In [1]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from IPython.display import display
from pprint import pprint
```

In [2]:
```
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, SnowballStemmer, WordNetLemmatizer
from nltk.book import *
```

```
*** 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
```

Natural Language Processing With Python's NLTK Package

Natural Language Processing With Python's NLTK Package

Tokenizing

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.
"""

example_string = example_string.strip("\n").replace("\n", " ")
example_string
```

Out[3]:
```
"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 belie ve learning to be difficult."
```

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

Out[4]:
```
["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 [5]:
```
print(word_tokenize(example_string))
```

```
["Muad'Dib", 'learned', 'rapidly', 'because', 'his', 'first', 'training', 'wa
s', 'in', 'how', 'to', 'learn', '.', 'And', 'the', 'first', 'lesson', 'of', 'a
ll', 'was', 'the', 'basic', 'trust', 'that', 'he', 'could', 'learn', '.', 'I
t', "'s", 'shocking', 'to', 'find', 'how', 'many', 'people', 'do', 'not', 'bel
ieve', 'they', 'can', 'learn', ',', 'and', 'how', 'many', 'more', 'believe',
'learning', 'to', 'be', 'difficult', '.']
```

Filtering Stop Words

In [6]:
```
stop_words = set(stopwords.words("english"))
```

In [7]:
```
print(stop_words)
```

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

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

In [9]:
```
filtered_list = [word for word in words_in_quote if not word.casefold() in stop
filtered_list
```

Out[9]:
```
['Sir', ',', 'protest', '.', 'merry', 'man', '!']
```

Stemming

PorterStemmer

In [10]:
```
stemmer = PorterStemmer()
```

In [11]:
```
string_for_stemming = """
The crew of the USS Discovery discovered many discoveries.
Discovering is what explorers do.
"""

string_for_stemming = string_for_stemming.strip("\n").replace("\n", " ")
string_for_stemming
```

Out[11]:
```
'The crew of the USS Discovery discovered many discoveries. Discovering is wha
t explorers do.'
```

In [12]:
```
words = word_tokenize(string_for_stemming)
print(words)
```

```
['The', 'crew', 'of', 'the', 'USS', 'Discovery', 'discovered', 'many', 'discov
eries', '.', 'Discovering', 'is', 'what', 'explorers', 'do', '.']
```

In [13]:
```
stemmed_words = [stemmer.stem(word) for word in words]
print(stemmed_words)
```

```
['the', 'crew', 'of', 'the', 'uss', 'discoveri', 'discov', 'mani', 'discover
i', '.', 'discov', 'is', 'what', 'explor', 'do', '.']
```

SnowballStemmer

In [14]:
```
stemmer = SnowballStemmer("english")
```

In [15]:
```
stemmed_words = [stemmer.stem(word) for word in words]
print(stemmed_words)
```

```
['the', 'crew', 'of', 'the', 'uss', 'discoveri', 'discov', 'mani', 'discover
i', '.', 'discov', 'is', 'what', 'explor', 'do', '.']
```

Tagging

In [16]:
```
sagan_quote = """
If you wish to make an apple pie from scratch,
you must first invent the universe.
"""

sagan_quote = sagan_quote.strip("\n").replace("\n", " ")
sagan_quote
```

Out[16]:
```
'If you wish to make an apple pie from scratch, you must first invent the univ
erse.'
```

In [17]:
```
words_in_sagan_quote = word_tokenize(sagan_quote)
print(words_in_sagan_quote)
```

```
['If', 'you', 'wish', 'to', 'make', 'an', 'apple', 'pie', 'from', 'scratch',
',', 'you', 'must', 'first', 'invent', 'the', 'universe', '.']
```

In [18]:
```
nltk.pos_tag(words_in_sagan_quote)
```

Out[18]:
```
[('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 [19]:
```
# [lemmatizer.lemmatize(word, pos) for word, pos in nltk.pos_tag(words_in_sagar
```

In [20]:
```
nltk.pos_tag(words_in_sagan_quote)
```

Out[20]:
```
[('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 [21]:
```
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 [22]:
```
>>> 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.
"""

jabberwocky_excerpt = jabberwocky_excerpt.strip("\n").replace("\n", " ")
jabberwocky_excerpt
```

Out[22]:
```
"'Twas brillig, and the slithy toves did gyre and gimble in the wabe: all mims
y were the borogoves, and the mome raths outgrabe."
```

In [23]:
```
words_in_excerpt = word_tokenize(jabberwocky_excerpt)
```

In [24]:
```
nltk.pos_tag(words_in_excerpt)
```

Out[24]:
```
[("'T", 'NN'),
 ('was', 'VBD'),
 ('brillig', 'VBN'),
 (',', ','),
 ('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

Note: A *lemma* is a word that represents a whole group of words, and that group of words is called a lexeme. For example, if you were to look up the word **“blending”** in a dictionary, then you’d need to look at the entry for **“blend,”** but you would find “blending” listed in that entry. In this example, “blend” is the *lemma*, and “blending” is part of the *lexeme*. So when you *lemmatize* a word, you are reducing it to its *lemma*.

In [25]:
```
lemmatizer = WordNetLemmatizer()
```

In [26]:
```
lemmatizer.lemmatize("scarves")
```

Out[26]:
```
'scarf'
```

In [27]:
```
string_for_lemmatizing = "The friends of DeSoto love scarves."
```

In [28]:
```
words = word_tokenize(string_for_lemmatizing)
words
```

Out[28]:
```
['The', 'friends', 'of', 'DeSoto', 'love', 'scarves', '.']
```

In [29]:
```
lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
lemmatized_words
```

Out[29]:
```
['The', 'friend', 'of', 'DeSoto', 'love', 'scarf', '.']
```

In [30]:
```
lemmatizer.lemmatize("worst")
lemmatizer.lemmatize("worst", pos="a")
```

Out[30]:
```
'worst'
```

Out[30]:
```
'bad'
```

Chunking

In [31]:
```
lotr_quote = "It's a dangerous business, Frodo, going out your door."
```

In [32]:
```
words_in_lotr_quote = word_tokenize(lotr_quote)
print(words_in_lotr_quote)
```

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

In [33]:
```
lotr_pos_tags = nltk.pos_tag(words_in_lotr_quote)
print(lotr_pos_tags)
```

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

In [34]:
```
grammar = "NP: {<DT>?<JJ>*<NN>}"
```

In [35]:
```
chunk_parser = nltk.RegexpParser(grammar)
```

In [36]:
```
tree = chunk_parser.parse(lotr_pos_tags)
display(tree)
```

Chinking

In [37]:
```
grammar = r"""
Chunk: {<.*>+}
}<JJ>{
"""
```

In [38]:
```
chunk_parser = nltk.RegexpParser(grammar)
```

In [39]:
```
tree = chunk_parser.parse(lotr_pos_tags)
display(tree)
```

Using Named Entity Recognition (NER)

In [40]:
```
tree = nltk.ne_chunk(lotr_pos_tags)
display(tree)
```

In [41]:
```
tree = nltk.ne_chunk(lotr_pos_tags, binary=True)
display(tree)
```

In [42]:
```
>>> 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
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.
"""

quote = quote.strip("\n").replace("\n", " ")
quote
```

Out[42]:
```
'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 All that time
the Martians must have been getting ready.  During the opposition of 1894 a gr
eat light was seen on the illuminated part of the disk, first at the Lick Obse
rvatory, then by Perrotin of Nice, and then by other observers. English reader
s heard of it first in the issue of Nature dated August 2.'
```

In [43]:
```
def extract_ne(quote, language="english"):
    words = word_tokenize(quote, language)
    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") a
```

In [44]:
```
extract_ne(quote)
```

Out[44]:
```
{'Lick Observatory', 'Mars', 'Nature', 'Perrotin', 'Schiaparelli'}
```

Using a Concordance

In [45]:
```
text8.concordance("man")
```

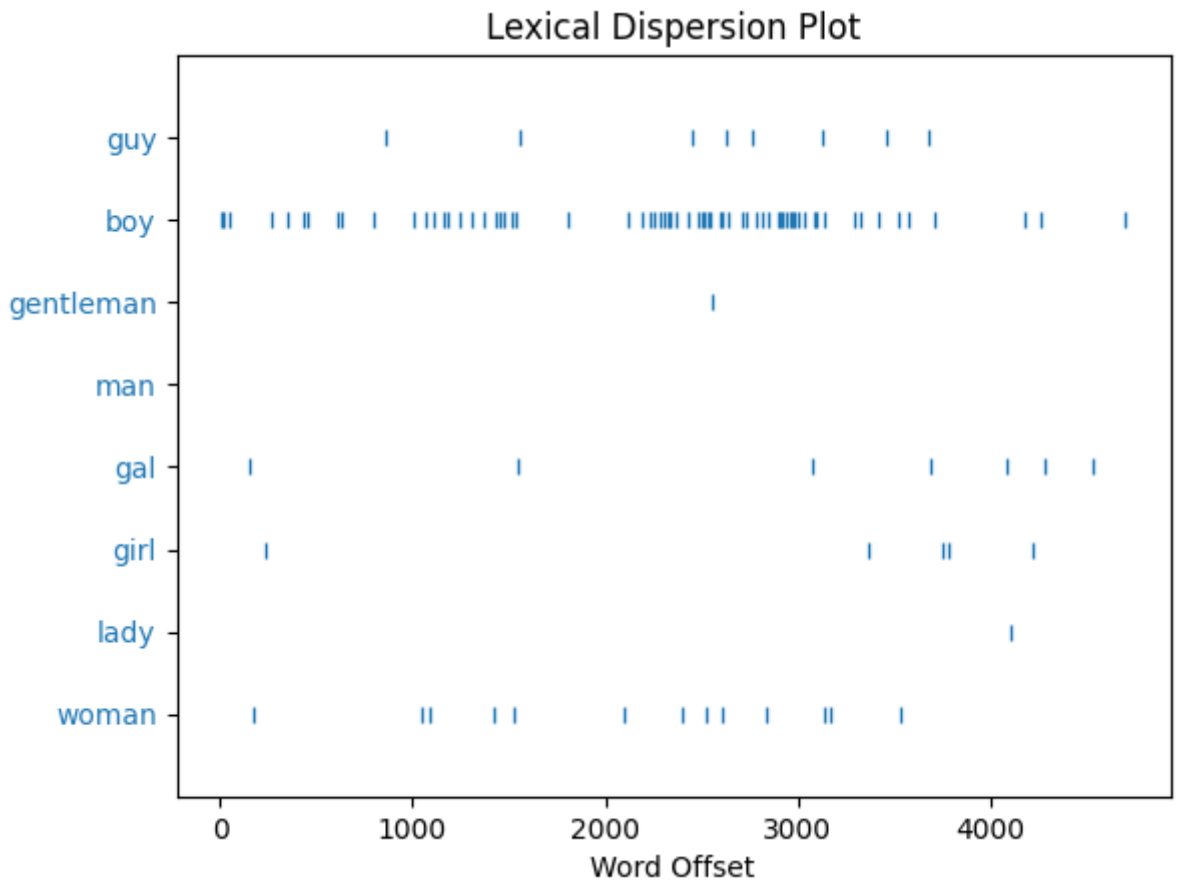
```
Displaying 14 of 14 matches:
 to hearing from you all . ABLE young man seeks , sexy older women . Phone for
ble relationship . GENUINE ATTRACTIVE MAN 40 y . o ., no ties , secure , 5 ft
.
ship , and quality times . VIETNAMESE MAN Single , never married , financially
ip . WELL DRESSED emotionally healthy man 37 like to meet full figured woman f
o
 nth subs LIKE TO BE MISTRESS of YOUR MAN like to be treated well . Bold DTE n
o
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 widow
e
discreet times . Sth E Subs . MARRIED MAN 42yo 6ft , fit , seeks Lady for disc
r
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 . bu
i
 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 [46]:
```
text8.concordance("woman")
```

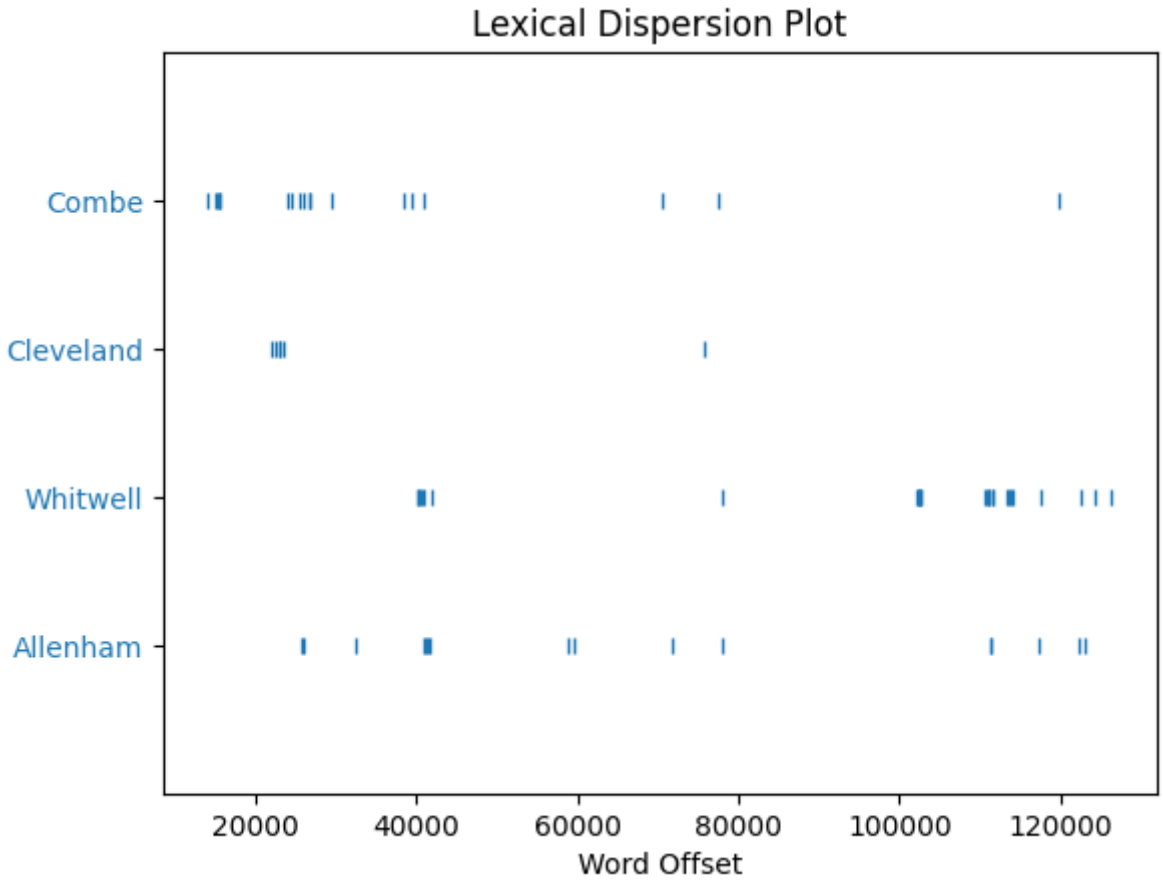
```
Displaying 11 of 11 matches:
at home . Seeking an honest , caring woman , slim or med . build , who enjoys
t
thy man 37 like to meet full figured woman for relationship . 48 slim , shy ,
S
rry . MALE 58 years old . Is there a Woman who would like to spend 1 weekend a
 other interests . Seeking Christian 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 SE
X
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 3
7
rinker , seeking slim - medium build woman who is happy in life , age open . A
C
. 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 [47]:
```
text8.dispersion_plot(["woman", "lady", "girl", "gal", "man", "gentleman", "boy
```



In [48]:
```
# Sense and Sensibility
text2.dispersion_plot(["Allenham", "Whitwell", "Cleveland", "Combe"])
```



Making a Frequency Distribution

In [49]:
```
frequency_distribution = nltk.FreqDist(text8)
print(frequency_distribution)
```

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

In [50]:
```
frequency_distribution.most_common(20)
```

Out[50]:
```
[(',', 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 [51]:
```
meaningful_words = [word for word in text8 if word.isalpha() and not word.casef
```

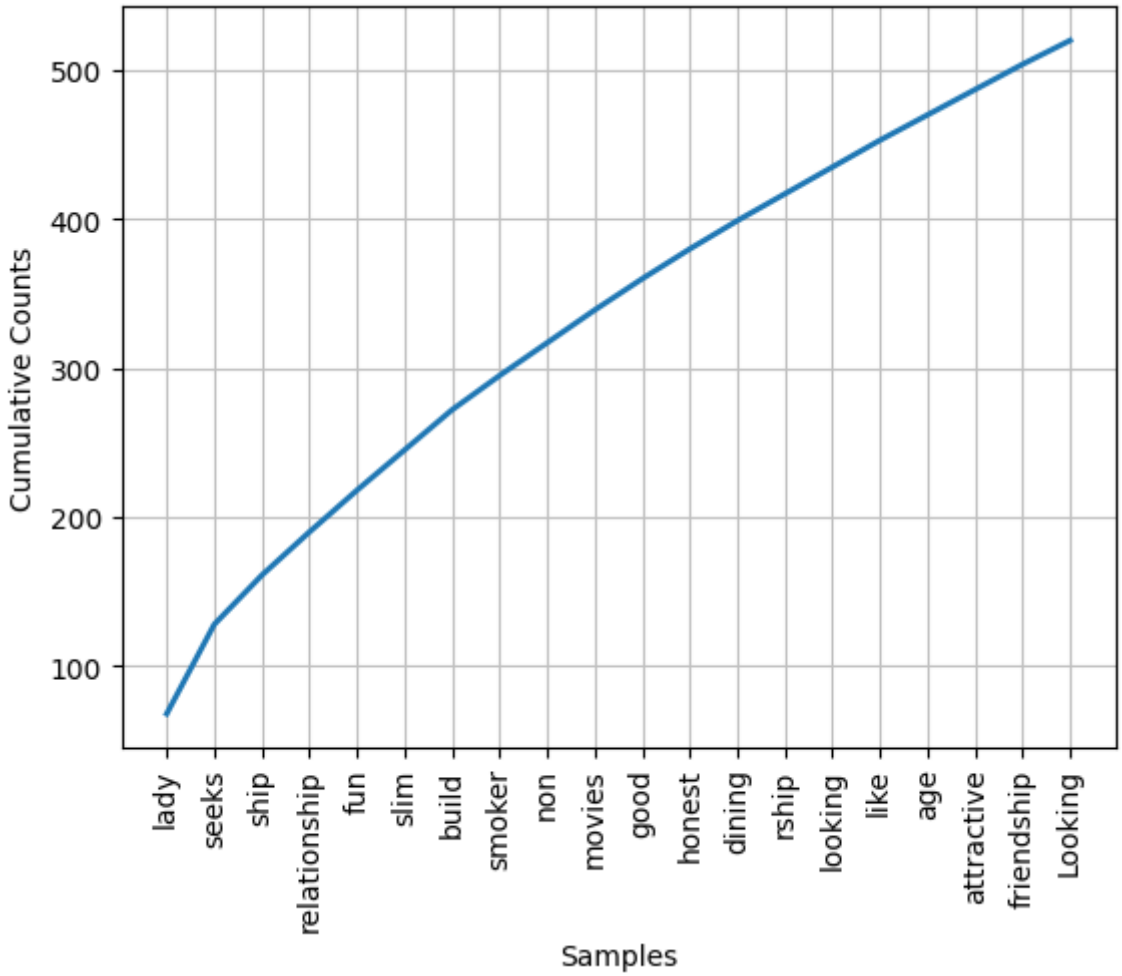
In [52]:
```
frequency_distribution = nltk.FreqDist(meaningful_words)
print(frequency_distribution)
```

```
<FreqDist with 885 samples and 2548 outcomes>
```

In [53]:
```
frequency_distribution.most_common(20)
```

Out[53]:
```
[('lady', 68),
 ('seeks', 60),
 ('ship', 33),
 ('relationship', 29),
 ('fun', 28),
 ('slim', 27),
 ('build', 27),
 ('smoker', 23),
 ('non', 22),
 ('movies', 22),
 ('good', 21),
 ('honest', 20),
 ('dining', 19),
 ('rship', 18),
 ('looking', 18),
 ('like', 18),
 ('age', 17),
 ('attractive', 17),
 ('friendship', 17),
 ('Looking', 16)]
```

In [54]:
```
frequency_distribution.plot(20, cumulative=True)
```



Out[54]: <Axes: xlabel='Samples', ylabel='Cumulative Counts'>

Finding Collocations

In [55]:
```
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 [56]:
```
lemmatizer = WordNetLemmatizer()
```

In [57]:
```
lemmatized_words = [lemmatizer.lemmatize(word) for word in text8]
```

In [58]:
```
new_text = nltk.Text(lemmatized_words)
```

In [59]:
```
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
```

In [ ]:

In [ ]:

Sentiment Analysis: First Steps With Python's NLTK Library

Sentiment Analysis: First Steps With Python's NLTK Library

In [60]:
```
stop_words = stopwords.words("english")
```

In [61]:
```
words = [w for w in nltk.corpus.state_union.words() if w.isalpha() and not w in
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: