Computation of emission probability

Hidden
States HS1 → HS2 → HS3 → HS4 →HS5 → HS6 → HS7 → HS8 → ……………………..HSN
Observed
Sates OS1 → OS2 → OS3 → Os4 →OS5 → OS6 → OS7 → OS8 → ……………………..OSN

where HS(i) is pos tags and OS(i) is given word

emission probability is calculated as P(word/tag) = count(word and tag) / count(tag)
we will do these for all words present in training data and letter used it for testing that is for observed words
so we will have **emission_matrix** as

	tag1	tag2	tag3	tag4	.	.	.	tagN
word1	count(word1 and tag1) / count(tag1)	count(word1 and tag2) / count(tag2)	count(word1 and tag3) / count(tag3)	count(word1 and tag4) / count(tag4)	.	.	.	count(word1 and tagN) / count(tagN)
word2	count(word2 and tag1) / count(tag1)	count(word2 and tag2) / count(tag2)	count(word2 and tag3) / count(tag3)	count(word2 and tag4) / count(tag4)	.	.	.	count(word2 and tagN) / count(tagN)
word3	.	.	.	.	.	.	.	.
word4	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
wordN	count(wordN and tag1) / count(tag1)	count(wordN and tag2) / count(tag2)	count(wordN and tag3) / count(tag3)	count(wordN and tag4) / count(tag4)	.	.	.	count(wordN and tagN) / count(tagN)

wordi and tagi belongs to training data

Computation of transition probability

Here we will generate the transition probability matrix it is tag x tag matrix and each value represents the p(tag_i/tag_j)=count(tag_i and tag_j) / count(tag_j)

transition_matrix

	tag1	tag2	.	.	tagN
tag1	count(tag1 and tag1) / count(tag1)	count(tag1 and tag2) / count(tag2)	.	.	count(tag1 and tagN) / count(tagN)
tag2					
.					
.					
tagN	count(tagN and tag1) / count(tag1)	count(tagN and tag2) / count(tag2)			count(tagN and tagN) / count(tagN)

	ADJ	VERB	NUM	DET	PRON	PRT	CONJ	ADV	.	ADP	NOUN	X
ADJ	0.056893	0.018291	0.005378	0.006028	0.004727	0.020654	0.039116	0.111971	0.012057	0.092002	0.632403	0.000480
VERB	0.055460	0.182813	0.007285	0.152601	0.066860	0.071045	0.015127	0.088674	0.102953	0.164836	0.092136	0.000209
NUM	0.056691	0.046654	0.019145	0.012825	0.007435	0.004833	0.040892	0.289777	0.018959	0.128625	0.373978	0.000186
DET	0.222712	0.068682	0.009055	0.005476	0.009450	0.002353	0.000613	0.011546	0.016073	0.008679	0.643502	0.001858
PRON	0.008877	0.696104	0.000828	0.018168	0.006255	0.024608	0.011913	0.117060	0.054827	0.052160	0.009153	0.000046
PRT	0.019644	0.580802	0.004215	0.081915	0.008510	0.014156	0.015429	0.096867	0.045650	0.102275	0.030539	0.000000
CONJ	0.103482	0.218757	0.021771	0.144931	0.081990	0.027632	0.000279	0.022399	0.092317	0.070825	0.214988	0.000628
ADV	0.035533	0.091510	0.018665	0.140859	0.129119	0.040923	0.087010	0.148970	0.081794	0.101348	0.122857	0.001396
.	0.125768	0.225868	0.012744	0.065483	0.054004	0.030143	0.020246	0.203091	0.091649	0.140591	0.030324	0.000090
ADP	0.077002	0.040448	0.028783	0.465660	0.083272	0.015807	0.002223	0.007960	0.016358	0.020690	0.241417	0.000380
NOUN	0.012170	0.171115	0.009190	0.015968	0.019424	0.018152	0.060312	0.288262	0.028780	0.243172	0.133104	0.000352
X	0.006173	0.078189	0.000000	0.008230	0.002058	0.016461	0.024691	0.298354	0.014403	0.039095	0.067901	0.444444

here tagi belongs to training data
it follows markov property that is t th state depends on t-1 th state

computation of initial probabilities of hidden state(Pi)

it is calculated from transition_matrix
Repeated Matrix multiplication
pi = lim transition_matrix ^ n
n→ inf
but here we had take n as 10^2
but we can use **Monto carlo** or **left eigen vector** to compute the **pi**

Accuracy:

Algorithm Accuracy: 90.04975124378109

Trigram

Emission probabilities

	tag1&tag1	tag1&tag2	.	.	.	.	.	tagN&tagN
word1	count(word1 and tag1&tag1) / count(tag1&tag1)	count(word1 and tag1&tag2) / count(tag1&tag2)	.	.	.	.	.	count(word1 and tagN&tagN) / count(tagN&tagN)
word2	count(word2 and tag1&tag1) / count(tag1&tag1)	count(word2 and tag1&tag2) / count(tag1&tag2)	.	.	.	.	.	count(word2 and tagN&tagN) / count(tagN&tagN)
word3	.	.	.	.	.	.	.	.
word4	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
wordN	count(wordN and tag1&tagN) / count(tag1&tag1)	count(wordN and tag1&tag2) / count(tag1&tag2)	.	.	.	.	.	count(wordN and tagN&tagN) / count(tagN&tagN)

tarnsition probabilities

	tag1&tag1	tag1&tag2	.	.	tagN&tagN
tag1	count(tag1 and tag1&tag1) / count(tag1&tag1)	count(tag1 and tag1&tag2) / count(tag1&tag2)	.	.	count(tag1 and tagN&tagN) / count(tagN&tagN)
tag2					
.					
.					
tagN	count(tagN and tag1&tag1) / count(tag1&tag1)	count(tagN and tag1&tag2) / count(tag1&tag2)			count(tagN and tagN&tagN) / count(tagN&tagN)