Aim- To understand the concept of chunking and get familior with the basic chunk target

Theory - Chunk extraction or partial passing is a process of meaningfull extracting short phrases from the sentence tagged with parts of speech. Chunks are made up of words & the kinds of words are defined using the POS tags. One can even define a pattern or words that can't be a part of a chunk & such words are known as chunks

Defining chunk patterns.

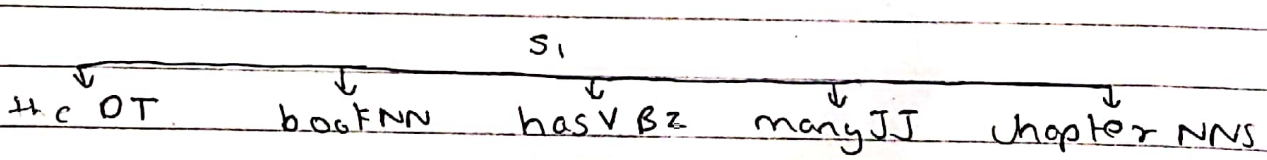
Chunk pattern are normal regular expression which are modified & designed to match the POS tag designed to match sequence of part of speech tags. Angle brackets are used to specify an individual tag for example to match a noun tag.

One can define multiple tags in the same way.

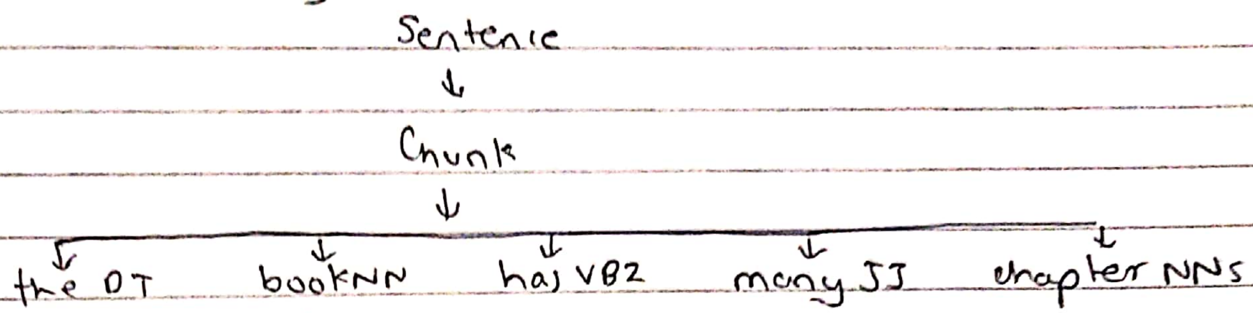
Steps to follow:-

1) Converting chunks to RegEx pattern
2) Passing the sentence using the Regex.

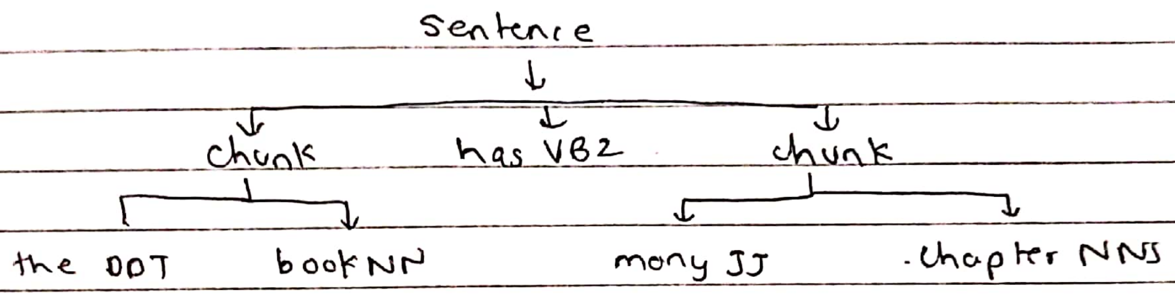
For eg. The book has many chapters
let us convert the sentence into flat tree



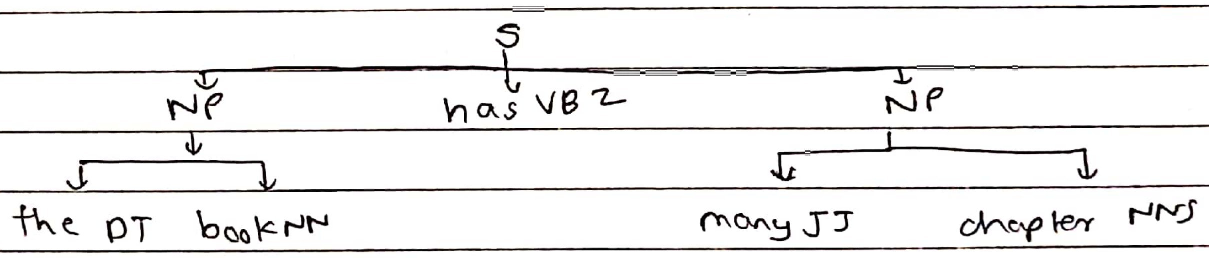
Step 2 - Forming chunks



Step 3 - Splitting bigger chunks into smaller chunks



Thus convert the chunktree into 2 subtree



∴ The chunks are
(the book, NP), (has, VP), (many chapter, NP)

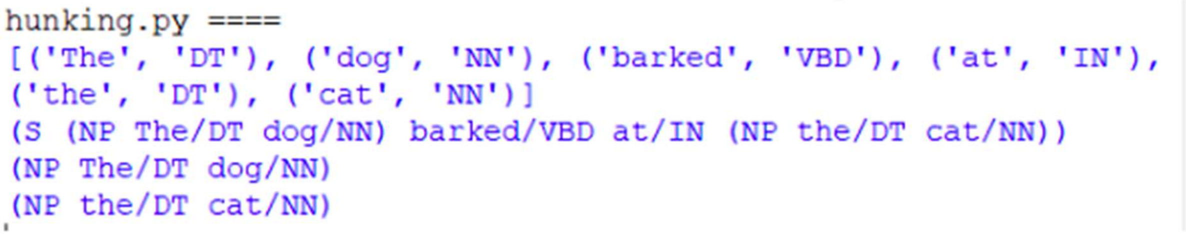
Conclusion : Thus we have successfully implemented chunking using basic chunk target

EXP 7: CHUNKING

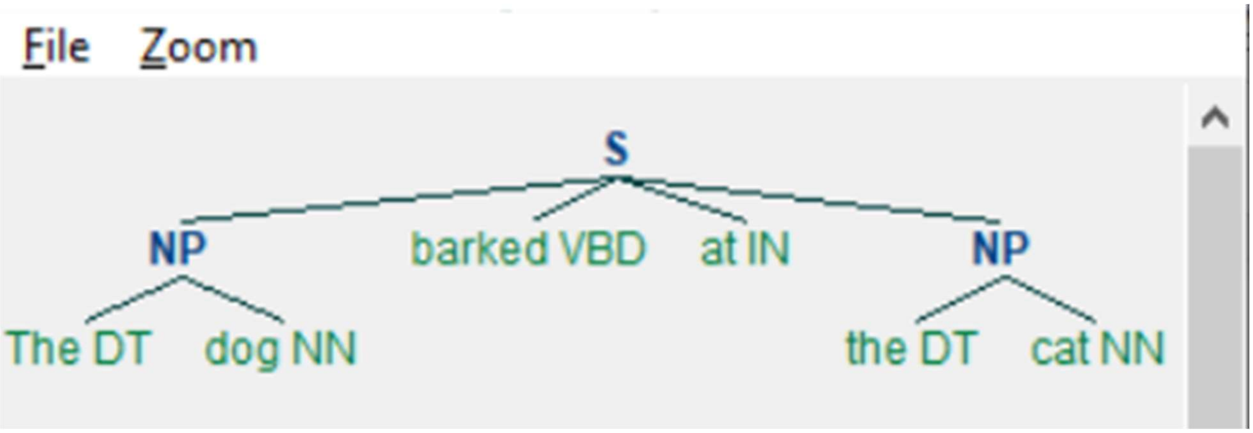
PROGRAM:

```
import nltk
txt = "The dog barked at the cat"
grammar = ('''
    NP: {<DT>?<JJ>*<NN>} # NP
    ''')
chunkParser = nltk.RegexpParser(grammar)
tagged = nltk.pos_tag(nltk.word_tokenize(txt))
print(tagged)
tree = chunkParser.parse(tagged)
for subtree in tree.subtrees():
    print(subtree)
tree.draw()
```

OUTPUT:



```
hunking.py ====
[('The', 'DT'), ('dog', 'NN'), ('barked', 'VBD'), ('at', 'IN'),
('the', 'DT'), ('cat', 'NN')]
(S (NP The/DT dog/NN) barked/VBD at/IN (NP the/DT cat/NN))
(NP The/DT dog/NN)
(NP the/DT cat/NN)
```


File
Zoom
S
NP
barked VBD
at IN
NP
The DT
dog NN
the DT
cat NN