropriate matrix
- find the number of all states(name is N)
- calculate logarithm value of both matrices
- consider the fact that $log(\frac{a}{b}) = log(a) - log(b)$ we can avoid deviding small numbers

The algorithm's complication is $O(N \times M)$

4 Finding The Most Probable State Sequence

consider state sequence as $S_0S_1S_2S_3$ for tokens of $T = t_0t_1t_2t_3$, we have:

$$P(T) = P(S_0|\$)\prod_{1}^{3} P(S_i|S_{i-1}) * P(t_i|S_i) = e^{log(P(T))}$$

$$log(P(T)) = P(S_0|\$)\sum_{1}^{3} P(S_i|S_{i-1}) + P(t_i|S_i)$$

as we know $f(x) = e^x$ is a ascending function, which concludes from, if and only if we maximize $log(P(T))$ the $P(T)$ will maximize too.

Too maximize the $log(P(T))$ we can localy focus on each

$$P(S_i|S_{i-1}) + P(t_i|S_i)$$

term, which has N conditions for each token(N is the number of states).

In order too find this value(and also corresponding state sequence), we will form a matrix in which its columns name are tokens and rows name indicates the states.In the next step, for each cell(S_i, t_j) we will sum two arrays index by index,one taken from $stateTransision$ matrix's i's row, and the other taken from $tokenProbability$ matrix's j's column. At last, we will find and replace the maximum value for whole array with the previous column.

The algorithm's complication is $O(N^2 \times M)$