Hidden Markov Model

Devbrat Anuragi

4/12/2020

—

Computational Linguistic

—

DES 308

Hidden Markov Model

Hidden Markov Model is a probabi listic sequence model, that comput es probabilities of sequences based on a prior and selects the best poss ible sequence that has the maximu m probability. Here the sentence f or which ithe POS tagging is done i s 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 onl y on the previous state. For exampl e, consider the problem of weather forecast with three possible states f or each day, namely; sunny and rai ny. The Markov chain model states that the probability of weather bei ng sunny today depends on whethe r yesterday was sunny or rainy. It d oes not take into account of what w as the weather day before yesterda y.

Markov Chain

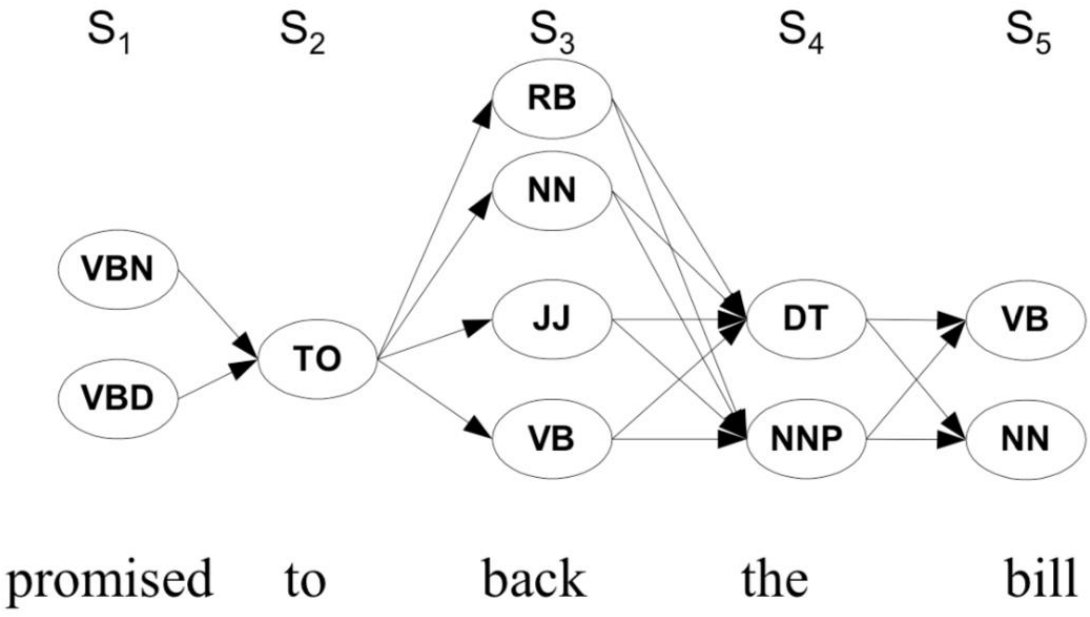
Mathematical defination

- 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

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.

- $p(t_i|t_j)$ is the transition probability from tag t_i to tag t_j
- $p(w_i|t_i)$ is the emission probability of w_i from tag t_i.
- T is the set of all possible tags and W is the set of all possible words.

Let *{w_1 w_2 w_3...w_n}* represent a sentence and *{t_1 t_2 t_3...t_n}* r epresent 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* i s assigned the tag *t_i* for all *1≤i≤n*. This can be calculated with the help H MM.

- $p(t_i|t_j)$ is the transition probability from tag t_i to tag t_j
- $p(w_i|t_i)$ is the emission probability of w_i from tag t_i.
- T and W are the set of all possible tags and words respectively.

Viterbi Algorithm

The main key part of this algorithm is to use dynamic programming. It uses the co ncept of “memoisation”.

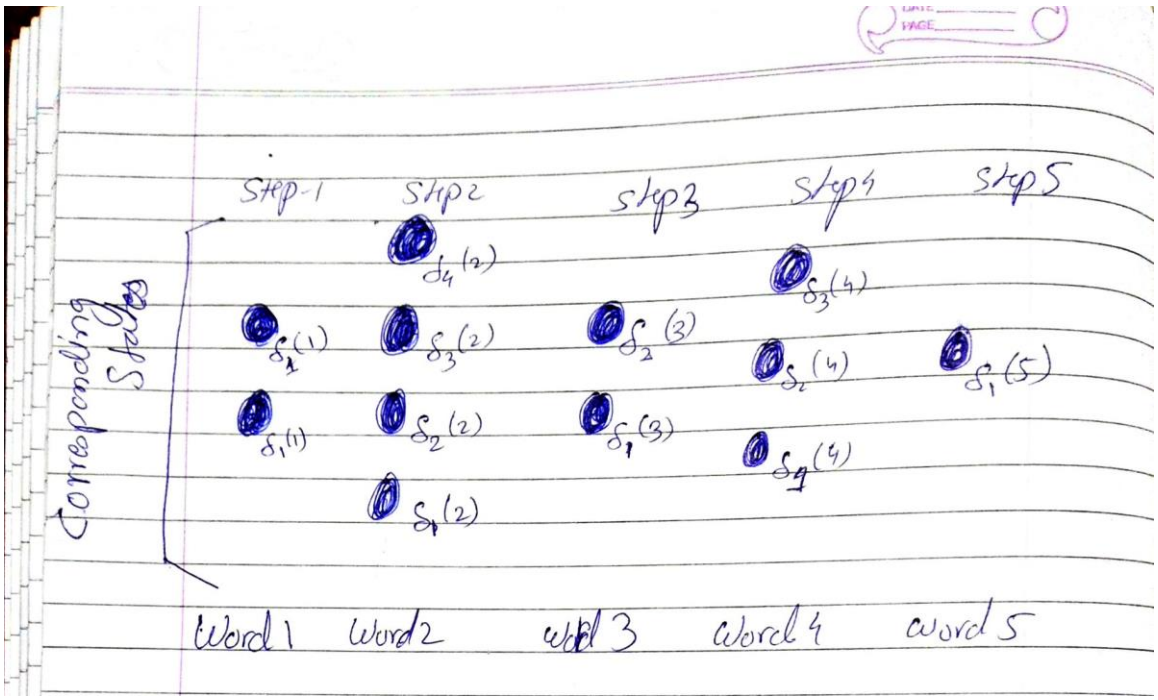
We store the probability and the information of the path as follows:

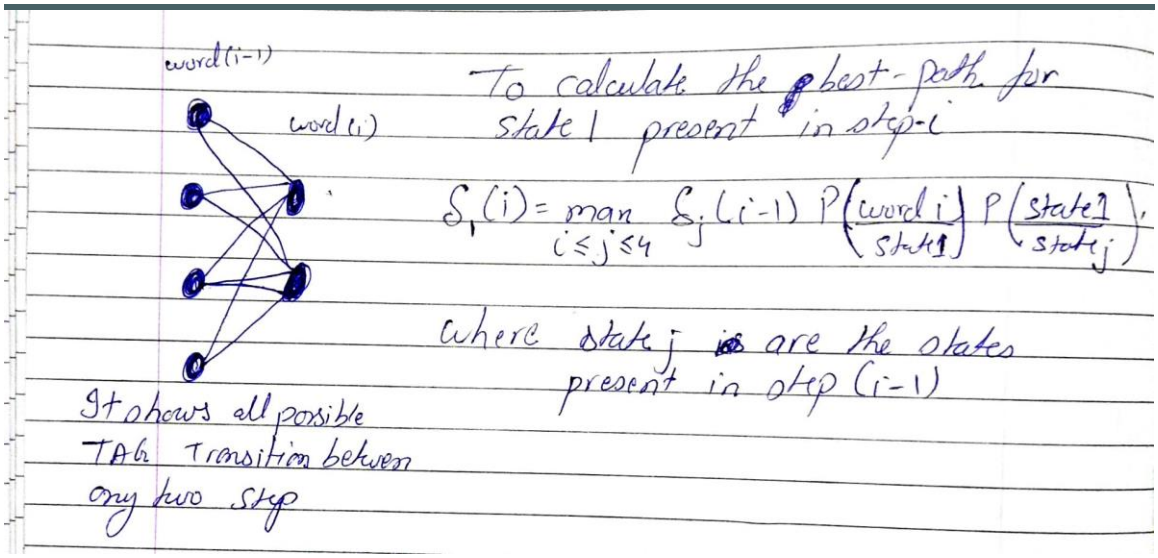
$$\delta_j(s+1) = \max_{1 \le i \le n} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$$

$$\psi_j(s+1) = \arg\max_{1 \le i \le n} \delta_i(s) p(t_j|t_i) p(w_{s+1}|t_j)$$

where,

- $\delta_j(s)$ stores the information on the lowest cost(maximum probability) in reaching j^{th} state at step s.
- $\psi_j(s)$ stores the information on the best path to j^{th} state at step s.





Dataset

The dataset I have used is the nltk's Brown corpus, More precisely I ha ve used the tagged_sents() of it.

Summary of the code:

As mentioned above I have used Tagged Sentence of Brown corpus. Point to note t hat 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 s et(10%). Then is calculated the frequency of each word tagged with a particular T AG. Then I created the emission probability matrix by simply dividing the frequen cy of each word with the total count of words of that state.

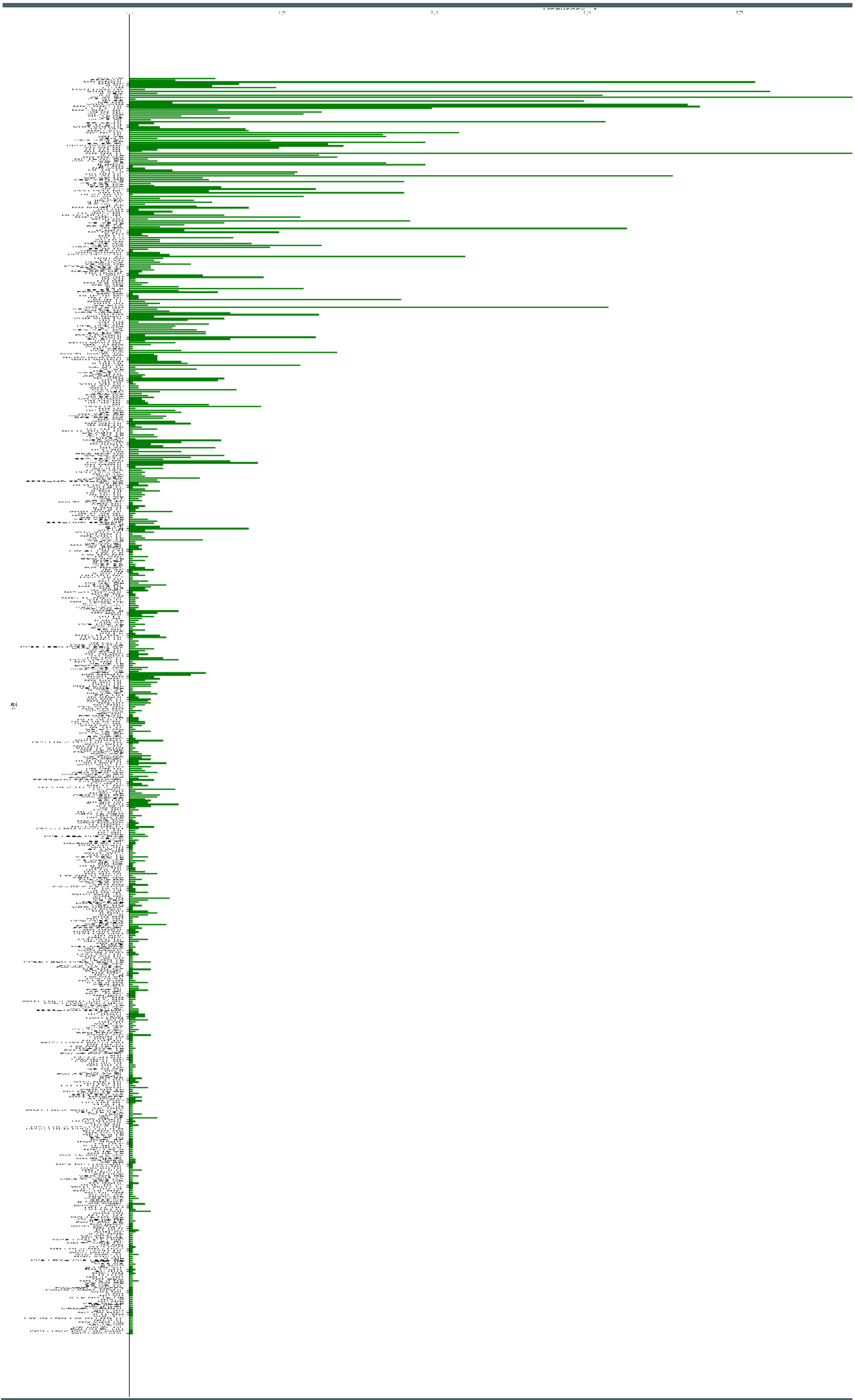
Now once we have emission probabitlties I have found the sequence of tag as bigr ams. This will be used In finding the transition probabilities. Once we have transit ion probabilities. I have found out what are the possible tags which are given to a word in the brown corpus.

Now everything is set up we need to implement the Viterbi algorithm. The comple te detailed/commented code is given in the jupyter notebook.

After training and testing My model's Accuracy is around: **92.0%**
And loss is around **0.07**

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 clickihere.
So if you have a look at the x-axis you with find each labels of the x-axis in this 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.

	Ground-Truth	Predicted-Tag	Misclassified-Word	Sentence	frequency
0	IN	IN-HL	['of', 'into', 'from', 'over', 'in', 'off', 'o...	['## also , he was weary of plantation drudger...	276
1	NN	NN-TL	['horror', 'life', 'fox', 'county', 'hospital'...	['## guests stared with horror at madame lalau...	276
2	VBN	VBD	['warned', 'troubled', 'jumped', 'missed', 'co...	["## had dandy been older or wiser , instinct ...	210
3	NN	NN-HL	['house', 'basket', 'politics', 'food', 'churc...	['## `` dandy is to be our house guest , louis...	205
4	NNS	NNS-HL	['brothers', 'guests', 'arms', 'records', 'nut...	['## airless and dingy though it was , the att...	187
5	VBD	VBN	['represented', 'uttered', 'pronounced', 'star...	['## airless and dingy though it was , the att...	183
6	AT	AT-HL	['the', 'an', 'a', 'no']	["## so the verdict was `` death at the hands ...	178
7	IN	TO	['to']	['## he liked to savor his meat before he tast...	163
8	NN-TL	NN	['man', 'mass', 'ash', 'exchange', 'grandma', ...	['## they neither liked nor disliked the old m...	157
9	.	.-HL	['.', '!', '?']	["## when he finally left the sinister mansion...	156

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 words 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.

Experiment 2:

I tried to generate the text using my HMM model. The following are some result:

```
Sentence 0  there's his water '' .
Sentence 1  that the time of the new time . .
Sentence 2  did not make the years of time , and time time and the other time years .
Sentence 3  here '' , the president president , of new years , mr. mr. , only of his time is the mr. said of mr. .
Sentence 4  is our president of york president .
Sentence 5  comedie dolce comedie , that he said the going time that i make of years and new of one years are made of going of it .
Sentence 6  habe le facto , he was made time said it of this time would make it , time was the new president mr. and only , new , going the
Sentence 7  birds' time to make his years make time '' of other years only made .
Sentence 8  se is made , i said only .
Sentence 9  first new new and years said up of which i would be made to be new .
```

Observation:

Here I have generated only 10 sentences. The results are not very practical but see ms interesting to me. Following are my key observations

- Almost all sentences ends with a full stop “.”
- Subsentence of long sentence (3-5 word subsentence) have meaning in English(like “did not make the years ”, “is our president of your”)

NOTE: The above result may not be found in the jupyter notebook. But you can de finitely find the similar or better results.

Possible Reason for this (negative) observation:

- My algorithm to use the HMM to predict the sentence is very noob.
- In my algorithm I select a ramdom starting tag. then I find the best next tag based on the transition probability, keep on find the best next tag until I ge t the end tag i.e. “&&”. Then once I got the tag sequence I choose the word c orresponding to each tag with highest emission probability.

Possibility of future work:

Even though the way I used the my HMM model to generate the random text is ve ry basic, I got some sentence with literal meaning. If that algorithm will be modifi ed then one can get far better result. One should train this same model on more d ata to get higher accuracy. otherwise one can try different n-
gram to improve the model. Since For now we are able to generate some sentence this could be the first step toward the idea of “Talking Computer”.