Introduction To PUNKT Model using NLTK

In [1]:
```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from prettytable import PrettyTable
```

In [2]:
```
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\Uditya\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

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

In [3]:
```
text = "It's a dangerous business, Frodo, going out your door. You step onto t
```

In [4]:
```
words = word_tokenize(text)
sentence = sent_tokenize(text)
```

In [5]:
```
# words
sentence
```

Out[5]:
```
["It's a dangerous business, Frodo, going out your door.",
 "You step onto the road, and if you don't keep your feet, there's no knowing
where you might be swept off to."]
```

Filtering Stop Words

In [6]:
```
from nltk.corpus import stopwords
import spacy
```

In [7]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Uditya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[7]:
```
True
```

In [8]:
```
stop_words = stopwords.words('english')
```

In [9]:
```
print(stop_words)
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an',
'and', 'any', 'are', 'aren', "aren't", 'as', 'at', 'be', 'because', 'been', 'be
fore', 'being', 'below', 'between', 'both', 'but', 'by', 'can', 'couldn', "coul
dn't", 'd', 'did', 'didn', "didn't", 'do', 'does', 'doesn', "doesn't", 'doing',
'don', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'ha
d', 'hadn', "hadn't", 'has', 'hasn', "hasn't", 'have', 'haven', "haven't", 'hav
ing', 'he', "he'd", "he'll", 'her', 'here', 'hers', 'herself', "he's", 'him',
'himself', 'his', 'how', 'i', "i'd", 'if', "i'll", "i'm", 'in', 'into', 'is',
'isn', "isn't", 'it', "it'd", "it'll", "it's", 'its', 'itself', "i've", 'just',
'll', 'm', 'ma', 'me', 'mightn', "mightn't", 'more', 'most', 'mustn', "must
n't", 'my', 'myself', 'needn', "needn't", 'no', 'nor', 'not', 'now', 'o', 'of',
'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours', 'ourselves', 'out',
'over', 'own', 're', 's', 'same', 'shan', "shan't", 'she', "she'd", "she'll",
"she's", 'should', 'shouldn', "shouldn't", "should've", 'so', 'some', 'such',
't', 'than', 'that', "that'll", 'the', 'their', 'theirs', 'them', 'themselves',
'then', 'there', 'these', 'they', "they'd", "they'll", "they're", "they've", 't
his', 'those', 'through', 'to', 'too', 'under', 'until', 'up', 've', 'very', 'w
as', 'wasn', "wasn't", 'we', "we'd", "we'll", "we're", 'were', 'weren', "were
n't", "we've", 'what', 'when', 'where', 'which', 'while', 'who', 'whom', 'why',
'will', 'with', 'won', "won't", 'wouldn', "wouldn't", 'y', 'you', "you'd", "yo
u'll", 'your', "you're", 'yours', 'yourself', 'yourselves', "you've"]
```

In [10]:
```
nlp = spacy.load('en_core_web_sm')
spacy_stop_words = nlp.Defaults.stop_words
```

In [11]:
```
print(spacy_stop_words)
```

```
{'really', 'back', 'twenty', 'using', '’ll', 'had', 'and', 'becoming', 'made',
'anywhere', 'bottom', 'thereupon', 'under', 'beyond', 'part', 'during', 'namel
y', 'sometime', 'whither', '’ve', 'becomes', 'done', 'us', 'twelve', 'each', 't
op', 'seem', 'an', 'did', 'were', 'should', '‘s', 'after', 'thereby', 'from',
'every', 'became', 'eight', 'several', 'please', 'itself', 'him', 'n‘t', 'up',
'throughout', 'them', 'besides', 'among', 'because', '‘ve', 'nobody', 'mine',
'doing', 'last', 'whole', 'have', 'still', 'whereby', 'ours', 'take', 'amount',
'seemed', 'once', 'often', 'nor', 'therefore', 'a', 'too', '‘d', '’d', 'almos
t', 'nowhere', 'five', 'seems', 'whose', 'into', 'toward', 'anyway', 'could',
'nine', 'two', 'how', 'behind', 'all', "'re", 'meanwhile', 'he', 'ten', 'everyo
ne', 'anyhow', 'towards', 'used', '‘m', 'see', 'without', 'though', 'until', 'h
ereupon', 'also', 'due', 'been', 'can', 'you', 'former', "'ve", 'another', 'aft
erwards', 'anything', 'between', 'wherein', 'hereafter', 'at', 'eleven', 'becom
e', 'yourself', 'cannot', 'ca', 'they', 'their', 'which', 'whence', 'whenever',
'well', '’s', 'latter', 'it', 'therein', 'rather', 'we', 'so', 'those', 'alon
e', 'whereafter', 'thru', 'serious', 'was', 'whoever', 'not', 'with', 'alread
y', 'than', 'very', 'out', 'same', 'both', 'give', 'whereas', 'the', 'next', 'a
gain', 'call', 'someone', 'for', 'hers', 'would', '‘ll', 'no', 'herein', 'belo
w', 'neither', 'elsewhere', "'d", 'why', 'fifty', 'down', 'few', 'either', 'eve
rywhere', 'first', 'is', 'herself', 'now', 'about', 'empty', 'thence', 'unles
s', 'has', 'much', 'on', "n't", 'she', 'whom', 'his', 'regarding', 'n’t', 'alwa
ys', 'moreover', 'sometimes', 'do', 'yours', 'my', 'somewhere', 'to', 'whereupo
n', '’re', "'s", 'upon', 'whether', 'enough', 'whatever', 'else', 'what', 'ont
o', '’m', 'otherwise', 'via', 'per', 'of', 'then', 'everything', 'such', 'himse
lf', 'say', 'go', 'somehow', 'while', 'over', 'latterly', 'none', 'are', "'ll",
'three', 'in', 'across', 'show', 'own', 'noone', 'themselves', 'anyone', 'whe
n', 'yet', 'thereafter', 'indeed', 'wherever', 'forty', 'even', 'nothing', 'nev
er', 'myself', 'may', 'although', 're', 'other', 'six', 'along', 'if', 'just',
'beforehand', 'before', 'some', 'move', 'yourselves', 'most', 'who', 'hereby',
'name', 'might', 'four', 'make', 'this', 'be', 'ever', 'but', 'must', 'amongs
t', 'mostly', 'being', 'various', 'perhaps', 'me', 'sixty', 'these', 'ourselve
s', 'above', 'will', 'get', 'against', 'together', 'third', 'does', 'others',
'through', 'fifteen', 'put', 'side', 'less', 'am', 'formerly', 'least', 'furthe
r', 'seeming', 'keep', 'here', 'one', 'front', 'its', 'within', 'i', 'that', 'b
eside', 'by', 'hence', 'full', 'since', 'except', 'there', 'only', 'more', 'o
r', 'hundred', 'many', 'around', 'nevertheless', 'as', 'her', 'off', 'however',
'your', 'quite', 'thus', "'m", 'something', 'where', '‘re', 'any', 'our'}
```

In [12]:
```
len(stop_words), len(spacy_stop_words)
```

Out[12]:
```
(198, 326)
```

In [13]:
```
text = "Docker and Docker Compose have become essential tools for developers a
```

In [14]:
```
words = word_tokenize(text.lower())
```

In [15]:
```
sw_len = set([word for word in words if word in stop_words])
len(sw_len)
```

Out[15]:
```
28
```

In [16]:
```
print(sw_len)
```

```
{'or', 'of', 'll', 'once', 'most', 'will', 'and', 'up', 'out', 'this', 're', 'b
e', 'a', 'each', 'is', 'through', 'an', 'your', 'have', 'the', 'can', 'we', 'i
n', 'you', 'to', 'some', 'for', 'on'}
```

Stemming

In [17]:
```
from nltk.stem import PorterStemmer
text = 'The scientists discover new species every year, Last year, they discov
```

In [18]:
```
stemmer = PorterStemmer()
words = word_tokenize(text)
stemmed_words = [stemmer.stem(word) for word in words]

print(stemmed_words)
```

```
['the', 'scientist', 'discov', 'new', 'speci', 'everi', 'year', ',', 'last', 'y
ear', ',', 'they', 'discov', 'an', 'ancient', 'artifact', '.', 'they', 'are',
'discov', 'new', 'techniqu', 'with', 'their', 'recent', 'discoveri']
```

In [19]:
```
from prettytable import PrettyTable
table = PrettyTable()
table.field_names = ['word', 'stemmed word']
for word, stw in zip(words, stemmed_words):
        table.add_row([word, stw])

print(table)
```

```
+-------------+--------------+
|     word    | stemmed word |
+-------------+--------------+
|     The     |     the      |
|  scientists |  scientist   |
|   discover  |    discov    |
|     new     |     new      |
|   species   |    speci     |
|    every    |    everi     |
|     year    |     year     |
|      ,      |      ,       |
|     Last    |     last     |
|     year    |     year     |
|      ,      |      ,       |
|     they    |     they     |
|  discovered |    discov    |
|      an     |      an      |
|   ancient   |   ancient    |
|   artifact  |   artifact   |
|      .      |      .       |
|     they    |     they     |
|     are     |     are      |
| discovering |    discov    |
|     new     |     new      |
|  techniques |   techniqu   |
|     with    |     with     |
|    their    |    their     |
|    recent   |    recent    |
|  discovery  |  discoveri   |
+-------------+--------------+
```

Pos Tagging

In [20]:
```
from nltk import pos_tag
```

In [21]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Uditya\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[21]: True

In [22]:
```
tagged_words = pos_tag(words)
```

In [23]:
```
nltk.download('tagsets_json')
```

```
[nltk_data] Downloading package tagsets_json to
[nltk_data]     C:\Users\Uditya\AppData\Roaming\nltk_data...
[nltk_data]   Package tagsets_json is already up-to-date!
```

Out[23]: True

In [24]:
```
nltk.help.upenn_tagset()
```

```
$: dollar
    $ -$ --$ A$ C$ HK$ M$ NZ$ S$ U.S.$ US$
'': closing quotation mark
    ' ''
(: opening parenthesis
    ( [ {
): closing parenthesis
    ) ] }
,: comma
    ,
--: dash
    --
.: sentence terminator
    . ! ?
:: colon or ellipsis
    : ; ...
CC: conjunction, coordinating
    & 'n and both but either et for less minus neither nor or plus so
    therefore times v. versus vs. whether yet
CD: numeral, cardinal
    mid-1890 nine-thirty forty-two one-tenth ten million 0.5 one forty-
    seven 1987 twenty '79 zero two 78-degrees eighty-four IX '60s .025
    fifteen 271,124 dozen quintillion DM2,000 ...
DT: determiner
    all an another any both del each either every half la many much nary
    neither no some such that the them these this those
EX: existential there
    there
FW: foreign word
    gemeinschaft hund ich jeux habeas Haementeria Herr K'ang-si vous
    lutihaw alai je jour objets salutaris fille quibusdam pas trop Monte
    terram fiche oui corporis ...
IN: preposition or conjunction, subordinating
    astride among uppon whether out inside pro despite on by throughout
    below within for towards near behind atop around if like until below
    next into if beside ...
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first separable
    ectoplasmic battery-powered participatory fourth still-to-be-named
    multilingual multi-disciplinary ...
JJR: adjective, comparative
    bleaker braver breezier briefer brighter brisker broader bumper busier
    calmer cheaper choosier cleaner clearer closer colder commoner costlier
    cozier creamier crunchier cuter ...
JJS: adjective, superlative
    calmest cheapest choicest classiest cleanest clearest closest commonest
    corniest costliest crassest creepiest crudest cutest darkest deadliest
    dearest deepest densest dinkiest ...
LS: list item marker
    A A. B B. C C. D E F First G H I J K One SP-44001 SP-44002 SP-44005
    SP-44007 Second Third Three Two * a b c d first five four one six three
    two
MD: modal auxiliary
    can cannot could couldn't dare may might must need ought shall should
```

```
    shouldn't will would
NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed thermostat
    investment slide humour falloff slick wind hyena override subhumanity
    machinist ...
NNP: noun, proper, singular
    Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos
    Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin ODI Darryl CTCA
    Shannon A.K.C. Meltex Liverpool ...
NNPS: noun, proper, plural
    Americans Americas Amharas Amityvilles Amusements Anarcho-Syndicalists
    Andalusians Andes Andruses Angels Animals Anthony Antilles Antiques
    Apache Apaches Apocrypha ...
NNS: noun, common, plural
    undergraduates scotches bric-a-brac products bodyguards facets coasts
    divestitures storehouses designs clubs fragrances averages
    subjectivists apprehensions muses factory-jobs ...
PDT: pre-determiner
    all both half many quite such sure this
POS: genitive marker
    ' 's
PRP: pronoun, personal
    hers herself him himself hisself it itself me myself one oneself ours
    ourselves ownself self she thee theirs them themselves they thou thy us
PRP$: pronoun, possessive
    her his mine my our ours their thy your
RB: adverb
    occasionally unabatingly maddeningly adventurously professedly
    stirringly prominently technologically magisterially predominately
    swiftly fiscally pitilessly ...
RBR: adverb, comparative
    further gloomier grander graver greater grimmer harder harsher
    healthier heavier higher however larger later leaner lengthier less-
    perfectly lesser lonelier longer louder lower more ...
RBS: adverb, superlative
    best biggest bluntest earliest farthest first furthest hardest
    heartiest highest largest least less most nearest second tightest worst
RP: particle
    aboard about across along apart around aside at away back before behind
    by crop down ever fast for forth from go high i.e. in into just later
    low more off on open out over per pie raising start teeth that through
    under unto up up-pp upon whole with you
SYM: symbol
    % & ' '' ''. ) ). * + ,. < = > @ A[fj] U.S U.S.S.R * ** ***
TO: "to" as preposition or infinitive marker
    to
UH: interjection
    Goodbye Goody Gosh Wow Jeepers Jee-sus Hubba Hey Kee-reist Oops amen
    huh howdy uh dammit whammo shucks heck anyways whodunnit honey golly
    man baby diddle hush sonuvabitch ...
VB: verb, base form
    ask assemble assess assign assume atone attention avoid bake balkanize
    bank begin behold believe bend benefit bevel beware bless boil bomb
    boost brace break bring broil brush build ...
```

```
VBD: verb, past tense
    dipped pleaded swiped regummed soaked tidied convened halted registered
    cushioned exacted snubbed strode aimed adopted belied figgered
    speculated wore appreciated contemplated ...
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lactating
    hankerin' alleging veering capping approaching traveling besieging
    encrypting interrupting erasing wincing ...
VBN: verb, past participle
    multihulled dilapidated aerosolized chaired languished panelized used
    experimented flourished imitated reunifed factored condensed sheared
    unsettled primed dubbed desired ...
VBP: verb, present tense, not 3rd person singular
    predominate wrap resort sue twist spill cure lengthen brush terminate
    appear tend stray glisten obtain comprise detest tease attract
    emphasize mold postpone sever return wag ...
VBZ: verb, present tense, 3rd person singular
    bases reconstructs marks mixes displeases seals carps weaves snatches
    slumps stretches authorizes smolders pictures emerges stockpiles
    seduces fizzes uses bolsters slaps speaks pleads ...
WDT: WH-determiner
    that what whatever which whichever
WP: WH-pronoun
    that what whatever whatsoever which who whom whosoever
WP$: WH-pronoun, possessive
    whose
WRB: Wh-adverb
    how however whence whenever where whereby whereever wherein whereof why
``: opening quotation mark
    ` ``
```

In [25]:
```
table = PrettyTable(field_names=['Words', 'POS'])
for word, pos in zip(words, tagged_words):
        table.add_row([word, pos[1]])
print(table)
```

```
+-------------+------+
|    Words    | POS  |
+-------------+------+
|     The     |  DT  |
|  scientists | NNS  |
|   discover  | VBP  |
|     new     |  JJ  |
|   species   | NNS  |
|    every    |  DT  |
|     year    |  NN  |
|      ,      |  ,   |
|     Last    |  JJ  |
|     year    |  NN  |
|      ,      |  ,   |
|     they    | PRP  |
|  discovered | VBD  |
|      an     |  DT  |
|   ancient   |  JJ  |
|   artifact  |  NN  |
|      .      |  .   |
|     they    | PRP  |
|     are     | VBP  |
| discovering | VBG  |
|     new     |  JJ  |
|  techniques | NNS  |
|     with    |  IN  |
|    their    | PRP$ |
|    recent   |  JJ  |
|  discovery  |  NN  |
+-------------+------+
```

Lemmatizing

In [26]:
```
from nltk.stem import WordNetLemmatizer
```

In [27]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Uditya\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[27]: True

In [28]:
```
lemmetizer = WordNetLemmatizer()
```

In [29]:
```
words = word_tokenize(text)
```

In [30]:
```
lemma = [lemmetizer.lemmatize(word) for word in words]
```

In [31]:
```
table = PrettyTable(field_names=['word', 'stemmed word', 'lemma word'])
```

```
for word, stm, lm in zip(words, stemmed_words, lemma):
        table.add_row([word, stm, lm])

print(table)
```

```
+-------------+--------------+-------------+
|     word    | stemmed word |  lemma word |
+-------------+--------------+-------------+
|     The     |     the      |     The     |
|  scientists |  scientist   |  scientist  |
|   discover  |    discov    |   discover  |
|     new     |     new      |     new     |
|   species   |    speci     |    specie   |
|    every    |    everi     |    every    |
|     year    |     year     |     year    |
|      ,      |      ,       |      ,      |
|     Last    |     last     |     Last    |
|     year    |     year     |     year    |
|      ,      |      ,       |      ,      |
|     they    |     they     |     they    |
|  discovered |    discov    |  discovered |
|      an     |      an      |      an     |
|   ancient   |   ancient    |   ancient   |
|   artifact  |   artifact   |   artifact  |
|      .      |      .       |      .      |
|     they    |     they     |     they    |
|     are     |     are      |     are     |
| discovering |    discov    | discovering |
|     new     |     new      |     new     |
|  techniques |   techniqu   |  technique  |
|     with    |     with     |     with    |
|    their    |    their     |    their    |
|    recent   |    recent    |    recent   |
|  discovery  |  discoveri   |  discovery  |
+-------------+--------------+-------------+
```

Chunking

In [32]:
```
text = "Docker and Docker Compose have become essential tools for developers a
```

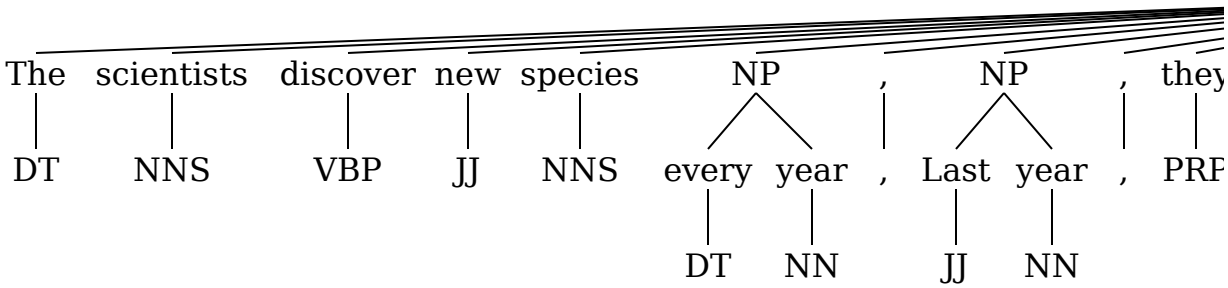
In [33]:
```
from nltk.chunk import RegexpParser
grammar = "NP: {<DT>?<JJ>*<NN>}"
```

In [34]:
```
chunk_parser = RegexpParser(grammar)
chunks = chunk_parser.parse(tagged_words)
```

In [35]:
```
# !pip install svgling
```

In [36]:
```
chunks
```

Out[36]:



In [37]:
```
print(chunks)
```

```
(S
  The/DT
  scientists/NNS
  discover/VBP
  new/JJ
  species/NNS
  (NP every/DT year/NN)
  ,/,
  (NP Last/JJ year/NN)
  ,/,
  they/PRP
  discovered/VBD
  (NP an/DT ancient/JJ artifact/NN)
  ./.
  they/PRP
  are/VBP
  discovering/VBG
  new/JJ
  techniques/NNS
  with/IN
  their/PRP$
  (NP recent/JJ discovery/NN))
```

Named Entity Recognition(NER)

In [38]:
```
from nltk.chunk import ne_chunk
```

In [39]:
```
nltk.download('words')
nltk.download('maxent_ne_chunker_tab')
```

```
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\Uditya\AppData\Roaming\nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package maxent_ne_chunker_tab to
[nltk_data]     C:\Users\Uditya\AppData\Roaming\nltk_data...
[nltk_data]   Package maxent_ne_chunker_tab is already up-to-date!
```

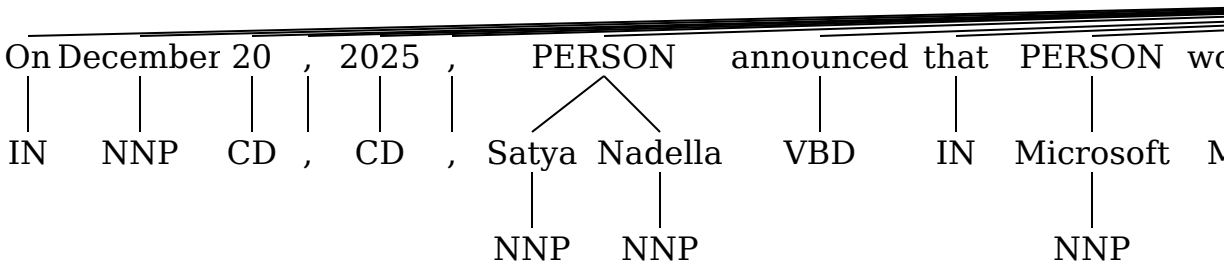
Out[39]: True

In [40]:
```
text = """
On December 20, 2025, Satya Nadella announced that Microsoft would invest $5 b
"""

words = word_tokenize(text)

tagged_words = pos_tag(words)
```

In [41]:
```
ner_tree = ne_chunk(tagged_words)
```

In [42]:
```
ner_tree
```

Out[42]:



In [43]:
```
for ner in ner_tree:
        if hasattr(ner, 'label'):
                print(' '.join(c[0] for c in ner), ":", ner.label())
```

```
Satya Nadella : PERSON
Microsoft : PERSON
London : GPE
Kings Cross : PERSON
Google : ORGANIZATION
Mountain View : GPE
California : GPE
```