21AIE314

AI IN NATURAL LANGUAGE PROCESSING

LAB MANUAL

TABLE OF CONTENTS

NLP LAB 1

AIM:

To explore NLTK for analyzing and processing human language data, utilizing its comprehensive set of tools and resources for tasks such as tokenization, stemming, part-of-speech tagging, and more

In [68]:
```
pip install nltk==3.5
```

```
Requirement already satisfied: nltk==3.5 in c:\users\akile\anaconda3\lib\site-package
s (3.5)
Requirement already satisfied: click in c:\users\akile\anaconda3\lib\site-packages (f
rom nltk==3.5) (8.0.4)
Requirement already satisfied: joblib in c:\users\akile\anaconda3\lib\site-packages
(from nltk==3.5) (1.2.0)
Requirement already satisfied: regex in c:\users\akile\anaconda3\lib\site-packages (f
rom nltk==3.5) (2022.7.9)
Requirement already satisfied: tqdm in c:\users\akile\anaconda3\lib\site-packages (fr
om nltk==3.5) (4.65.0)
Requirement already satisfied: colorama in c:\users\akile\anaconda3\lib\site-packages
(from click->nltk==3.5) (0.4.6)
Note: you may need to restart the kernel to use updated packages.
```

In [69]:
```
pip install numpy matplotlib
```

```
Requirement already satisfied: numpy in c:\users\akile\anaconda3\lib\site-packages
(1.24.3)
Requirement already satisfied: matplotlib in c:\users\akile\anaconda3\lib\site-packag
es (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\akile\anaconda3\lib\site-
packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\akile\anaconda3\lib\site-pack
ages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\akile\anaconda3\lib\site
-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\akile\anaconda3\lib\site
-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\akile\anaconda3\lib\site-p
ackages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\akile\anaconda3\lib\site-pac
kages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\akile\anaconda3\lib
\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\akile\anaconda3\lib\s
ite-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\akile\anaconda3\lib\site-packages
(from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Tokenizing

By tokenizing, you can conveniently split up text by word or by sentence. This will allow you to work with smaller pieces of text that are still relatively

coherent and meaningful even outside of the context of the rest of the text

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

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

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

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

In [3]:
```
example_string = """
... Muad'Dib learned rapidly because his first training was in how to learn.
... And the first lesson of all was the basic trust that he could learn.
... It's shocking to find how many people do not believe they can learn,
... and how many more believe learning to be difficult."""
```

In [4]:
```
sent_tokenize(example_string)
```

Out[4]:
```
["\nMuad'Dib learned rapidly because his first training was in how to learn.",
 'And the first lesson of all was the basic trust that he could learn.',
 "It's shocking to find how many people do not believe they can learn,\nand how many more believe learning to be difficult."]
```

Filtering Stop Words

Stop words are words that you want to ignore, so you filter them out of your text when you're processing it. Very common words like 'in', 'is', and 'an' are often used as stop words since they don't add a lot of meaning to a text in and of themselves.

In [5]:
```
nltk.download("stopwords")
```

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

Out[5]:
```
True
```

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

In [7]:
```
from nltk.tokenize import word_tokenize
```

In [8]:
```
worf_quote = "Sir, I protest. I am not a merry man!"
```

In [9]:
```
words_in_quote = word_tokenize(worf_quote)
```

In [10]:
```
words_in_quote
```

Out[10]:
```
['Sir', ',', 'I', 'protest', '.', 'I', 'am', 'not', 'a', 'merry', 'man', '!']
```

In [11]:
```
stop_words = set(stopwords.words("english"))
```

In [12]:
```
filtered_list = []
```

In [13]:
```
for word in words_in_quote:
...     if word.casefold() not in stop_words:
...         filtered_list.append(word)
```

In [14]:
```
for word in words_in_quote:
...     if word.casefold() not in stop_words:
...         filtered_list.append(word)
```

In [15]:
```
filtered_list
```

Out[15]:
```
['Sir',
 ',',
 'protest',
 '.',
 'merry',
 'man',
 '!',
 'Sir',
 ',',
 'protest',
 '.',
 'merry',
 'man',
 '!']
```

Stemming

Stemming is a text processing task in which you reduce words to their root, which is the core part of a word. For example, the words “helping” and “helper” share the root “help.” Stemming allows you to zero in on the basic meaning of a word rather than all the details of how it’s being used

In [16]:
```
from nltk.stem import PorterStemmer
```

In [17]:
```
from nltk.tokenize import word_tokenize
```

In [18]:
```
stemmer = PorterStemmer()
```

In [19]:
```
string_for_stemming = """
... The crew of the USS Discovery discovered many discoveries.
... Discovering is what explorers do."""
```

In [20]:
```
words = word_tokenize(string_for_stemming)
```

In [21]:
```
words
```

Out[21]:
```
['The',
 'crew',
 'of',
 'the',
 'USS',
 'Discovery',
 'discovered',
 'many',
 'discoveries',
 '.',
 'Discovering',
 'is',
 'what',
 'explorers',
 'do',
 '.']
```

In [22]:
```
stemmed_words = [stemmer.stem(word) for word in words]
```

In [23]:
```
stemmed_words
```

Out[23]:
```
['the',
 'crew',
 'of',
 'the',
 'uss',
 'discoveri',
 'discov',
 'mani',
 'discoveri',
 '.',
 'discov',
 'is',
 'what',
 'explor',
 'do',
 '.']
```

Tagging Parts of Speech

Tagging parts of speech, or POS tagging, is the task of labeling the words in your text according to their part of speech.

In [24]:
```
from nltk.tokenize import word_tokenize
```

In [25]:
```
sagan_quote = """
... If you wish to make an apple pie from scratch,
... you must first invent the universe."""
```

In [26]:
```
words_in_sagan_quote = word_tokenize(sagan_quote)
```

In [27]:
```
import nltk
nltk.pos_tag(words_in_sagan_quote)
```

Out[27]:
```
[('If', 'IN'),
 ('you', 'PRP'),
 ('wish', 'VBP'),
 ('to', 'TO'),
 ('make', 'VB'),
 ('an', 'DT'),
 ('apple', 'NN'),
 ('pie', 'NN'),
 ('from', 'IN'),
 ('scratch', 'NN'),
 (',', ','),
 ('you', 'PRP'),
 ('must', 'MD'),
 ('first', 'VB'),
 ('invent', 'VB'),
 ('the', 'DT'),
 ('universe', 'NN'),
 ('.', '.')]
```

In [28]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
```

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

Out[28]:
```
True
```

In [29]:
```
import nltk
nltk.download('tagsets')
```

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

Out[29]:
```
True
```

In [30]:
```
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 [31]:
```
jabberwocky_excerpt = """
... 'Twas brillig, and the slithy toves did gyre and gimble in the wabe:
... all mimsy were the borogoves, and the mome raths outgrabe."""
```

In [32]:
```
words_in_excerpt = word_tokenize(jabberwocky_excerpt)
```

In [33]:
```
nltk.pos_tag(words_in_excerpt)
```

Out[33]:
```
[("'Twas", 'CD'),
 ('brillig', 'NN'),
 (',', ','),
 ('and', 'CC'),
 ('the', 'DT'),
 ('slithy', 'JJ'),
 ('toves', 'NNS'),
 ('did', 'VBD'),
 ('gyre', 'NN'),
 ('and', 'CC'),
 ('gimble', 'JJ'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('wabe', 'NN'),
 (':', ':'),
 ('all', 'DT'),
 ('mimsy', 'NNS'),
 ('were', 'VBD'),
 ('the', 'DT'),
 ('borogoves', 'NNS'),
 (',', ','),
 ('and', 'CC'),
 ('the', 'DT'),
 ('mome', 'JJ'),
 ('raths', 'NNS'),
 ('outgrabe', 'RB'),
 ('.', '.')]
```

Lemmatizing

Lemmatizing reduces words to their core meaning, but it will give you a complete English word that makes sense on its own instead of just a fragment of a word like 'discoveri'

In [34]:
```
from nltk.stem import WordNetLemmatizer
```

In [35]:
```
lemmatizer = WordNetLemmatizer()
```

In [36]:
```
import nltk
nltk.download('wordnet')
```

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

Out[36]:
```
True
```

In [37]:
```
lemmatizer.lemmatize("scarves")
```

Out[37]:
```
'scarf'
```

In [38]:
```
string_for_lemmatizing = "The friends of DeSoto love scarves."
```

In [39]:
```
words = word_tokenize(string_for_lemmatizing)
```

In [40]:
```
words
```

Out[40]:
```
['The', 'friends', 'of', 'DeSoto', 'love', 'scarves', '.']
```

In [41]:
```
lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
```

In [42]:
```
lemmatized_words
```

Out[42]:
```
['The', 'friend', 'of', 'DeSoto', 'love', 'scarf', '.']
```

In [43]:
```
lemmatizer.lemmatize("worst")
```

Out[43]:
```
'worst'
```

In [44]:
```
lemmatizer.lemmatize("worst", pos="a")
```

Out[44]:
```
'bad'
```

Chunking

chunking allows you to identify phrases

In [45]:
```
from nltk.tokenize import word_tokenize
```

In [46]:
```
lotr_quote = "It's a dangerous business, Frodo, going out your door."
```

In [47]:
```
words_in_lotr_quote = word_tokenize(lotr_quote)
```

In [48]:
```
words_in_lotr_quote
```

Out[48]:
```
['It',
 "'s",
 'a',
 'dangerous',
 'business',
 ',',
 'Frodo',
 ',',
 'going',
 'out',
 'your',
 'door',
 '.']
```

In [49]:
```
nltk.download("averaged_perceptron_tagger")
```

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

Out[49]:
```
True
```

In [50]:
```
lotr_pos_tags = nltk.pos_tag(words_in_lotr_quote)
```

In [51]:
```
lotr_pos_tags
```

Out[51]:
```
[('It', 'PRP'),
 ("'s", 'VBZ'),
 ('a', 'DT'),
 ('dangerous', 'JJ'),
 ('business', 'NN'),
 (',', ','),
 ('Frodo', 'NNP'),
 (',', ','),
 ('going', 'VBG'),
 ('out', 'RP'),
 ('your', 'PRP$'),
 ('door', 'NN'),
 ('.', '.')]
```

In [52]:
```
grammar = "NP: {<DT>?<JJ>*<NN>}"
```

In [53]:
```
chunk_parser = nltk.RegexpParser(grammar)
```

In [54]:
```
tree = chunk_parser.parse(lotr_pos_tags)
```

In [55]:
```
tree.draw()
```

Chinking

Chinking is used together with chunking, but while chunking is used to include a pattern, chinking is used to exclude a pattern

In [56]:
```
lotr_pos_tags
```

Out[56]:
```
[('It', 'PRP'),
 ("'s", 'VBZ'),
 ('a', 'DT'),
 ('dangerous', 'JJ'),
 ('business', 'NN'),
 (',', ','),
 ('Frodo', 'NNP'),
 (',', ','),
 ('going', 'VBG'),
 ('out', 'RP'),
 ('your', 'PRP$'),
 ('door', 'NN'),
 ('.', '.')]
```

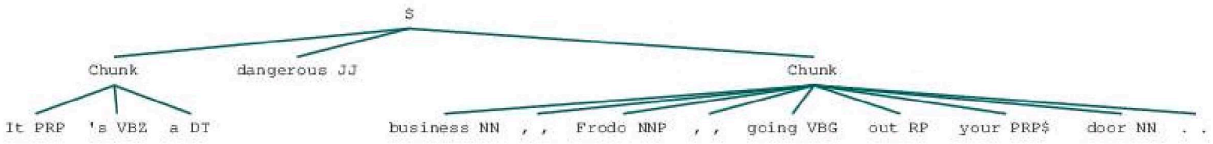
In [57]:
```
grammar = """
... Chunk: {<.*>+}
...        }<JJ>{"""
```

In [58]:
```
chunk_parser = nltk.RegexpParser(grammar)
```

In [59]:
```
tree = chunk_parser.parse(lotr_pos_tags)
```

In [60]:
```
tree
```

Out[60]:



In [61]:
```
tree.draw()
```

Using Named Entity Recognition (NER)

Named entities are noun phrases that refer to specific locations, people, organizations, and so on

In [62]:
```
nltk.download("maxent_ne_chunker")
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker.zip.
```

Out[62]:
```
True
```

In [63]:
```
nltk.download("words")
```

```
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
```

Out[63]:
```
True
```

In [64]:
```
tree = nltk.ne_chunk(lotr_pos_tags)
```

In [65]:
```
tree.draw()
```

In [66]:
```
tree = nltk.ne_chunk(lotr_pos_tags, binary=True)
```

In [67]:
```
tree.draw()
```

In [68]:
```
quote = """
... Men like Schiaparelli watched the red planet—it is odd, by-the-bye, that
... for countless centuries Mars has been the star of war—but failed to
... interpret the fluctuating appearances of the markings they mapped so well.
... All that time the Martians must have been getting ready.
...
... During the opposition of 1894 a great light was seen on the illuminated
... part of the disk, first at the Lick Observatory, then by Perrotin of Nice,
... and then by other observers. English readers heard of it first in the
... issue of Nature dated August 2."""
```

In [71]:
```
def extract_ne(quote):
...     words = word_tokenize(quote, language="English")
...     tags = nltk.pos_tag(words)
...     tree = nltk.ne_chunk(tags, binary=True)
...     return set(
...         " ".join(i[0] for i in t)
...         for t in tree
...         if hasattr(t, "label") and t.label() == "NE"
...     )
```

In [72]:
```
extract_ne(quote)
```

Out[72]:
```
{'Lick Observatory', 'Mars', 'Nature', 'Perrotin', 'Schiaparelli'}
```

Getting Text to Analyze

In [73]:
```
nltk.download("book")
from nltk.book import *
```

```
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\abc.zip.
[nltk_data]    | Downloading package brown to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\brown.zip.
[nltk_data]    | Downloading package chat80 to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\chat80.zip.
[nltk_data]    | Downloading package cmudict to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\cmudict.zip.
[nltk_data]    | Downloading package conll2000 to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\conll2000.zip.
[nltk_data]    | Downloading package conll2002 to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\conll2002.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\dependency_treebank.zip.
[nltk_data]    | Downloading package genesis to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\genesis.zip.
[nltk_data]    | Downloading package gutenberg to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\gutenberg.zip.
[nltk_data]    | Downloading package ieer to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\ieer.zip.
[nltk_data]    | Downloading package inaugural to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\inaugural.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\movie_reviews.zip.
[nltk_data]    | Downloading package nps_chat to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\nps_chat.zip.
[nltk_data]    | Downloading package names to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\names.zip.
[nltk_data]    | Downloading package ppattach to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\ppattach.zip.
[nltk_data]    | Downloading package reuters to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    | Downloading package senseval to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\senseval.zip.
[nltk_data]    | Downloading package state_union to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\state_union.zip.
[nltk_data]    | Downloading package stopwords to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package swadesh to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
```

```
[nltk_data]    |   Unzipping corpora\swadesh.zip.
[nltk_data]    | Downloading package timit to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\timit.zip.
[nltk_data]    | Downloading package treebank to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\treebank.zip.
[nltk_data]    | Downloading package toolbox to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\toolbox.zip.
[nltk_data]    | Downloading package udhr to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\udhr.zip.
[nltk_data]    | Downloading package udhr2 to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\unicode_samples.zip.
[nltk_data]    | Downloading package webtext to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\webtext.zip.
[nltk_data]    | Downloading package wordnet to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\wordnet_ic.zip.
[nltk_data]    | Downloading package words to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping taggers\maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping taggers\universal_tagset.zip.
[nltk_data]    | Downloading package punkt to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping grammars\book_grammars.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Unzipping corpora\city_database.zip.
[nltk_data]    | Downloading package tagsets to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Package tagsets is already up-to-date!
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     C:\Users\akile\AppData\Roaming\nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection book
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Using a Concordance

When you use a concordance, you can see each time a word is used, along with its immediate context

In [74]:
```
text8.concordance("man")
```

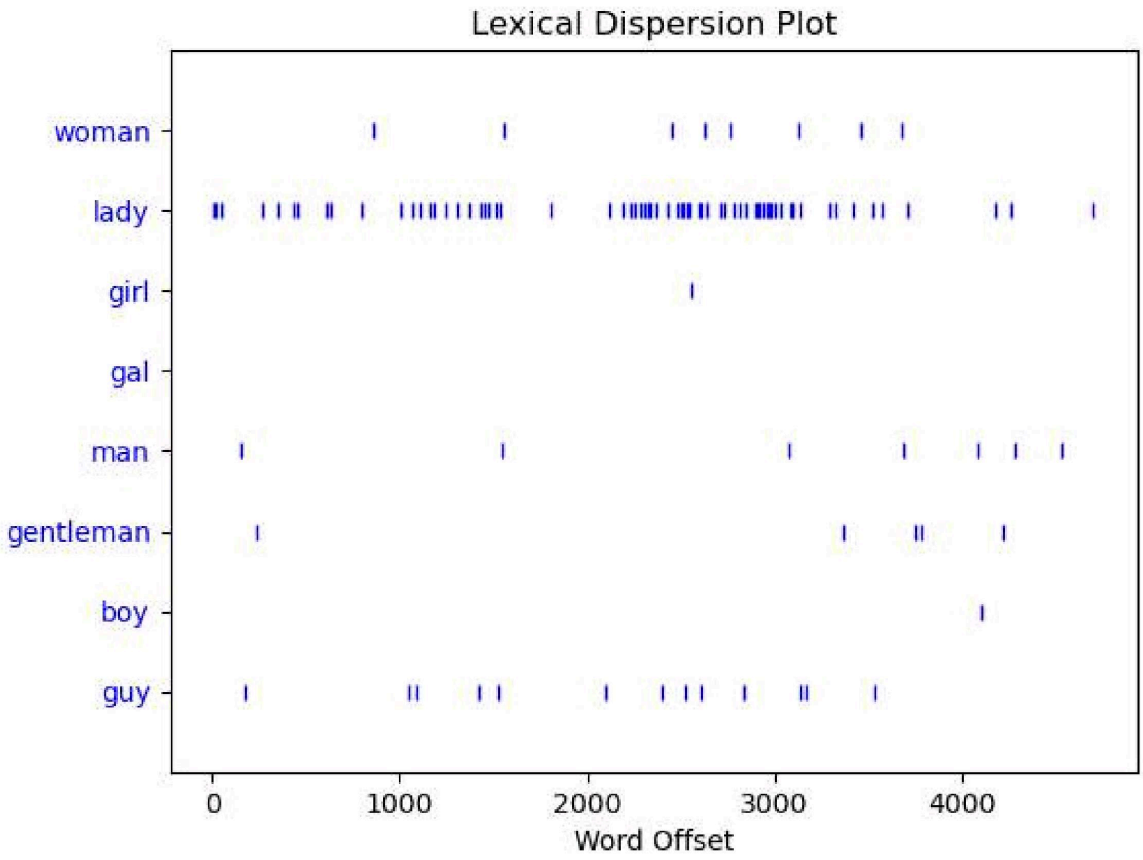
```
Displaying 14 of 14 matches:
 to hearing from you all . ABLE young man seeks , sexy older women . Phone for
ble relationship . GENUINE ATTRACTIVE MAN 40 y . o ., no ties , secure , 5 ft .
ship , and quality times . VIETNAMESE MAN Single , never married , financially
ip . WELL DRESSED emotionally healthy man 37 like to meet full figured woman fo
 nth subs LIKE TO BE MISTRESS of YOUR MAN like to be treated well . Bold DTE no
eeks lady in similar position MARRIED MAN 50 , attrac . fit , seeks lady 40 - 5
eks nice girl 25 - 30 serious rship . Man 46 attractive fit , assertive , and k
 40 - 50 sought by Aussie mid 40s b / man f / ship r / ship LOVE to meet widowe
discreet times . Sth E Subs . MARRIED MAN 42yo 6ft , fit , seeks Lady for discr
woman , seeks professional , employed man , with interests in theatre , dining
 tall and of large build seeks a good man . I am a nonsmoker , social drinker ,
lead to relationship . SEEKING HONEST MAN I am 41 y . o ., 5 ft . 4 , med . bui
 quiet times . Seeks 35 - 45 , honest man with good SOH & similar interests , f
 genuine , caring , honest and normal man for fship , poss rship . S / S , S /
```

In [75]:
```
text8.concordance("woman")
```

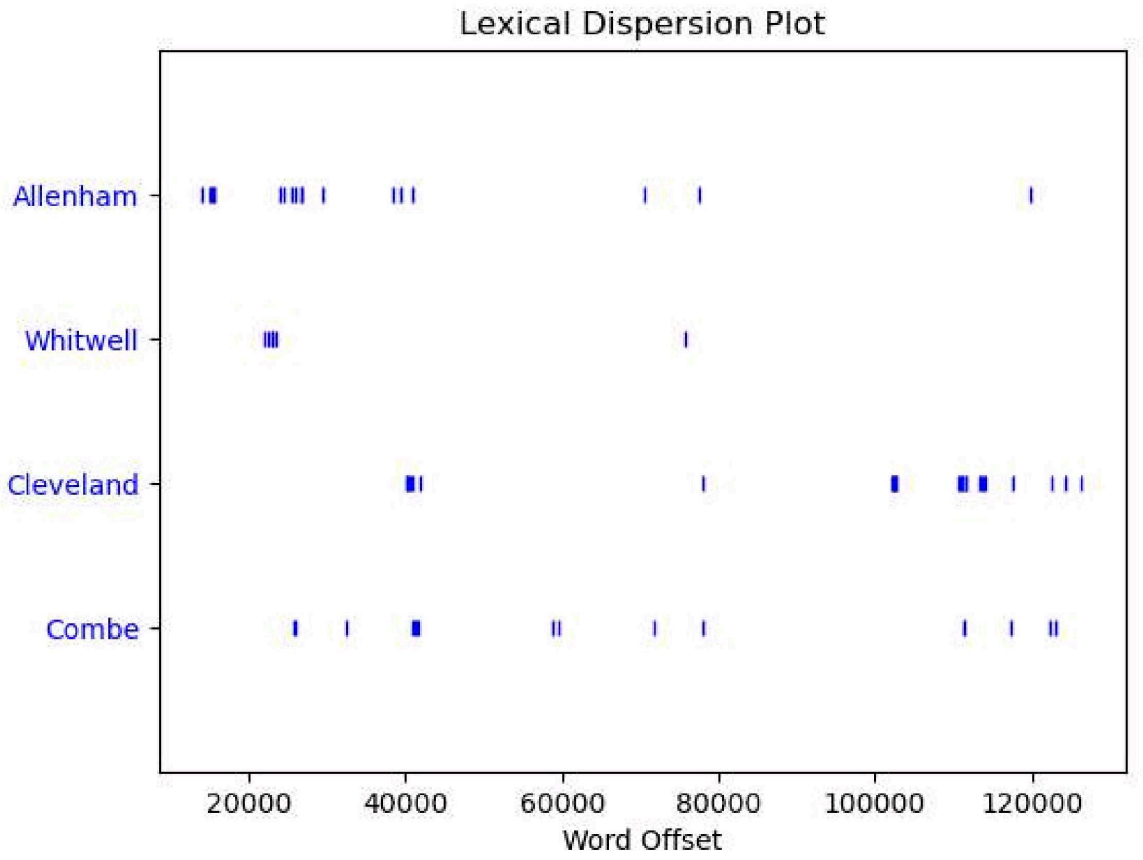
```
Displaying 11 of 11 matches:
at home . Seeking an honest , caring woman , slim or med . build , who enjoys t
thy man 37 like to meet full figured woman for relationship . 48 slim , shy , S
rry . MALE 58 years old . Is there a Woman who would like to spend 1 weekend a
 other interests . Seeking Christian Woman for fship , view to rship . SWM 45 D
ALE 60 - burly beared seeks intimate woman for outings n / s s / d F / ston / P
ington . SCORPIO 47 seeks passionate woman for discreet intimate encounters SEX
le dad . 42 , East sub . 5 " 9 seeks woman 30 + for f / ship relationship TALL
personal trainer looking for married woman age open for fun MARRIED Dark guy 37
rinker , seeking slim - medium build woman who is happy in life , age open . AC
. O . TERTIARY Educated professional woman , seeks professional , employed man
 real romantic , age 50 - 65 y . o . WOMAN OF SUBSTANCE 56 , 59 kg ., 50 , fit
```

Making a Dispersion Plot

In [76]:
```
text8.dispersion_plot(
...     ["woman", "lady", "girl", "gal", "man", "gentleman", "boy", "guy"]
... )
```



In [77]:
```
text2.dispersion_plot(["Allenham", "Whitwell", "Cleveland", "Combe"])
```



Making a Frequency Distribution

In [78]:
```
from nltk import FreqDist
```

In [79]:
```
frequency_distribution = FreqDist(text8)
print(frequency_distribution)
```

```
<FreqDist with 1108 samples and 4867 outcomes>
```

In [80]:
```
frequency_distribution.most_common(20)
```

Out[80]:
```
[(',', 539),
 ('.', 353),
 ('/', 110),
 ('for', 99),
 ('and', 74),
 ('to', 74),
 ('lady', 68),
 ('-', 66),
 ('seeks', 60),
 ('a', 52),
 ('with', 44),
 ('S', 36),
 ('ship', 33),
 ('&', 30),
 ('relationship', 29),
 ('fun', 28),
 ('in', 27),
 ('slim', 27),
 ('build', 27),
 ('o', 26)]
```

In [81]:
```
meaningful_words = [
...     word for word in text8 if word.casefold() not in stop_words
... ]
```

In [82]:
```
frequency_distribution.most_common(20)
```

Out[82]:
```
[(',', 539),
 ('.', 353),
 ('/', 110),
 ('for', 99),
 ('and', 74),
 ('to', 74),
 ('lady', 68),
 ('-', 66),
 ('seeks', 60),
 ('a', 52),
 ('with', 44),
 ('S', 36),
 ('ship', 33),
 ('&', 30),
 ('relationship', 29),
 ('fun', 28),
 ('in', 27),
 ('slim', 27),
 ('build', 27),
 ('o', 26)]
```

Finding Collocations

A collocation is a sequence of words that shows up often

In [83]:
```
text8.collocations()
```

```
would like; medium build; social drinker; quiet nights; non smoker;
long term; age open; Would like; easy going; financially secure; fun
times; similar interests; Age open; weekends away; poss rship; well
presented; never married; single mum; permanent relationship; slim
build
```

In [84]:
```
lemmatized_words = [lemmatizer.lemmatize(word) for word in text8]
```

In [85]:
```
new_text = nltk.Text(lemmatized_words)
```

In [86]:
```
new_text.collocations()
```

```
medium build; social drinker; non smoker; quiet night; long term;
would like; age open; easy going; financially secure; Would like; fun
time; similar interest; Age open; weekend away; well presented; never
married; single mum; permanent relationship; year old; slim build
```

INFERENCE:

In this NLP lab, we harnessed the power of the Natural Language Toolkit (NLTK) to conduct a thorough exploration of fundamental language processing tasks. Beginning with tokenization and the curation of relevant content by filtering out stop words, we seamlessly navigated through stemming, part-of-speech tagging, and lemmatization. The use of chunking and chinking mechanisms provided insights into syntactic structures, while Named Entity Recognition (NER) illuminated entities within the text. The culmination of our analysis involved text exploration through concordance, dispersion plots, frequency distributions, and collocations, revealing nuanced patterns and occurrences. NLTK's versatility emerged prominently,

underlining its pivotal role in unraveling the intricacies of human language for diverse applications.

NLP LAB 2

Introduction to NLP and spaCy

AIM

To explore various Natural Language Processing (NLP) operations and spaCy functionalities

In [1]:
```
pip install spacy
```

```
     ---------------------------------------- 0.0/45.9 kB ? eta -:--:--
     ---------------------------------------- 45.9/45.9 kB ? eta 0:00:00
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in c:\users\akile
\anaconda3\lib\site-packages (from spacy) (5.2.1)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\akile\anaco
nda3\lib\site-packages (from spacy) (4.65.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\akile\a
naconda3\lib\site-packages (from spacy) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
c:\users\akile\anaconda3\lib\site-packages (from spacy) (1.10.8)
Requirement already satisfied: jinja2 in c:\users\akile\anaconda3\lib\site
-packages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in c:\users\akile\anaconda3\lib
\site-packages (from spacy) (68.0.0)
Requirement already satisfied: packaging>=20.0 in c:\users\akile\anaconda3
\lib\site-packages (from spacy) (23.1)
Collecting langcodes<4.0.0,>=3.2.0 (from spacy)
  Downloading langcodes-3.3.0-py3-none-any.whl (181 kB)
     ---------------------------------------- 0.0/181.6 kB ? eta -:--:--
     ------------ ------------------------- 61.4/181.6 kB 3.2 MB/s eta 0:
```

The Doc Object for Processed Text

In [2]:
```
import spacy
nlp = spacy.load("en_core_web_sm")
nlp
```

Out[2]:
```
<spacy.lang.en.English at 0x11c00708d50>
```

In [3]:
```
introduction_doc = nlp("This tutorial is about Natural Language Processing in
type(introduction_doc)
```

Out[3]:
```
spacy.tokens.doc.Doc
```

In [4]:
```
[token.text for token in introduction_doc]
```

Out[4]:
```
['This',
 'tutorial',
 'is',
 'about',
 'Natural',
 'Language',
 'Processing',
 'in',
 'spaCy',
 '.']
```

In [7]:
```
import pathlib
file_name = "Introduction.txt.txt"
introduction_doc = nlp(pathlib.Path(file_name).read_text(encoding="utf-8"))
print ([token.text for token in introduction_doc])
```

```
['HI', 'All', 'I', 'am', 'Akilesh']
```

Sentence Detection

Sentence detection is the process of locating where sentences start and end in a given text.

In [8]:
```
about_text = (
...   "Gus Proto is a Python developer currently"
...     " working for a London-based Fintech"
...     " company. He is interested in learning"
...     " Natural Language Processing."
... )
```

In [9]:
```
about_doc = nlp(about_text)
sentences = list(about_doc.sents)
len(sentences)
```

Out[9]:
```
2
```

In [10]:
```
for sentence in sentences:
...     print(f"{sentence[:5]}...")
```

```
Gus Proto is a Python...
He is interested in learning...
```

In [11]:
```
ellipsis_text = (
...     "Gus, can you, ... never mind, I forgot"
...     " what I was saying. So, do you think"
...     " we should ..."
... )
```

In [14]:
```
from spacy.language import Language
@Language.component("set_custom_boundaries")
def set_custom_boundaries(doc):
    for token in doc[:-1]:
        if token.text == "...":
            doc[token.i + 1].is_sent_start = True
    return doc
```

In [15]:
```
custom_nlp = spacy.load("en_core_web_sm")
```

In [16]:
```
custom_nlp.add_pipe("set_custom_boundaries", before="parser")
```

Out[16]:
```
<function __main__.set_custom_boundaries(doc)>
```

In [17]:
```
custom_ellipsis_doc = custom_nlp(ellipsis_text)
```

In [18]:
```
custom_ellipsis_sentences = list(custom_ellipsis_doc.sents)
```

In [19]:
```
custom_ellipsis_sentences = list(custom_ellipsis_doc.sents)
```

In [20]:
```
for sentence in custom_ellipsis_sentences:
...     print(sentence)
```

```
Gus, can you, ...
never mind, I forgot what I was saying.
So, do you think we should ...
```

Tokens in spaCy

In [22]:
```
import spacy
nlp = spacy.load("en_core_web_sm")
about_text = (
     "Gus Proto is a Python developer currently"
     " working for a London-based Fintech"
     " company. He is interested in learning"
     " Natural Language Processing."
 )
about_doc = nlp(about_text)

for token in about_doc:
    print (token, token.idx)
```

```
Gus 0
Proto 4
is 10
a 13
Python 15
developer 22
currently 32
working 42
for 50
a 54
London 56
- 62
based 63
Fintech 69
company 77
. 84
He 86
is 89
interested 92
in 103
learning 106
Natural 115
Language 123
Processing 132
. 142
```

In [26]:
```
print(
   f'{"Text with Whitespace":22}'
   f'{"Is Alphanumeric?":15}'
   f'{"Is Punctuation?":18}'
   f'{"Is Stop Word?"}'
)
```

```
Text with Whitespace  Is Alphanumeric?Is Punctuation?   Is Stop Word?
```

In [27]:
```
for token in about_doc:
    print(
        f'{str(token.text_with_ws):22}'
        f'{str(token.is_alpha):15}'
        f'{str(token.is_punct):18}'
        f'{str(token.is_stop)}'
    )
```

```
Gus                    True           False          False
Proto                  True           False          False
is                     True           False          True
a                      True           False          True
Python                 True           False          False
developer              True           False          False
currently              True           False          False
working                True           False          False
for                    True           False          True
a                      True           False          True
London                 True           False          False
-                      False          True           False
based                  True           False          False
Fintech                True           False          False
company                True           False          False
.                      False          True           False
He                     True           False          True
is                     True           False          True
interested             True           False          False
in                     True           False          True
learning               True           False          False
Natural                True           False          False
Language               True           False          False
Processing             True           False          False
.                      False          True           False
```

In [28]:
```
custom_about_text = (
    "Gus Proto is a Python developer currently"
    " working for a London@based Fintech"
    " company. He is interested in learning"
    " Natural Language Processing."
)


tokens_to_print = [token.text for token in about_doc[8:15]]
print(tokens_to_print)
```

```
['for', 'a', 'London', '-', 'based', 'Fintech', 'company']
```

In [29]:
```
import re
import spacy
from spacy.tokenizer import Tokenizer

custom_nlp = spacy.load("en_core_web_sm")

prefix_re = spacy.util.compile_prefix_regex(custom_nlp.Defaults.prefixes)
suffix_re = spacy.util.compile_suffix_regex(custom_nlp.Defaults.suffixes)

custom_infixes = [r"@"]
infix_re = spacy.util.compile_infix_regex(
    list(custom_nlp.Defaults.infixes) + custom_infixes
)

custom_nlp.tokenizer = Tokenizer(
    custom_nlp.vocab,
    prefix_search=prefix_re.search,
    suffix_search=suffix_re.search,
    infix_finditer=infix_re.finditer,
    token_match=None,
)

custom_about_text = (
    "Gus Proto is a Python developer currently"
    " working for a London@based Fintech"
    " company. He is interested in learning"
    " Natural Language Processing."
)

custom_tokenizer_about_doc = custom_nlp(custom_about_text)

print([token.text for token in custom_tokenizer_about_doc[8:15]])
```

```
['for', 'a', 'London', '@', 'based', 'Fintech', 'company']
```

Stop Words

In [30]:
```
import spacy

spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS
len(spacy_stopwords)

for stop_word in list(spacy_stopwords)[:10]:
    print(stop_word)
```

```
next
your
less
above
although
who
becomes
within
this
does
```

In [31]:
```
import spacy

custom_about_text = (
    "Gus Proto is a Python developer currently"
    " working for a London-based Fintech"
    " company. He is interested in learning"
    " Natural Language Processing."
)

nlp = spacy.load("en_core_web_sm")
about_doc = nlp(custom_about_text)

filtered_tokens = [token.text for token in about_doc if not token.is_stop]

print(filtered_tokens)
```

```
['Gus', 'Proto', 'Python', 'developer', 'currently', 'working', 'London', '-
', 'based', 'Fintech', 'company', '.', 'interested', 'learning', 'Natural',
'Language', 'Processing', '.']
```

Lemmatization

In [32]:
```
import spacy

nlp = spacy.load("en_core_web_sm")
conference_help_text = (
    "Gus is helping organize a developer"
    " conference on Applications of Natural Language"
    " Processing. He keeps organizing local Python meetups"
    " and several internal talks at his workplace."
)

conference_help_doc = nlp(conference_help_text)

for token in conference_help_doc:
    if str(token) != str(token.lemma_):
        print(f"{str(token):>20} : {str(token.lemma_)}")
```

```
                  is : be
                  He : he
               keeps : keep
           organizing : organize
             meetups : meetup
               talks : talk
```

Word Frequency

In [33]:
```
import spacy
from collections import Counter

nlp = spacy.load("en_core_web_sm")

complete_text = (
    "Gus Proto is a Python developer currently"
    " working for a London-based Fintech company. He is"
    " interested in learning Natural Language Processing."
    " There is a developer conference happening on 21 July"
    ' 2019 in London. It is titled "Applications of Natural'
    ' Language Processing". There is a helpline number'
    " available at +44-1234567891. Gus is helping organize it."
    " He keeps organizing local Python meetups and several"
    " internal talks at his workplace. Gus is also presenting"
    ' a talk. The talk will introduce the reader about "Use'
    ' cases of Natural Language Processing in Fintech".'
    " Apart from his work, he is very passionate about music."
    " Gus is learning to play the Piano. He has enrolled"
    " himself in the weekend batch of Great Piano Academy."
    " Great Piano Academy is situated in Mayfair or the City"
    " of London and has world-class piano instructors."
)

complete_doc = nlp(complete_text)

words = [
    token.text
    for token in complete_doc
    if not token.is_stop and not token.is_punct
]

print(Counter(words).most_common(5))
```

```
[('Gus', 4), ('London', 3), ('Natural', 3), ('Language', 3), ('Processing', 3)]
```

In [34]:
```
Counter(
    [token.text for token in complete_doc if not token.is_punct]
 ).most_common(5)
```

Out[34]:
```
[('is', 10), ('a', 5), ('in', 5), ('Gus', 4), ('of', 4)]
```

Part-of-Speech Tagging

In [35]:
```
import spacy

nlp = spacy.load("en_core_web_sm")

about_text = (
    "Gus Proto is a Python developer currently"
    " working for a London-based Fintech"
    " company. He is interested in learning"
    " Natural Language Processing."
)

about_doc = nlp(about_text)

for token in about_doc:
    print(
        f"""
TOKEN: {str(token)}
=====
TAG: {str(token.tag_):10} POS: {token.pos_}
EXPLANATION: {spacy.explain(token.tag_)}"""
    )
```

```
TOKEN: Natural
=====
TAG: NNP        POS: PROPN
EXPLANATION: noun, proper singular

TOKEN: Language
=====
TAG: NNP        POS: PROPN
EXPLANATION: noun, proper singular

TOKEN: Processing
=====
TAG: NNP        POS: PROPN
EXPLANATION: noun, proper singular

TOKEN: .
=====
TAG: .          POS: PUNCT
EXPLANATION: punctuation mark, sentence closer
```

In [36]:
```
nouns = []
adjectives = []

for token in about_doc:
    if token.pos_ == "NOUN":
        nouns.append(token)
    if token.pos_ == "ADJ":
        adjectives.append(token)
```

In [37]:
```
nouns
```

Out[37]:
```
[developer, company]
```

In [38]:
```
adjectives
```

Out[38]:
```
[interested]
```

Visualization: Using displaCy

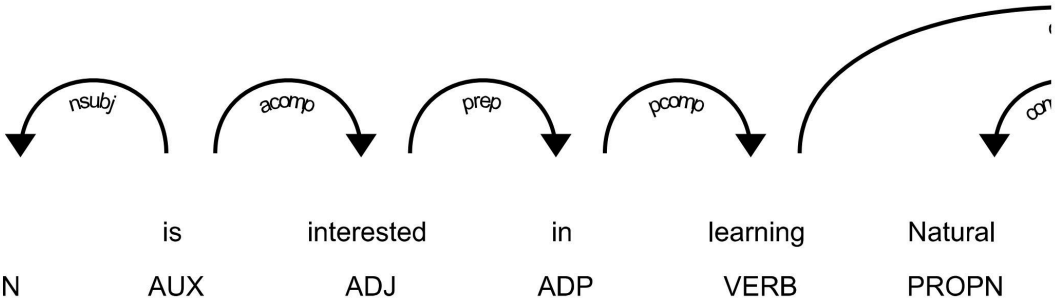
In [2]:
```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")

about_interest_text = "He is interested in learning Natural Language Processin
about_interest_doc = nlp(about_interest_text)

# Generate HTML code for the dependency parse tree
html_code = displacy.render(about_interest_doc, style="dep", options={"distanc

# Print or save the HTML code as needed
print(html_code)
```



None

Preprocessing Functions

In [40]:
```
import spacy

nlp = spacy.load("en_core_web_sm")

complete_text = (
    "Gus Proto is a Python developer currently"
    " working for a London-based Fintech company. He is"
    " interested in learning Natural Language Processing."
    " There is a developer conference happening on 21 July"
    ' 2019 in London. It is titled "Applications of Natural'
    ' Language Processing". There is a helpline number'
    " available at +44-1234567891. Gus is helping organize it."
    " He keeps organizing local Python meetups and several"
    " internal talks at his workplace. Gus is also presenting"
    ' a talk. The talk will introduce the reader about "Use'
    ' cases of Natural Language Processing in Fintech".'
    " Apart from his work, he is very passionate about music."
    " Gus is learning to play the Piano. He has enrolled"
    " himself in the weekend batch of Great Piano Academy."
    " Great Piano Academy is situated in Mayfair or the City"
    " of London and has world-class piano instructors."
)

complete_doc = nlp(complete_text)

def is_token_allowed(token):
    return bool(
        token
        and str(token).strip()
        and not token.is_stop
        and not token.is_punct
    )

def preprocess_token(token):
    return token.lemma_.strip().lower()

complete_filtered_tokens = [
    preprocess_token(token)
    for token in complete_doc
    if is_token_allowed(token)
]

print(complete_filtered_tokens)
```

```
['gus', 'proto', 'python', 'developer', 'currently', 'work', 'london', 'bas
e', 'fintech', 'company', 'interested', 'learn', 'natural', 'language', 'proc
essing', 'developer', 'conference', 'happen', '21', 'july', '2019', 'london',
'title', 'application', 'natural', 'language', 'processing', 'helpline', 'num
ber', 'available', '+44', '1234567891', 'gus', 'helping', 'organize', 'keep',
'organize', 'local', 'python', 'meetup', 'internal', 'talk', 'workplace', 'gu
s', 'present', 'talk', 'talk', 'introduce', 'reader', 'use', 'case', 'natura
l', 'language', 'processing', 'fintech', 'apart', 'work', 'passionate', 'musi
c', 'gus', 'learn', 'play', 'piano', 'enrol', 'weekend', 'batch', 'great', 'p
iano', 'academy', 'great', 'piano', 'academy', 'situate', 'mayfair', 'city',
'london', 'world', 'class', 'piano', 'instructor']
```

Rule-Based Matching Using spaCy

Rule-based matching is one of the steps in extracting information from unstructured text. It's used to identify and extract tokens and phrases according to patterns (such as lowercase) and grammatical features (such as part of speech).

In [41]:
```
import spacy

nlp = spacy.load("en_core_web_sm")

about_text = (
    "Gus Proto is a Python developer currently"
    " working for a London-based Fintech"
    " company. He is interested in learning"
    " Natural Language Processing."
)

about_doc = nlp(about_text)

from spacy.matcher import Matcher

matcher = Matcher(nlp.vocab)

def extract_full_name(nlp_doc):
    pattern = [{"POS": "PROPN"}, {"POS": "PROPN"}]
    matcher.add("FULL_NAME", [pattern])
    matches = matcher(nlp_doc)
    for _, start, end in matches:
        span = nlp_doc[start:end]
        yield span.text

# Print the extracted full name
print(next(extract_full_name(about_doc)))
```

```
Gus Proto
```

Dependency Parsing Using spaCy

Dependency parsing is the process of extracting the dependency graph of a sentence to