Practical 7

Name: Shivam Tawari
Roll no: A-58
Subject: NLP

Aim: Write a python program for parts of speech tagging for users given sentence.

Theory:

Tagging a kind of classification is the automatic assignment of description of the tokens. We call the desprition tag which represents on the parts of speech semantic information and so on.

Parts of Speech (POS) tagging it may be defined as the process of converting a sentence in the form of list of words into a list of tupples. Here types are the form of (word tag). We can also, it as process of ~~ang~~ assignment one of the parts of speech to the given word.

Default tagging :

It is a basic step for the parts of speech tagging. It is performed using the default tagger class. The default tagger class takes 'tag' as single argument. NN is 'tag' as a single argument for singular nouns.

Default tagger is most useful when it gets to write with most common parts of speech tag.

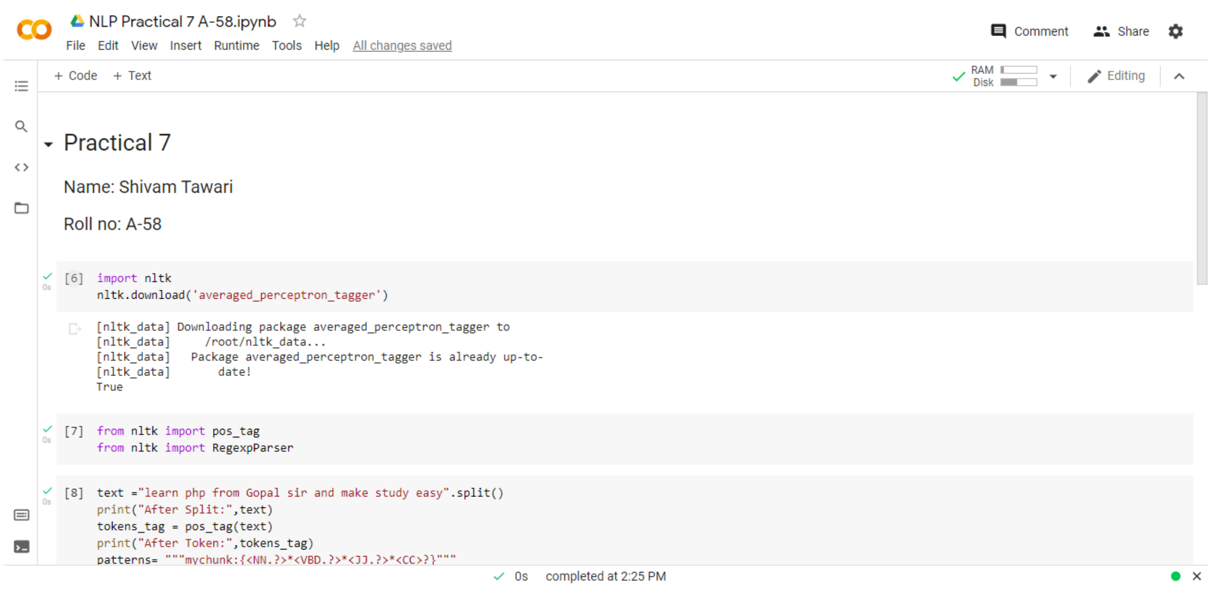
Conclusion: Hence we have successfully implemented python program for parts of speech (POS) tagging for users given sentence.

Practical 7

Name: Shivam Tawari

Roll no: A - 58

Code:


NLP Practical 7 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing

Practical 7

Name: Shivam Tawari

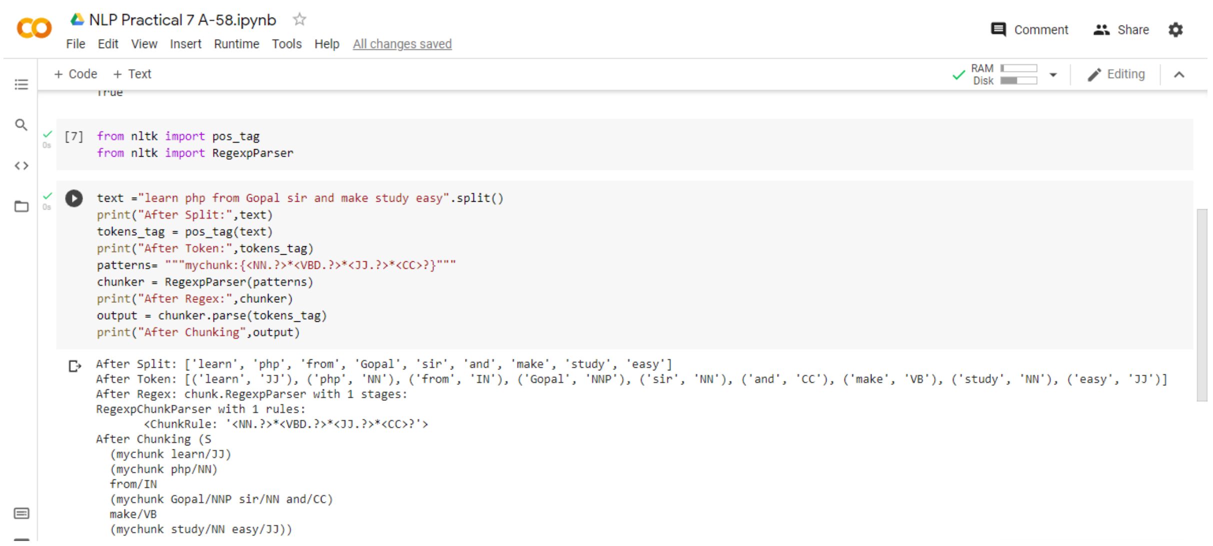
Roll no: A-58

```
[6] import nltk
    nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
True
```

```
[7] from nltk import pos_tag
    from nltk import RegexpParser
```

```
[8] text ="learn php from Gopal sir and make study easy".split()
    print("After Split:",text)
    tokens_tag = pos_tag(text)
    print("After Token:",tokens_tag)
    patterns= """mychunk:{<NN.?>*<VBD.?>*<JJ.?>*<CC>?}"""
```


NLP Practical 7 A-58.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing

```
True
```

```
[7] from nltk import pos_tag
    from nltk import RegexpParser
```

```
text ="learn php from Gopal sir and make study easy".split()
print("After Split:",text)
tokens_tag = pos_tag(text)
print("After Token:",tokens_tag)
patterns= """mychunk:{<NN.?>*<VBD.?>*<JJ.?>*<CC>?}"""
chunker = RegexpParser(patterns)
print("After Regex:",chunker)
output = chunker.parse(tokens_tag)
print("After Chunking",output)
```

```
After Split: ['learn', 'php', 'from', 'Gopal', 'sir', 'and', 'make', 'study', 'easy']
After Token: [('learn', 'JJ'), ('php', 'NN'), ('from', 'IN'), ('Gopal', 'NNP'), ('sir', 'NN'), ('and', 'CC'), ('make', 'VB'), ('study', 'NN'), ('easy', 'JJ')]
After Regex: chunk.RegexpParser with 1 stages:
RegexpChunkParser with 1 rules:
       <ChunkRule: '<NN.?>*<VBD.?>*<JJ.?>*<CC>?'>
After Chunking (S
  (mychunk learn/JJ)
  (mychunk php/NN)
  from/IN
  (mychunk Gopal/NNP sir/NN and/CC)
  make/VB
  (mychunk study/NN easy/JJ))
```