

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | Hidden Markov Model | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | DevbratAnuragi |  | | | |
|  | | | | 4/12/2020—Computational Linguistic—DES 308 |  | | | |
|  | | |  | | |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | | |  | | |  | | | |  | | |
|  | | Hidden Markov Model | | | | | | | | | |  | | |
|  | |  | | |  | | |  | | | |  | | |
|  | |  | |  | | | | |  | | |  | | |
|  | |  |  | Hidden Markov Model is a probabilistic sequence model, that computes probabilities of sequences based on a prior and selects the best possible sequence that has the maximum probability. Here the sentence for which ithe POS tagging is done is considered as a set of sequence of words and sequence of tags. HMM is an extension of Markov chain  Markov chain models the problem by assuming that the probability of the current state is dependent only on the previous state. For example, consider the problem of weather forecast with three possible states for each day, namely; sunny and rainy. The Markov chain model states that the probability of weather being sunny today depends on whether yesterday was sunny or rainy. It does not take into account of what was the weather day before yesterday. | | | | |  |  | |  | | |
|  | | | | | | |
|  | | Markov Chain | | | | | | | | |  | | |
|  | |  | | |  | | |  | | |  | | |
|  | Mathematical defination  * N = Number of states. In the above example, N=2(sunny, rainy). * p(a/b) = probability of state a occurring given that the previous state is b. This is called the transition probability.   In HMM the states are not observable, as is the case with POS tagging problem. The states are the tags which are hidden and only the words are observable. Therefore HMM the following components along with components of Markov chain model mentioned above:   * p(o/b) = probability of state b giving out o as the output. This is called as the emission probability.  Modeling POS tagging as HMM   Source: [Mayank Singh NLP 2019](https://sites.google.com/a/iitgn.ac.in/nlp-autmn-2019/)  The problem of POS tagging is modeled by considering the tags as states and the words as observations. For example, in the image above, for the observation back there are 4 possible states. Consequently the transition and emission probabilities are also modified as follows.    Let {w\_1 w\_2 w\_3…w\_n} represent a sentence and {t\_1 t\_2 t\_3…t\_n} represent the sequence tags, such that w\_i and t\_i belong to the set W and T for all 1≤i≤n respectively then,  p(w\_1 w\_2 w\_3…w\_n, t\_1 t\_2 t\_3…t\_n) is the probability that the w\_i is assigned the tag t\_i for all 1≤i≤n. This can be calculated with the help HMM.   Viterbi Algorithm The main key part of this algorithm is to use dynamic programming. It uses the concept of “memoisation”.  We store the probability and the information of the path as follows:     Dataset The dataset I have used is the nltk’s Brown corpus, More precisely I have used the tagged\_sents() of it. Summary of the code: As mentioned above I have used Tagged Sentence of Brown corpus. Point to note that in order to identifies the start and end of the data set I have used ('##','##') as starting and ('&&','&&'). Then I splitted the data into training set (90%) and test set(10%). Then is calculated the frequency of each word tagged with a particular TAG. Then I created the emission probability matrix by simply dividing the frequency of each word with the total count of words of that state.  Now once we have emission probabitlties I have found the sequence of tag as bigrams. This will be used In finding the transition probabilities. Once we have transition probabilities. I have found out what are the possible tags which are given to a word in the brown corpus.  Now everything is set up we need to implement the Viterbi algorithm. The complete detailed/commented code is given in the jupyter notebook.  After training and testing My model’s Accuracy is around: **89.0% – 92.0%**  And loss is around **0.107** Experiment : I plotted the graph for wrongly tagged word and their frequency. the graph is very big. I have attached the image separately for better understanding [click here](https://drive.google.com/file/d/1fVEJJjg9IaVkwYQfvEQYihGpPPTPDCT_/view?usp=sharing).  So if you have a look at the x-axis you with find each labels of the x-axis in thi format: “NN JJ“ these means that each the first tag is the Ground Truth and the second tag( separated by a <space>) is the tag predicted b my HMM model.   Observation: The word with ground truth tag of **VBD** is mostly predicted as **VBN** tag.  Following is the table for better understanding. Possible Reason for this observation:  * The VBN and VBD tags are interchangeable as seen from the above images , but if you look at the Misclassified-words column of the above figure you will notice both row 0 and row 2 have word that ends with “ed”. In present perfect tense and past tense we always have verb in past tense but the overall sentence in may be in the present or past tense. May be because of that those word are tagged wrongly. * The word “to” is mostly tagged as “TO” instead of “IN” this may be because in the training set tag word “to” has been mostly tagged as “TO”. The inference from here is that in the trained model “TO” has higher emission probability for “to” that “IN” tag. | | | | | | | | | | | |  |
|  |  | | | | |  |  | | | | | |  |
|  | |  | | |  | | |  | | |  | | |
|  |  | | | | | | | | | | | |  |