**Code**

import nltk

import numpy as np

import pandas as pd

import random

from sklearn.model\_selection import train\_test\_split

import pprint, time

from IPython.display import display

nltk.download('treebank')

nltk.download('universal\_tagset')

nltk\_data = list(nltk.corpus.treebank.tagged\_sents(tagset='universal'))

train\_set,test\_set =train\_test\_split(nltk\_data,train\_size=0.80,test\_size=0.20,random\_state = 101)

train\_tagged\_words = [ tup for sent in train\_set for tup in sent ]

tags = {tag for word,tag in train\_tagged\_words}

def word\_given\_tag(word, tag, train\_bag = train\_tagged\_words):

tag\_list = [pair for pair in train\_bag if pair[1]==tag]

count\_tag = len(tag\_list)

w\_given\_tag\_list = [pair[0] for pair in tag\_list if pair[0]==word]

count\_w\_given\_tag = len(w\_given\_tag\_list)

return (count\_w\_given\_tag, count\_tag)

def t2\_given\_t1(t2, t1, train\_bag = train\_tagged\_words):

tags = [pair[1] for pair in train\_bag]

count\_t1 = len([t for t in tags if t==t1])

count\_t2\_t1 = 0

for index in range(len(tags)-1):

if tags[index]==t1 and tags[index+1] == t2:

count\_t2\_t1 += 1

return (count\_t2\_t1, count\_t1)

tags\_matrix = np.zeros((len(tags), len(tags)), dtype='float32')

for i, t1 in enumerate(list(tags)):

for j, t2 in enumerate(list(tags)):

tags\_matrix[i, j] = t2\_given\_t1(t2, t1)[0]/t2\_given\_t1(t2, t1)[1]

tags\_df = pd.DataFrame(tags\_matrix, columns = list(tags), index=list(tags))

def Viterbi(words, train\_bag = train\_tagged\_words):

state = []

T = list(set([pair[1] for pair in train\_bag]))

for key, word in enumerate(words):

p = []

for tag in T:

if key == 0:

transition\_p = tags\_df.loc['.', tag]

else:

transition\_p = tags\_df.loc[state[-1], tag]

emission\_p = word\_given\_tag(words[key], tag)[0]/word\_given\_tag(words[key], tag)[1]

state\_probability = emission\_p \* transition\_p

p.append(state\_probability)

pmax = max(p)

state\_max = T[p.index(pmax)]

state.append(state\_max)

return list(zip(words, state))

text = "Book a car. Park the car. The book is in the car. The car is in a park."

words = text.split()

POS\_tagging = Viterbi(words)

print("--> POS Tagging of words")

print()

print(POS\_tagging)

pos\_tags = []

for i in POS\_tagging:

if i not in pos\_tags:

pos\_tags.append(i[1])

Transition\_matrix = np.zeros((len(POS\_tagging), len(POS\_tagging)), dtype='float32')

for i, t1 in enumerate(tags):

for j, t2 in enumerate(POS\_tagging):

if t2\_given\_t1(t2[1], t1, POS\_tagging)[1] != 0:

Transition\_matrix[i, j] = t2\_given\_t1(t2[1], t1)[0]/t2\_given\_t1(t2[1], t1)[1]

else:

Transition\_matrix[i, j] = 0.0

print("--> Transition Matrix")

print()

Transition\_df = pd.DataFrame(Transition\_matrix, columns = list(pos\_tags), index=list(pos\_tags))

display(Transition\_df)

words = []

for i in POS\_tagging:

if i not in words:

words.append(i[0])

Emission\_matrix = np.zeros((len(POS\_tagging), len(POS\_tagging)), dtype='float32')

for i, t1 in enumerate(tags):

for j, t2 in enumerate(POS\_tagging):

Emission\_matrix[i, j] = word\_given\_tag(t2[0], t2[1], POS\_tagging)[0]

print("--> Emision Matrix")

print()

Emision\_df = pd.DataFrame(Emission\_matrix, columns = list(words), index=list(pos\_tags))

display(Emision\_df)

**Output**

