Practical 3: Chunking and chinking

chunking means to extract the components of a sentance and saperate them into noun, adj, avd etc...

chinkinh means to remove a chunk from chunk

Steps involved in Chunking operation

1. Tokenize the sentence and perform POS Tagging

2. Define the grammar to perform the chunking. This is important step because grammar lays the rules of chunking

3. Using this grammar we create a chunk parser with the help of RegexpParser and apply it to our sentence

4. Draw the Graph for better visualization.

Import the required libraries

In [16]:
```
import nltk
from nltk.chunk import RegexpParser
```

Initialize the sentence

In [17]:
```
sentence = " The little yellow dog barkes at the cat"
```

Tokenize the sentence

In [18]:
```
tokens = nltk.word_tokenize(sentence)
print(tokens)
```

```
['The', 'little', 'yellow', 'dog', 'barkes', 'at', 'the', 'cat']
```

Perform POS Tagging

In [19]:
```
tag = nltk.pos_tag(tokens)
print(tag)
```

```
[('The', 'DT'), ('little', 'JJ'), ('yellow', 'JJ'), ('dog', 'NN'), ('barkes', 'NN
S'), ('at', 'IN'), ('the', 'DT'), ('cat', 'NN')]
```

Define the grammar rules

In [20]:
```
grammar = "NP: {<DT>?<JJ>*<NN>}"
```

Create chunk parser for this grammar using RegexpParser

In [21]:
```
chunk_parser = nltk.RegexpParser(grammar)
result = chunk_parser.parse(tag)
```

Print the results

In [27]:
```
print(result)
```

```
(S
  (NP The/DT little/JJ yellow/JJ dog/NN)
  barkes/NNS
  at/IN
  (NP the/DT cat/NN))
```

Print using .draw() methos for better visualization

In [33]:
```
result.draw()
```

In [39]:
```
!pip install opencv-python
```

```
Collecting opencv-python
  Downloading opencv_python-4.7.0.72-cp37-abi3-win_amd64.whl (38.2 MB)
Requirement already satisfied: numpy>=1.19.3 in c:\users\hp\anaconda3\lib\site-packa
ges (from opencv-python) (1.20.3)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.7.0.72
```

In [ ]:
```
import nltk
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer

train_text = state_union.raw("2005-GWBush.txt")
sample_text = state_union.raw("2006-GWBush.txt")

custom_sent_tokenizer = PunktSentenceTokenizer(train_text)

tokenized = custom_sent_tokenizer.tokenize(sample_text)

def process_content():
    try:
        for i in tokenized[5:]:
            words = nltk.word_tokenize(i)
            tagged = nltk.pos_tag(words)

            chunkGram = r"""Chunk: {<.*>+}
                                }<VB.?|IN|DT|TO>+{"""

            chunkParser = nltk.RegexpParser(chunkGram)
            chunked = chunkParser.parse(tagged)

            chunked.draw()

    except Exception as e:
        print(str(e))

process_content()
```

Chinking operation

In [1]:
```
import nltk
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer
```

In [6]:
```
sentence2 = "This is a sentence that means nothing"
```

In [7]:
```
#### tokenize the sentence
```

In [8]:
```
tokens = nltk.word_tokenize(sentence2)
print(tokens)
```

```
['This', 'is', 'a', 'sentence', 'that', 'means', 'nothing']
```

In [10]:
```
tags = nltk.pos_tag(tokens)
print(tags)
```

```
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sentence', 'NN'), ('that', 'WDT'),
('means', 'VBZ'), ('nothing', 'NN')]
```

In [14]:
```
grammar_for_chinking = r"""Chunk: {<.*>+}
                                  }<VB.?|IN|DT|TO>+{"""
```

In [16]:
```
chunk_parser = nltk.RegexpParser(grammar_for_chinking)
result = chunk_parser.parse(tags)
```

In [17]:
```
result.draw()
```

In [ ]:
```

```