towardsdatascience.com

Part Of Speech Tagging for Beginners

Kurtis Pykes

9–11 minutes

Natural Language Processing Notes

Photo by Edho Pratama on Unsplash

Part-of-speech (POS) tagging is a popular Natural Language Processing process which refers to categorizing words in a text (corpus) in correspondence with a particular part of speech, depending on the definition of the word and its context.

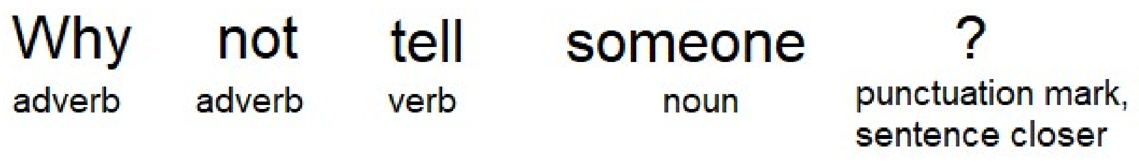
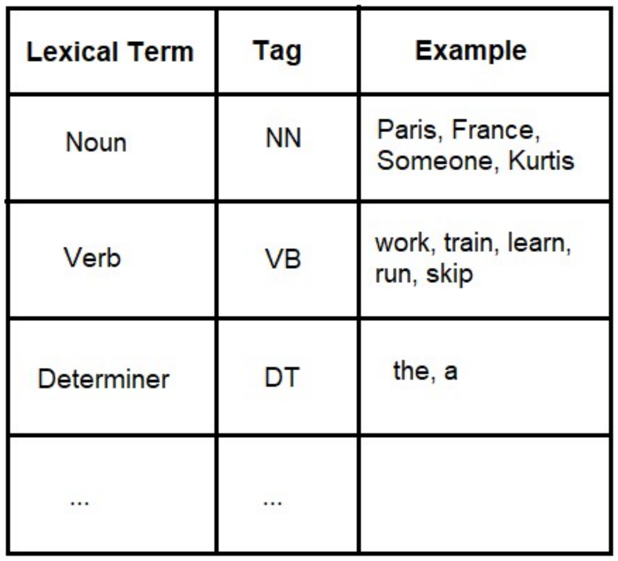


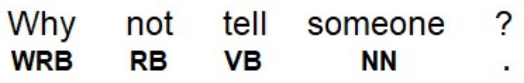
Figure 1: Example of POS tagging (Image by Author)

In Figure 1, we can see each word has its own lexical term written underneath, however, having to constantly write out these full terms when we perform text analysis can very quickly become cumbersome – especially as the size of the corpus grows. Thence, we use a short representation referred to as "tags" to represent the categories.

As earlier mentioned, the process of assigning a specific tag to a word in our corpus is referred to as part-of-speech tagging (POS tagging for short) since the POS tags are used to describe the lexical terms that we have within our text.



Lexical Term	Tag	Example
Noun	NN	Paris, France, Someone, Kurtis
Verb	VB	work, train, learn, run, skip
Determiner	DT	the, a
...	...	



Why	not	tell	someone	?
WRB	RB	VB	NN	.

Figure 2: Grid displaying different types of lexical terms, their tags, and random examples (Image By Author)

Part-of-speech tags describe the characteristic structure of lexical terms within a sentence or text, therefore, we can use them for making assumptions about semantics. Other applications of POS tagging include:

- Named Entity Recognition
- Co-reference Resolution
- Speech Recognition

When we perform POS tagging, it’s often the case that our tagger will encounter words that were not within the vocabulary that was used. Consequently, augmenting your dataset to include unknown word tokens will aid the tagger in selecting appropriate tags for those words.

Markov Chains

Photo by [Matthew Lancaster](#) on [Unsplash](#)

Taking the example text we used in Figure 1, "*Why not tell someone?*", imaging the sentence is truncated to "Why not tell … " and we want to determine whether the following word in the sentence is a noun, verb, adverb, or some other part-of-speech.

Now, if you are familiar with English, you'd instantly identify the verb and assume that it is more likely the word is followed by a noun rather than another verb. Therefore, the idea as shown in this example is that the POS tag that is assigned to the next word is dependent on the POS tag of the previous word.

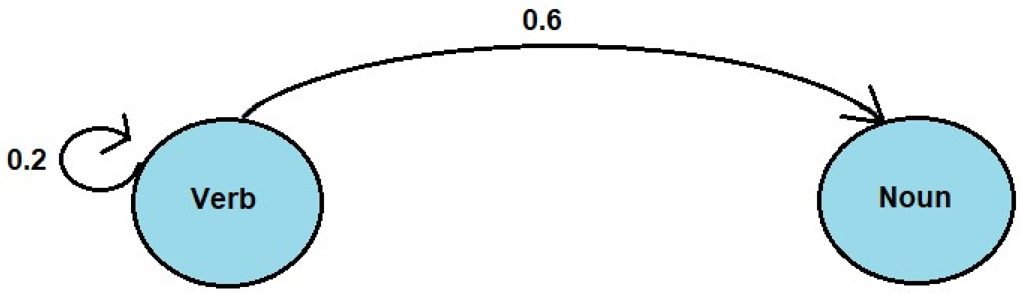


Figure 3: Representing Likelihoods visually (Image by Author)

By associating numbers with each arrow direction, of which imply the likelihood of the next word given the current word, we can say there is a higher likelihood the next word in our sentence would be a noun since it has a higher likelihood than the next word being a verb if we are currently on a verb. The image in Figure 3, is a great example of how a Markov Model works on a very small scale.

Given this example, we may now describe markov models as "*a stochastic model used to model randomly changing systems. It is assumed that future states depend only on the current state, not on the events that occurred before it (that is, it assumes the Markov property).*"

(Source: Wikipedia.)). Therefore to get the probability of the next event, it needs only the states of the current event.

We can depict a markov chain as directed graph:

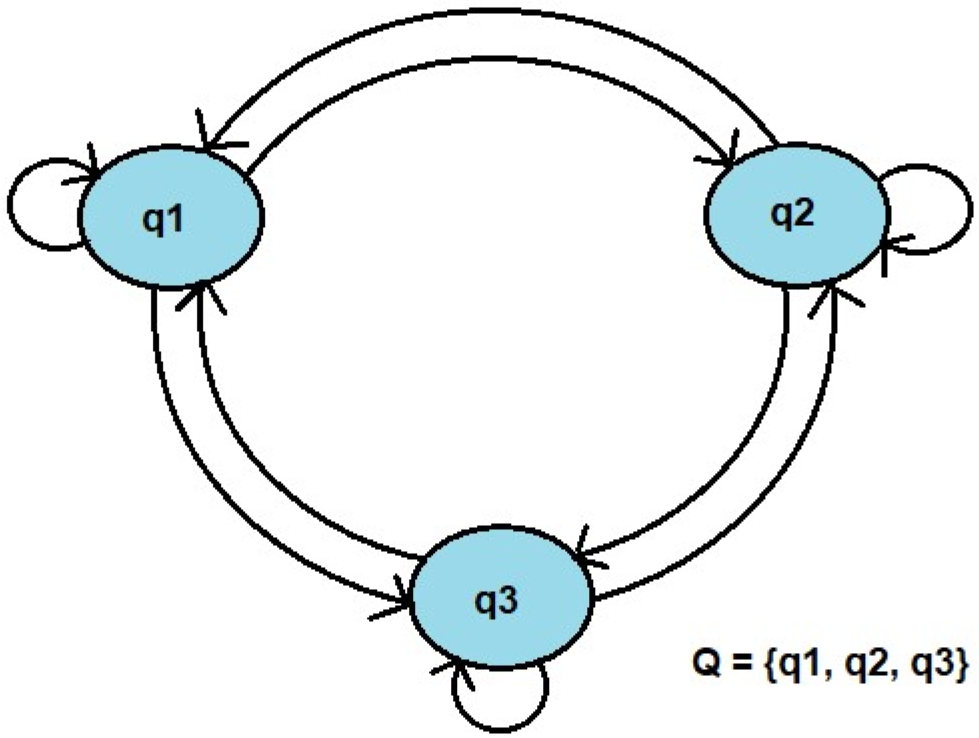
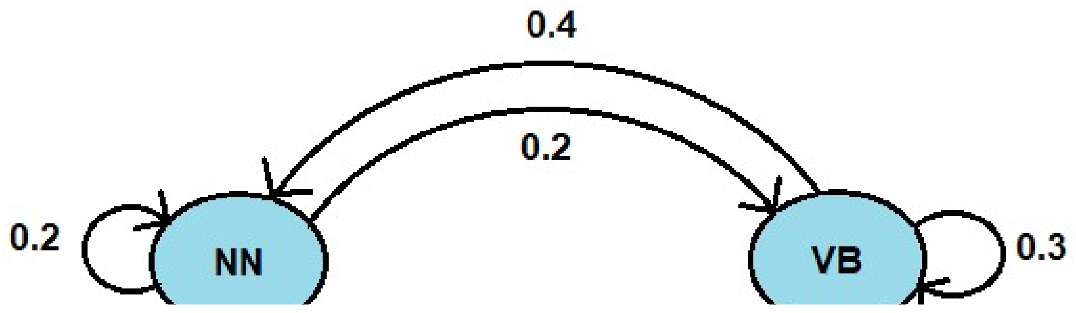


Figure 4: Depiction of Markov Model as Graph (Image By Author) – Replica of the image used in NLP Specialization Coursera Course 2, Week 2.

The lines with arrows are an indication of the direction hence the name "directed graph", and the circles may be regarded as the states of the model – a state is simply the condition of the present moment.

We could use this Markov model to perform POS. Considering we view a sentence as a sequence of words, we can represent the sequence as a graph where we use the POS tags as the events that occur which would be illustrated by the stats of our model graph.

For example, q1 in Figure 4 would become NN indicating a noun, q2 would be VB which is short for verb, and q3 would be O signifying all other tags that are not NN or VB. Like in Figure 3, the directed lines would be given a transition probability that define the probability of going from one state to the next.



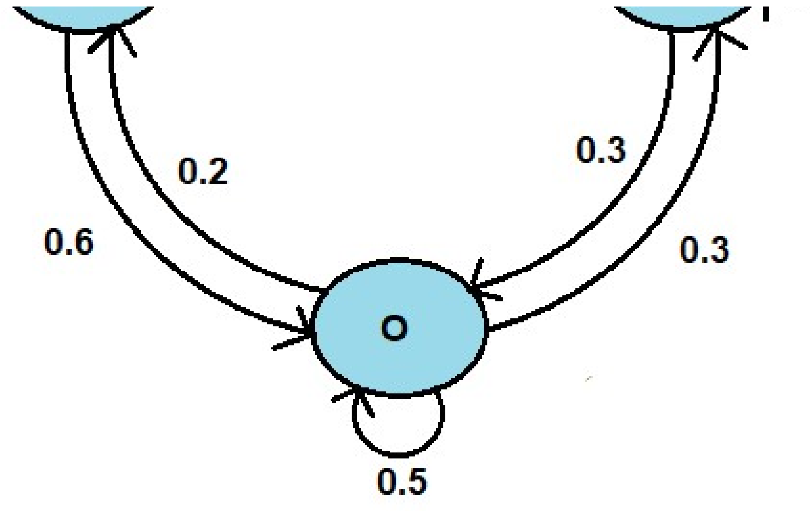
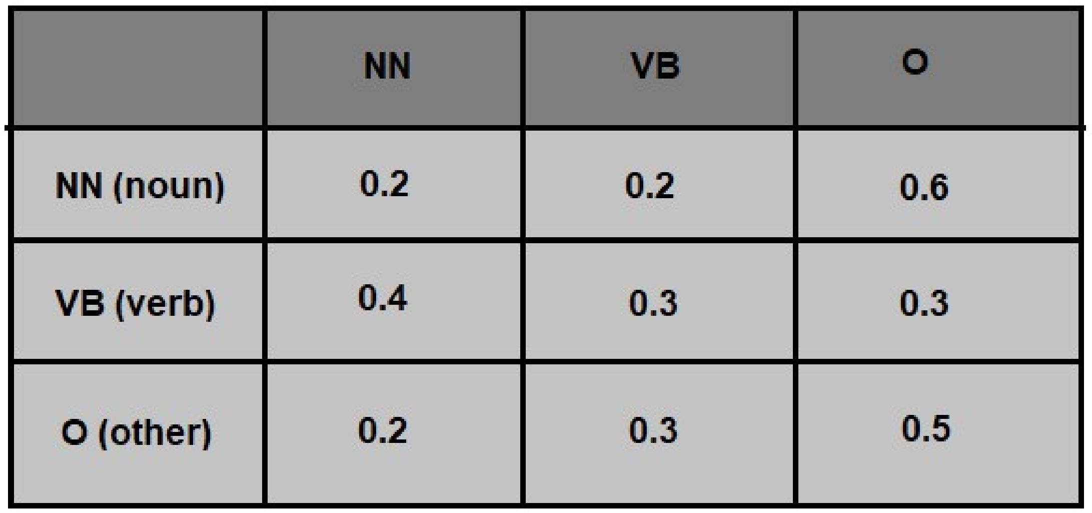


Figure 5: Example of Markov Model to perform POS tagging. (Image by Author)

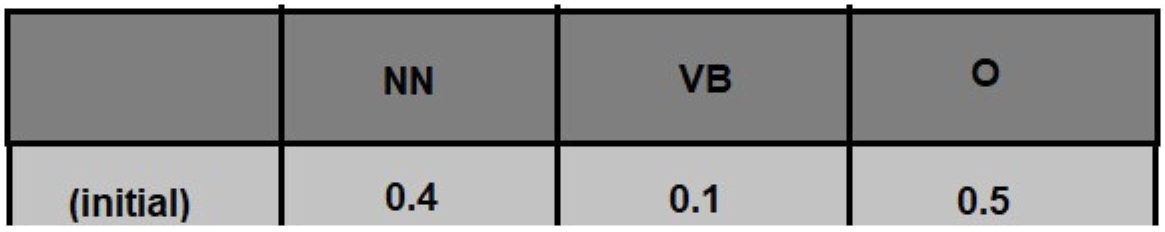
A more compact way to store the transition and state probabilities is using a table, better known as a "transition matrix".



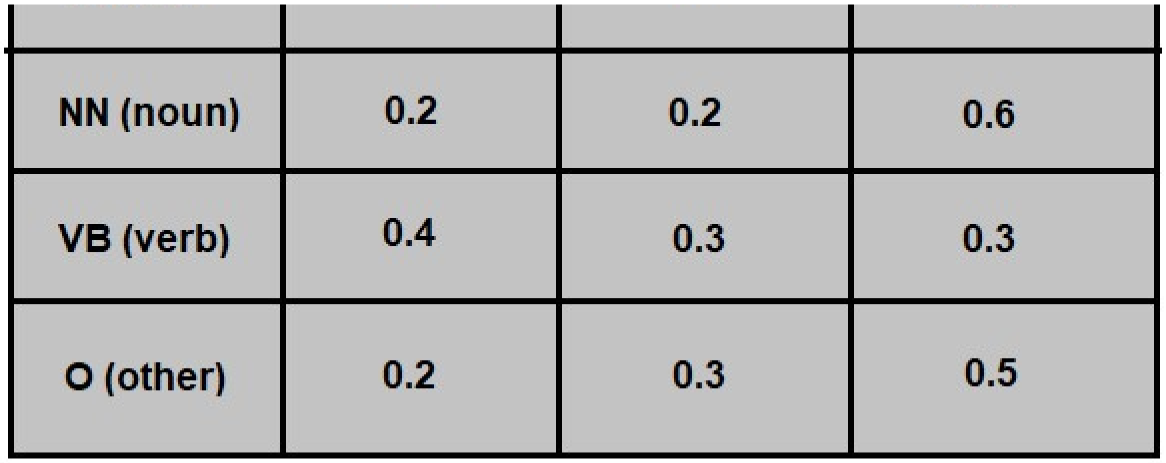
	NN	VB	O
NN (noun)	0.2	0.2	0.6
VB (verb)	0.4	0.3	0.3
O (other)	0.2	0.3	0.5

Figure 6: Transition Matrix (Image by Author)

Notice this model only tells us the transition probability of one state to the next when we know the previous word. Hence, this model does not show us what to do when there is no previous word. To handle this case, we add what is known as the "initial state".



	NN	VB	O
(initial)	0.4	0.1	0.5



NN (noun)	0.2	0.2	0.6
VB (verb)	0.4	0.3	0.3
O (other)	0.2	0.3	0.5

Figure 7: Adding an Initial State to deal with beginning of word matrix (Image by Author)

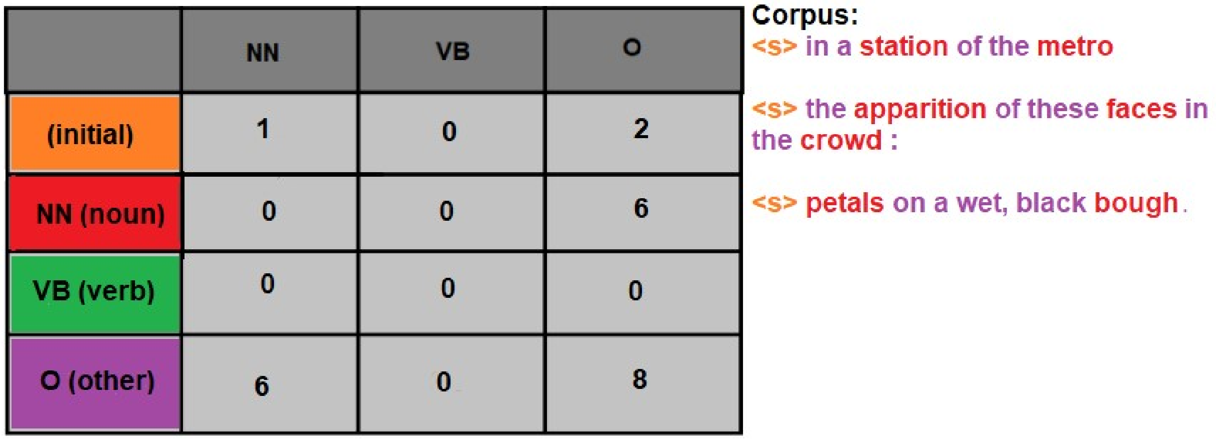
You may now be wondering, *how did we populate the transition matrix?* Great Question. I will use 3 sentences for our corpus. The first is "~~ in a station of the metro", " the apparition of these faces in the crowd", " petals on a wet, black bough." (Note these are the same sentences used in the course). Next, we will break down how to populate the matrix into steps:~~

1. ~~Count occurrences of tag pairs in the training dataset~~

$$C(t_{i-1}, t_i)$$

~~Figure 8: Counting the occurrences of the tag (Image by Author)~~

~~At the end of step one, our table would look something like this…~~



	NN	VB	O
(initial)	1	0	2
NN (noun)	0	0	6
VB (verb)	0	0	0
O (other)	6	0	8

Corpus:
<s> in a station of the metro

<s> the apparition of these faces in the crowd :

<s> petals on a wet, black bough.

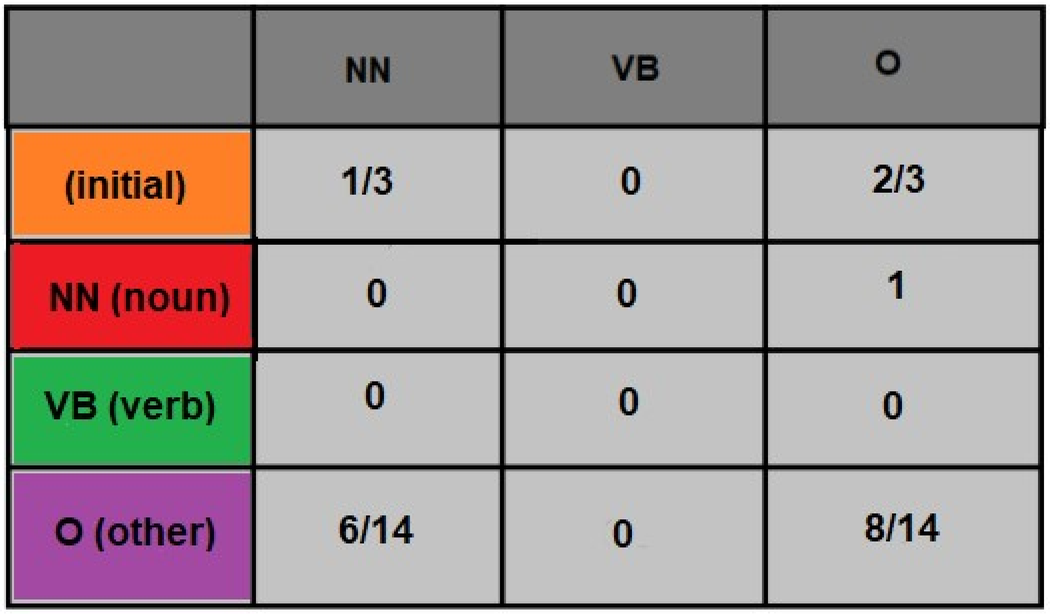
~~Figure 9: applying step one with our corpus. (Image by Author)~~

2. ~~Calculate the probability of using the counts~~

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1})}{\sum_{j=1}^{N} C(t_{i-1}, t_j)}$$

~~Figure 10: Calculate probabilities using the counts (Image by Author)~~

~~Appling the formula in Figure 10 to the table in Figure 9, our new table would look as follows…~~



	NN	VB	O
(initial)	1/3	0	2/3
NN (noun)	0	0	1
VB (verb)	0	0	0
O (other)	6/14	0	8/14

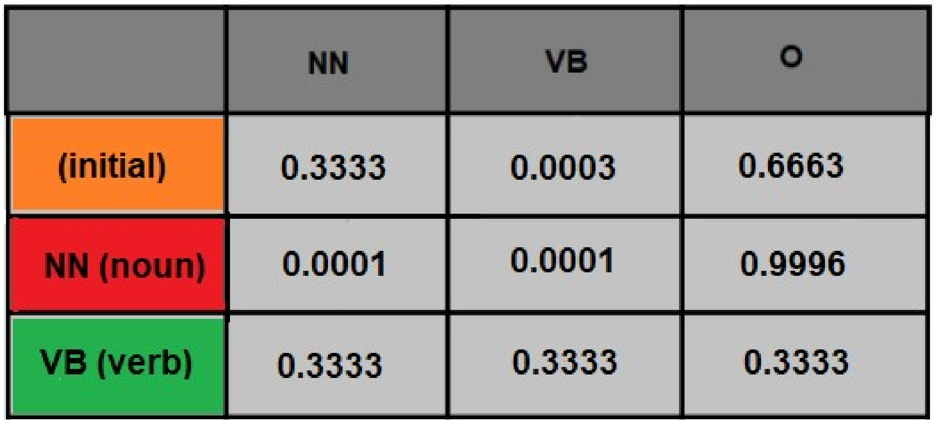
~~Figure 11: Probabilities populating the transition matrix. (Image by Author)~~

~~You may notice that there are many 0's in our transition matrix which would result in our model being incapable of generalizing to other text that may contain verbs. To overcome this problem, we add smoothing.~~

~~Adding smoothing requires we slightly we adjust the formula from Figure 10 by adding a small value, epsilon, to each of the counts in the numerator, and add N * epsilon to the denominator, such that the row sum still adds up to 1.~~

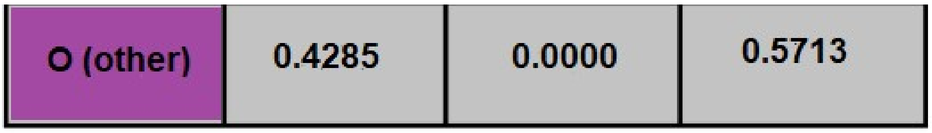
$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}) + \epsilon}{\sum_{j=1}^{N} C(t_{i-1}, t_j) + N * \epsilon}$$

~~Figure 12: Calculating the probabilities with smoothing (Image by Author)~~



	NN	VB	O
(initial)	0.3333	0.0003	0.6663
NN (noun)	0.0001	0.0001	0.9996
VB (verb)	0.3333	0.3333	0.3333

N = 3
epsilon = 0.001



O (other)	0.4285	0.0000	0.5713

~~Figure 13: New probabilities with smoothing added. N is the length of the corpus and epsilon is some very small number. (Image by Author)~~

~~Note: In a real world example, applying smoothing to the initial probabilities (the first row) as this would allow for a sentence to possibly start with any POS tag.~~

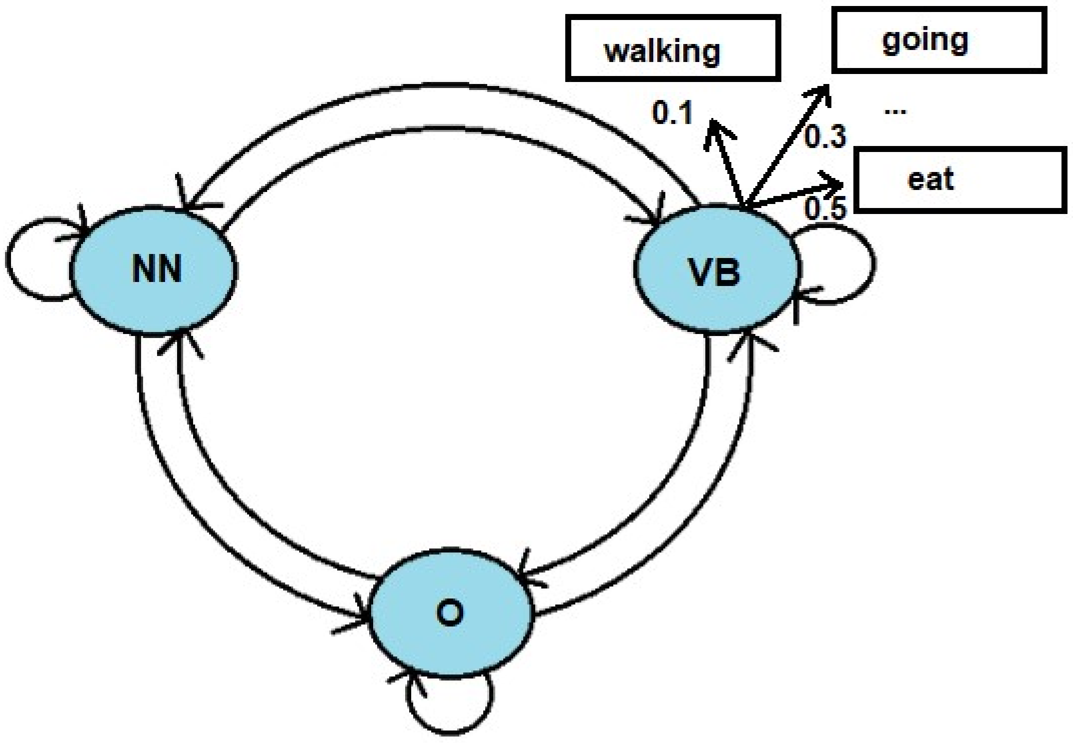
~~Hidden Markov Model~~

~~Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobservable ("*hidden*") states (Source: Wikipedia). In our case, the unobservable states are the POS tags of a word.~~

~~If we rewind back to our Markov Model in Figure 5, we see that the model has states for part of speech such as VB for verb and NN for a noun. We may now think of these as hidden states since they are not directly observable from the corpus. Though a human may be capable of deciphering what POS applies to a specific word, a machine only sees the text, hence making it observable, and is unaware of whether that word POS tag is noun, verb, or something else which in-turn means they are unobservable.~~

~~Both the Markov Model and Hidden Markov model have transition probabilities that describe the transition from one hidden state to the next, however, the Hidden Markov Model also has something known as emission probabilities.~~

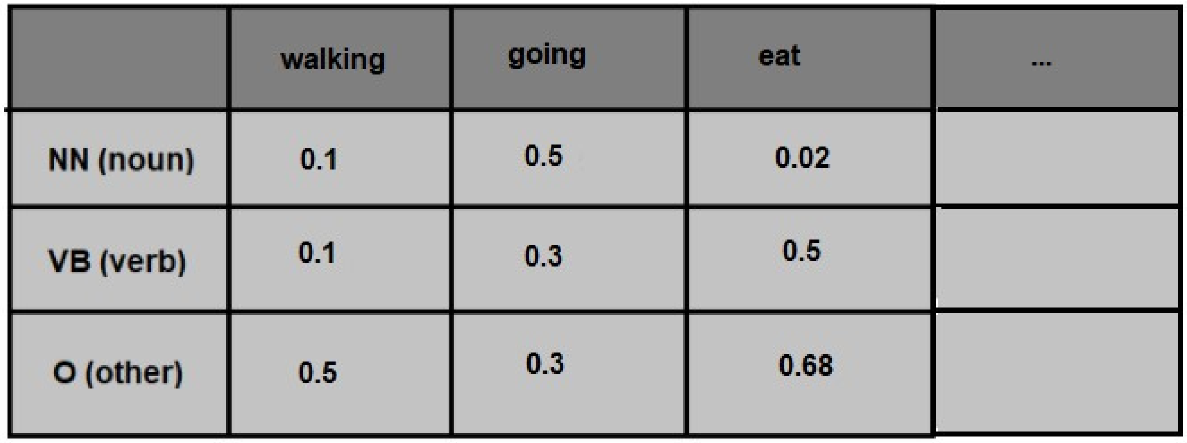
~~The emission probabilities describe the transitions from the hidden states in the model — remember the hidden states are the POS tags — to the observable states — remember the observable states are the words.~~



~~Figure 14: Example of Hidden Markov model. (Image by Author)~~

~~In Figure 14 we see that for the hidden VB state we have observable states. The emission probability from the hidden states VB to the observable eat is 0.5 hence there is a 50% chance that the model would output this word when the current hidden state is VB.~~

~~We can also represent the emission probabilities as a table…~~

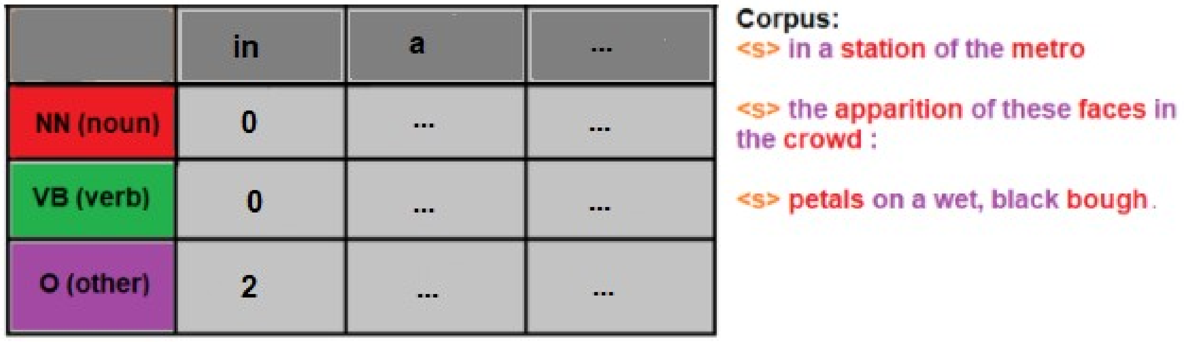


	walking	going	eat	...
NN (noun)	0.1	0.5	0.02	
VB (verb)	0.1	0.3	0.5	
O (other)	0.5	0.3	0.68	

~~Figure 15: Emission matrix expressed as a table — The numbers are not accurate representations, they are just random (Image by Author)~~

~~Similar to the transition probability matrix, the row values must sum to 1. Also, the reason all of our POS tags emission probabilities are more than 0 since words can have a different POS tag depending on the context.~~

~~To populate the emission matrix, we'd follow a procedure very similar to the way we'd populate the transition matrix. We'd first count how often a word is tagged with a specific tag.~~



	in	a	...
NN (noun)	0	...	...
VB (verb)	0	...	...
O (other)	2	...	...

Corpus:
<s> in a station of the metro

<s> the apparition of these faces in the crowd :

<s> petals on a wet, black bough .

~~Figure 16: Calculating the counts of a word and how often it is tagged with a specific tag.~~

~~Since the process is so similar to calculating the transition matrix, I will instead provide you with the formula with smoothing applied to see how it would be calculated.~~

$$P(w_i|t_i) = \frac{C(t_i, w_i) + \epsilon}{}$$

$$C(t_i) + N * \epsilon$$

~~Figure 17: Formula for calculating transition probabilities where N is the number of tags and epsilon is a very small number (Image by Author).~~

~~**Wrap Up**~~

~~You now know what a POS tag is and its different applications, as well as Markov Models, Hidden Markov Models, and transition and emission matrices and how to populate them with smoothing applied.~~

~~Thank you for reading to the end, feel free to connect with me on LinkedIn…~~

(~~**Kurtis Pykes – Data Scientist – Upwork | LinkedIn**~~