Vidya Vikas Education Trust's
Universal College of Engineering, Kaman Road, Vasai-401208
Accredited B+ Grade by NAAC

Experiment 7

Name of the Student: - Sanket Belekar
Roll No:- 67 Batch:- A3 Branch :- BE Comps-A
Date of Practical Performed: - Staff Signature with Date & Marks

Aim: Write a program to perform Part of Speech(POS) Tagging

Theory:

Text chunking, also referred to as shallow parsing, is a task that follows Part-Of-Speech Tagging and that adds more structure to the sentence. The result is a grouping of the words in "chunks" .

Chunk extraction or partial parsing is a process of meaningfully extracting short phrases from the sentence (tagged with Part-of-Speech). Chunks are made up of words and the kinds of words are defined using the part-of-speech tags.

A Chunking activity involves breaking down a difficult text into more manageable pieces and having students rewrite these "chunks" in their own words.
Now that we know the parts of speech, we can do what is called chunking, and group words into hopefully meaningful chunks.
One of the main goals of chunking is to group into what are known as "noun phrases." These are phrases of one or more words that contain a noun, maybe some descriptive words, maybe a verb, and maybe something like an adverb. The idea is to group nouns with the words that are in relation to them.
In order to chunk, we combine the part of speech tags with regular expressions.
Mainly from regular expressions, we are going to utilize the following:

+ = match 1 or more

? = match 0 or 1 repetitions.

* = match 0 or MORE repetitions.

. = Any character except a new line

See the tutorial linked above if you need help with regular expressions. The last things to note is that the part of speech tags are denoted with the "<" and ">" and we can also place regular expressions within the tags themselves, so account for things like "all nouns" ().

Code:

```
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk import pos_tag,ne_chunk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
def chunk_text(text):
  sentences= sent_tokenize(text)
  chunked_sentences=[]
  for sentence in sentences:
    words= word_tokenize (sentence)
    pos_tags= pos_tag (words)
    chunked=ne_chunk(pos_tags)
    chunked_sentences.append(chunked)
    return chunked_sentences
input_text= "barck obama was the 44th presidentof the unitec state
the born in hawai."
chunked_output=chunk_text(input_text)
for i, sentences in enumerate (chunked_output):
  print (f"sentence{i+1}:{sentences}")
```

Output:

```
sentence1:(S
  barck/NN
  obama/NN
  was/VBD
  the/DT
  44th/JJ
  presidentof/NN
  the/DT
  unitec/JJ
  state/NN
  the/DT
  born/NN
  in/IN
  hawai/NN
  ./.)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[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!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Conclusion: - Hence we have implement chunking in NLP.