Name: Roja Kamble

UNT ID: 11454258

Replace/change the code and input given sentence of Natura l Language Processing example of CSCE 5200 course Chapter 5. For this purpose, use own original names to mark variables, size and meanings of input data in the code. Execute the code in any relevant programming language. Describe the problem statement, input data, used methods, code and results obtained. Upload the file consisting of the code and related report to the UNT Canvas environment.

In [2]:
```
import nltk
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml (https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[2]:
```
True
```

In [3]:
```
pip install gensim
```

```
Requirement already satisfied: gensim in /Users/roja/opt/anaconda3/lib/python3.9/site-packages (4.1.2)
Requirement already satisfied: numpy>=1.17.0 in /Users/roja/opt/anaconda3/lib/python3.9/site-packages (from gensim) (1.20.3)
Requirement already satisfied: smart-open>=1.8.1 in /Users/roja/opt/anaconda3/lib/python3.9/site-packages (from gensim) (5.2.1)
Requirement already satisfied: scipy>=0.18.1 in /Users/roja/opt/anaconda3/lib/python3.9/site-packages (from gensim) (1.7.1)
WARNING: You are using pip version 21.2.3; however, version 22.3 is available.
You should consider upgrading via the '/Users/roja/opt/anaconda3/bin/python -m pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
pip install pattern
```

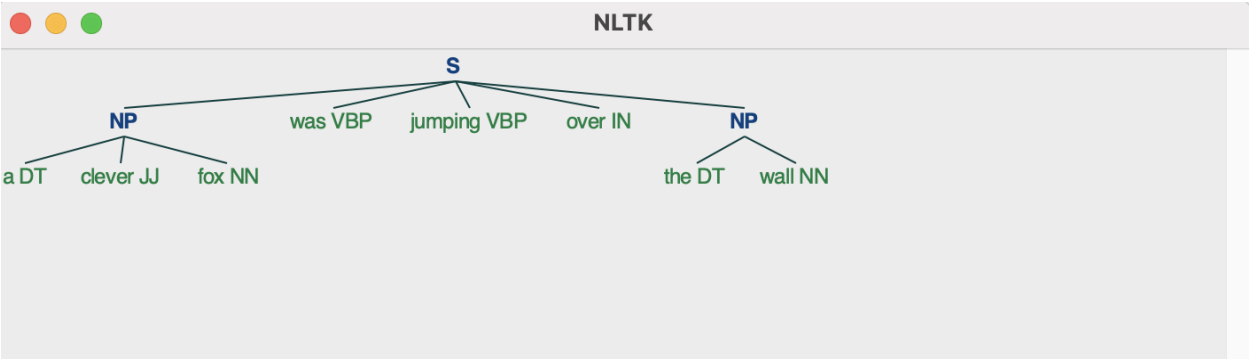
```
Collecting pattern
  Using cached Pattern-3.6.0.tar.gz (22.2 MB)
Requirement already satisfied: future in /Users/roja/opt/anaconda3/lib/py
thon3.9/site-packages (from pattern) (0.18.2)
Collecting backports.csv
  Using cached backports.csv-1.0.7-py2.py3-none-any.whl (12 kB)
Collecting mysqlclient
  Using cached mysqlclient-2.1.1.tar.gz (88 kB)
    ERROR: Command errored out with exit status 1:
     command: /Users/roja/opt/anaconda3/bin/python -c 'import io, os, sy
s, setuptools, tokenize; sys.argv[0] = '"'"'/private/var/folders/ty/4py3x
kds2xd231nxqy3v22j40000gp/T/pip-install-xoc66h6i/mysqlclient_d9c6f7002129
4877b4e26abb385d1737/setup.py'"'"'; __file__='"'"'/private/var/folders/t
y/4py3xkds2xd231nxqy3v22j40000gp/T/pip-install-xoc66h6i/mysqlclient_d9c6f
70021294877b4e26abb385d1737/setup.py'"'"';f = getattr(tokenize, '"'"'ope
n'"'"', open)(__file__) if os.path.exists(__file__) else io.StringIO
('"'"'from setuptools import setup; setup()'"'"');code = f.read().replace
('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__,
 '"'"'exec'"'"'))' egg_info --egg-base /private/var/folders/ty/4py3xkds2x
```

In [9]:
```
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.stem.lancaster import LancasterStemmer
```

In [10]:
```
sentence = [("a", "DT"),("clever","JJ"),("fox","NN"),("was","VBP"),
("jumping","VBP"),("over","IN"),("the","DT"),("wall","NN")]

grammar = "NP:{<DT>?<JJ>*<NN>}"

parser_chunking = nltk.RegexpParser(grammar)

parser_chunking.parse(sentence)

output = parser_chunking.parse(sentence)

print(output)

output.draw()

# Below,
# • DT is the determinant
# • VBP is the verb
# • JJ is the adjective
# • IN is the preposition
# • NN is the noun

# Note: When you run the code, the output is generates in separate window,
# or shown in ipynb unless you actually run. Hence included screenshot.
```

```
(S
  (NP a/DT clever/JJ fox/NN)
  was/VBP
  jumping/VBP
  over/IN
  (NP the/DT wall/NN))
```

Output Screenshot:

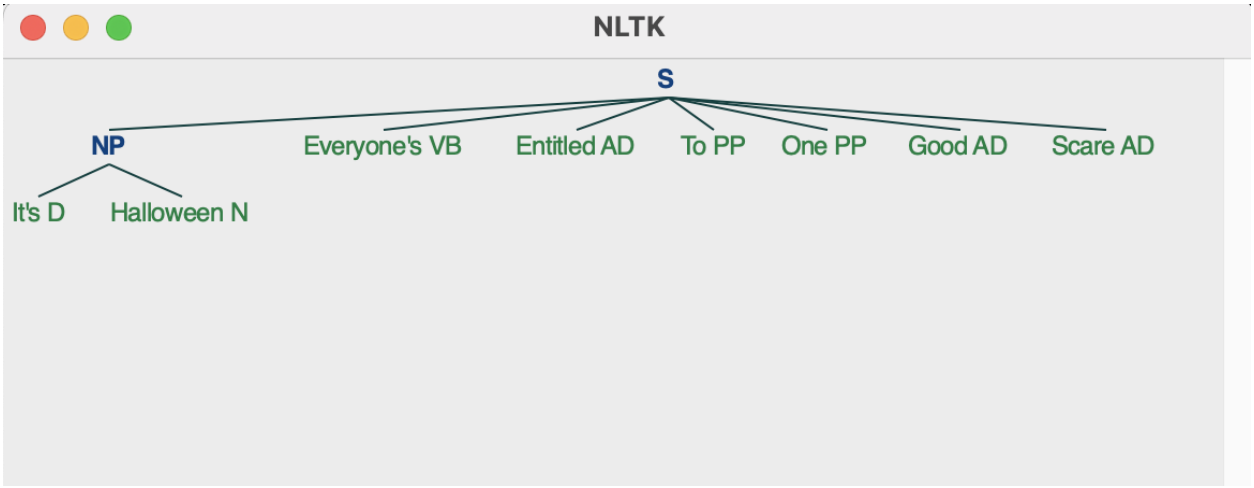


My code for parsing NLP with code and input change:

In [7]:
```
sen = [("It's", "D"),("Halloween", "N"),("Everyone's", "VB"),("Entitled","A
            ("Good","AD"),("Scare","AD")]

# Below,
# • D is the determinant
# • VB is the verb
# • AD is the adjective
# • PP is the preposition
# • N is the noun

grammar = "NP:{<D>?<AD>*<N>}"

txtParser = nltk.RegexpParser(grammar)

result = txtParser.parse(sen)

print(result)

result.draw()


# Note: When you run the code, the output is generates in separate window,
# or shown in ipynb unless you actually run. Hence included screenshot.
```

```
(S
  (NP It's/D Halloween/N)
  Everyone's/VB
  Entitled/AD
  To/PP
  One/PP
  Good/AD
  Scare/AD)
```

Output Screenshot:



Describe the problem statement, input data, used

methods, code and results obtained.

Problem Statement:

NLTK (Natural Language Toolkit), a Python toolkit created specifically for detecting and tagging elements of speech found in texts of natural language like English, is used to analyze text in English. Using NLP and the tokenization technique, determine the statements' fundamental meaning.

Input data:

It is an English sentence of your own choice.

Method:

1. NLP parsing approach includes NLTK library and tag patterns.
2. Tag patterns are used to describe groups of tagged words in the rules that make up a chunk of grammar. A tag pattern is a series of part-of-speech tags that are separated by angle brackets, such as “? *”. Like regular expression patterns are tag patterns.
3. If two matching instances of a tag pattern occur simultaneously, the leftmost match is given priority.
4. We'll start by thinking about the NP-chunking task, which involves looking for chunks that corresponds to certain noun phrases.
5. The Reg-exp Parser starts with a flat structure in which no tokens are chunked to determine the chunk structure for a given sentence. The chunk structure is updated by applying the chunking rules one at a time. The final chunk structure is returned when all the rules have been applied. Nothing but, we will first build a chunk grammar, which is a single regular-expression rule, to create an NP-chunk.

Information extraction is the process of quickly scanning text, searching for instances of a specific type of item, and discovering associations between objects.

A corpus of 500 million Web pages was analyzed by TEXTRUNNER to extract hundreds of millions of facts.

Code:

It is same as above.

Results:

The result is a tree, which we can either print, or display graphically. [When you run the code, output is generated as graphically display as shown in screenshot]

In [ ]: