NLP ASSIGNMENT NIBHRIT GARG 19103110

Ques 1: Write a program that tokenizes the surface text and performs tasks:

a. Count the number of tokens

b. Word analysis

- Count the frequency of words

- Identify words belonging to different POS tags

```
import nltk
from nltk.tokenize import word_tokenize

/ 120s

s = "Enter the text: This is first. This has $5.00. This is Nr. Nibhrit Garg."

#Tokenising
tokens = word_tokenize(s)

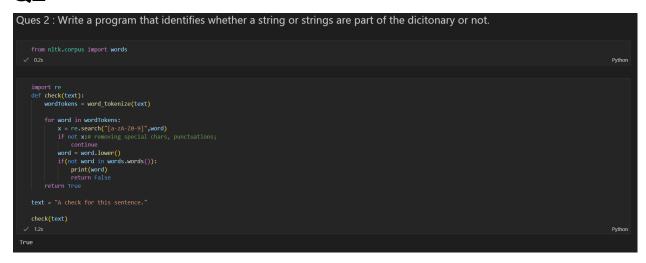
# Getting all tokens
print("sumber of tokens:", |en(tokens))
tokens

/ 0.3s

Number of tokens: 19

['Enter',
'the',
'text',
';',
'his',
'is',
'is',
'has',
's's',
's.00',
'.',
'his',
'is',
```

```
'5',
'5.00',
'7his',
'1s',
'1k',
'1k
```



Q3

Q4

Ques 5: Given a string as input. Write a program to identify the most probable POS tag sequence for the input using Viterbi algorithm.

	CONJ		х	VERB	DET	ADP	NOUN	PRT	ADV	NUM	PRON	ADJ
CONJ	0.000549	0.035126	0.009330	0.150384	0.123491	0.055982	0.349067	0.004391	0.057080	0.040615	0.060373	0.113611
	0.060079	0.092372	0.025641	0.089690	0.172192	0.092908	0.218539	0.002789	0.052569	0.078210	0.068769	0.046132
Х	0.010379	0.160869	0.075726	0.206419	0.056890	0.142226	0.061695	0.185086	0.025754	0.003075	0.054200	0.017682
VERB	0.005433	0.034807	0.215930	0.167956	0.133610	0.092357	0.110589	0.030663	0.083886	0.022836	0.035543	0.066390
DET	0.000431	0.017393	0.045134	0.040247	0.006037	0.009918	0.635906	0.000287	0.012074	0.022855	0.003306	0.206411
ADP	0.001012	0.038724	0.034548	0.008479	0.320931	0.016958	0.323589	0.001266	0.014553	0.063275	0.069603	0.107062
NOUN	0.042454	0.240094	0.028825	0.149134	0.013106	0.176827	0.262344	0.043935	0.016895	0.009144	0.004659	0.012584
PRT	0.002348	0.045010	0.012133	0.401174	0.101370	0.019569	0.250489	0.001174	0.009393	0.056751	0.017613	0.082975
ADV	0.006982	0.139255	0.022886	0.339022	0.071373	0.119472	0.032196	0.014740	0.081458	0.029868	0.012025	0.130721
NUM	0.014281	0.119243	0.202428	0.020707	0.003570	0.037487	0.351660	0.026062	0.003570	0.184220	0.001428	0.035345
PRON	0.005011	0.041913	0.088383	0.484738	0.009567	0.022323	0.212756	0.014123	0.036902	0.006834	0.006834	0.070615
ADJ	0.016893	0.066019	0.020971	0.011456	0.005243	0.080583	0.696893	0.011456	0.005243	0.021748	0.000194	0.063301
Time taken in seconds: 79.92651629447937												
Viterbi	Algorithm	ı Accuracy	: 93.779	9043062200	99							

```
#Check how a sentence is tagged by the two POS taggers
#and compare them

test_sent="will can see Marry"

# pred_tags_rule=Viterbi_rule_based(test_sent.split())

pred_tags_withoutRules= Viterbi(test_sent.split())

# print(pred_tags_withoutRules)

# will and Marry are tagged as NUM as they are unknown words for Viterbi Algorithm

166

Python

[('Will', 'CONJ'), ('can', 'VERB'), ('see', 'VERB'), ('Marry', 'CONJ')]
```

```
Ques 6: Explore different parsers and write code snippets to show their implementation.

import nltk
from nltk.tag import pos_tag
from nltk.tokenize import word_tokenize
from nltk.townize import tree
from nltk.townize import tree
from nltk.tag.mainport tree
from nltk.tag.mainport
from nltk.t
```

```
Ques 7: Write a program that takes sentence (atleast one word with multiple senses) as input and performs word sense disambiguation using the context
```

```
from nltk.wsd import lesk
from nltk.tokenize import word tokenize
al=lesk(word_tokenize('I like to eat bread jam'), 'jam')
print(al,al.definition())
a2 = lesk(word_tokenize('I am stuck in a traffic jam'), 'jam')
print(a2,a2.definition())

✓ 45s

Python

Synset('jam.v.06') crowd or pack to capacity
Synset('jam.v.05') get stuck and immobilized
```

Q8

Ques 8: Write a program for word generation given a context