Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Basic Text Analytics using NLTK

Name: Chong Mun Chen

IC Number: 960327-07-5097

Date : 5/7/2023

Introduction : Learning about text analytics using NLTK, that is short for Natural Language Toolkit.

Conclusion : I now know that I could separate words and punctuations using NLTK, as well know identifying their types.

NLTK - Basic Text Analytics

- Word & Sentence tokenizer
- Parts of Speech (POS) tagger
- Extracting Entities

Installation

conda install -c conda-forge nltk

In [1]:
```
import nltk

nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ACER\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ACER\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[1]: True

Text Analytics for Beginners using NLTK

Learn How to analyze text using NLTK. Analyze people's sentiments and classify movie reviews.

In today's area of internet and online services, data is generating at incredible speed and amount. Generally, Data analyst, engineer, and scientists are handling relational or tabular data. These tabular data columns have either numerical or categorical data. Generated data has a variety of structures such as text, image, audio, and video. Online activities such as articles, website text, blog posts, social media posts are generating unstructured textual data. Corporate and business need to analyze textual data to understand customer activities, opinion, and feedback to successfully derive their business. To compete with big textual data, text analytics is evolving at a faster rate than ever before.

Text Analytics has lots of applications in today's online world. By analyzing tweets on Twitter, we can find trending news and peoples reaction on a particular event. Amazon can understand user feedback or review on the specific product. BookMyShow can discover people's opinion about the movie. Youtube can also analyze and understand peoples viewpoints on a video.

In this tutorial, you are going to cover the following topics:

Text Analytics and NLP Compare Text Analytics, NLP and Text Mining Text Analysis Operations using NLTK Tokenization Stopwords Lexicon Normalization such as Stemming and Lemmatization POS Tagging Sentiment Analysis Text Classification Performing Sentiment Analysis using Text Classification

Text Analytics and NLP

Text communication is one of the most popular forms of day to day conversion. We chat, message, tweet, share status, email, write blogs, share opinion and feedback in our daily routine. All of these activities are generating text in a significant amount, which is unstructured in nature. I this area of the online marketplace and social media, It is essential to analyze vast quantities of data, to understand peoples opinion.

NLP enables the computer to interact with humans in a natural manner. It helps the computer to understand the human language and derive meaning from it. NLP is applicable in several problematic from speech recognition, language translation, classifying documents to information

extraction. Analyzing movie review is one of the classic examples to demonstrate a simple NLP Bag-of-words model. on movie reviews.

Compare Text Analytics, NLP and Text Mining

Text mining also referred to as text analytics. Text mining is a process of exploring sizeable textual data and find patterns. Text Mining process the text itself, while NLP process with the underlying metadata. Finding frequency counts of words, length of the sentence, presence/absence of specific words is known as text mining. Natural language processing is one of the components of text mining. NLP helps identified sentiment, finding entities in the sentence, and category of blog/article. Text mining is preprocessed data for text analytics. In Text Analytics, statistical and machine learning algorithm used to classify information.

Text Analysis Operations using NLTK

NLTK is a powerful Python package that provides a set of diverse natural languages algorithms. It is free, opensource, easy to use, large community, and well documented. NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition. NLTK helps the computer to analysis, preprocess, and understand the written text.

Tokenization

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

Sentence Tokenization

Sentence tokenizer breaks text paragraph into sentences.

In [2]:
```
import nltk

from nltk.tokenize import sent_tokenize, word_tokenize

# conda install -c anaconda nltk

url = 'https://www.timeout.com/kuala-lumpur/food-and-drink/food-reviews'

text = '''Table & Apron – formerly The Kitchen Table Restaurant & Bakery –
'''

sentences = nltk.sent_tokenize(text)

print(sentences)
```

```
['Table & Apron – formerly The Kitchen Table Restaurant & Bakery – does
n’t exist to disrupt the scene.', 'From the outside, it barely stretches
the boundaries of what is an already saturated restaurant-cum-bakery scen
e.', 'But none of it matters.', 'Because right from its birth in 2014, Ta
ble & Apron has proven to be a restaurant that has in spades a component
so elementary yet so rare – heart.', 'Through hard work, dedication and a
ll the boring old-fashioned virtues of an honest operation, owner Marcus
Low and his team have carved for us a little treasure in Damansara Kim.',
'(Credit must also be given to former co-owner Mei Wan Tan.)', 'The narci
ssism you’ll find in so many KL']
```

Here, the given text is tokenized into sentences.

Word Tokenization

Word tokenizer breaks text paragraph into words.

In [3]:
```
words = word_tokenize(text)

print(words)
```

```
['Table', '&', 'Apron', '–', 'formerly', 'The', 'Kitchen', 'Table', 'Rest
aurant', '&', 'Bakery', '–', 'doesn', '’', 't', 'exist', 'to', 'disrupt',
'the', 'scene', '.', 'From', 'the', 'outside', ',', 'it', 'barely', 'stre
tches', 'the', 'boundaries', 'of', 'what', 'is', 'an', 'already', 'satura
ted', 'restaurant-cum-bakery', 'scene', '.', 'But', 'none', 'of', 'it',
'matters', '.', 'Because', 'right', 'from', 'its', 'birth', 'in', '2014',
',', 'Table', '&', 'Apron', 'has', 'proven', 'to', 'be', 'a', 'restauran
t', 'that', 'has', 'in', 'spades', 'a', 'component', 'so', 'elementary',
'yet', 'so', 'rare', '–', 'heart', '.', 'Through', 'hard', 'work', ',',
'dedication', 'and', 'all', 'the', 'boring', 'old-fashioned', 'virtues',
'of', 'an', 'honest', 'operation', ',', 'owner', 'Marcus', 'Low', 'and',
'his', 'team', 'have', 'carved', 'for', 'us', 'a', 'little', 'treasure',
'in', 'Damansara', 'Kim', '.', '(', 'Credit', 'must', 'also', 'be', 'give
n', 'to', 'former', 'co-owner', 'Mei', 'Wan', 'Tan', '.', ')', 'The', 'na
rcissism', 'you', '’', 'll', 'find', 'in', 'so', 'many', 'KL']
```

In [4]:
```
# depending on domain, tokenizer
print(nltk.wordpunct_tokenize(text))
```

```
['Table', '&', 'Apron', '-', 'formerly', 'The', 'Kitchen', 'Table', 'Rest
aurant', '&', 'Bakery', '-', 'doesn', '’', 't', 'exist', 'to', 'disrupt',
'the', 'scene', '.', 'From', 'the', 'outside', ',', 'it', 'barely', 'stre
tches', 'the', 'boundaries', 'of', 'what', 'is', 'an', 'already', 'satura
ted', 'restaurant', '-', 'cum', '-', 'bakery', 'scene', '.', 'But', 'non
e', 'of', 'it', 'matters', '.', 'Because', 'right', 'from', 'its', 'birt
h', 'in', '2014', ',', 'Table', '&', 'Apron', 'has', 'proven', 'to', 'b
e', 'a', 'restaurant', 'that', 'has', 'in', 'spades', 'a', 'component',
'so', 'elementary', 'yet', 'so', 'rare', '-', 'heart', '.', 'Through', 'h
ard', 'work', ',', 'dedication', 'and', 'all', 'the', 'boring', 'old', '-
', 'fashioned', 'virtues', 'of', 'an', 'honest', 'operation', ',', 'owne
r', 'Marcus', 'Low', 'and', 'his', 'team', 'have', 'carved', 'for', 'us',
'a', 'little', 'treasure', 'in', 'Damansara', 'Kim', '.', '(', 'Credit',
'must', 'also', 'be', 'given', 'to', 'former', 'co', '-', 'owner', 'Mei',
'Wan', 'Tan', '.)', 'The', 'narcissism', 'you', '’', 'll', 'find', 'in',
'so', 'many', 'KL']
```

Difference between word_tokenize and wordpunct_tokenize

Reference: https://stackoverflow.com/questions/50240029/nltk-wordpunct-tokenize-vs-word-tokenize (https://stackoverflow.com/questions/50240029/nltk-wordpunct-tokenize-vs-word-tokenize)

Frequency Distribution

In [5]:
```
from nltk.probability import FreqDist

fdist= FreqDist(words)

print(fdist)
```

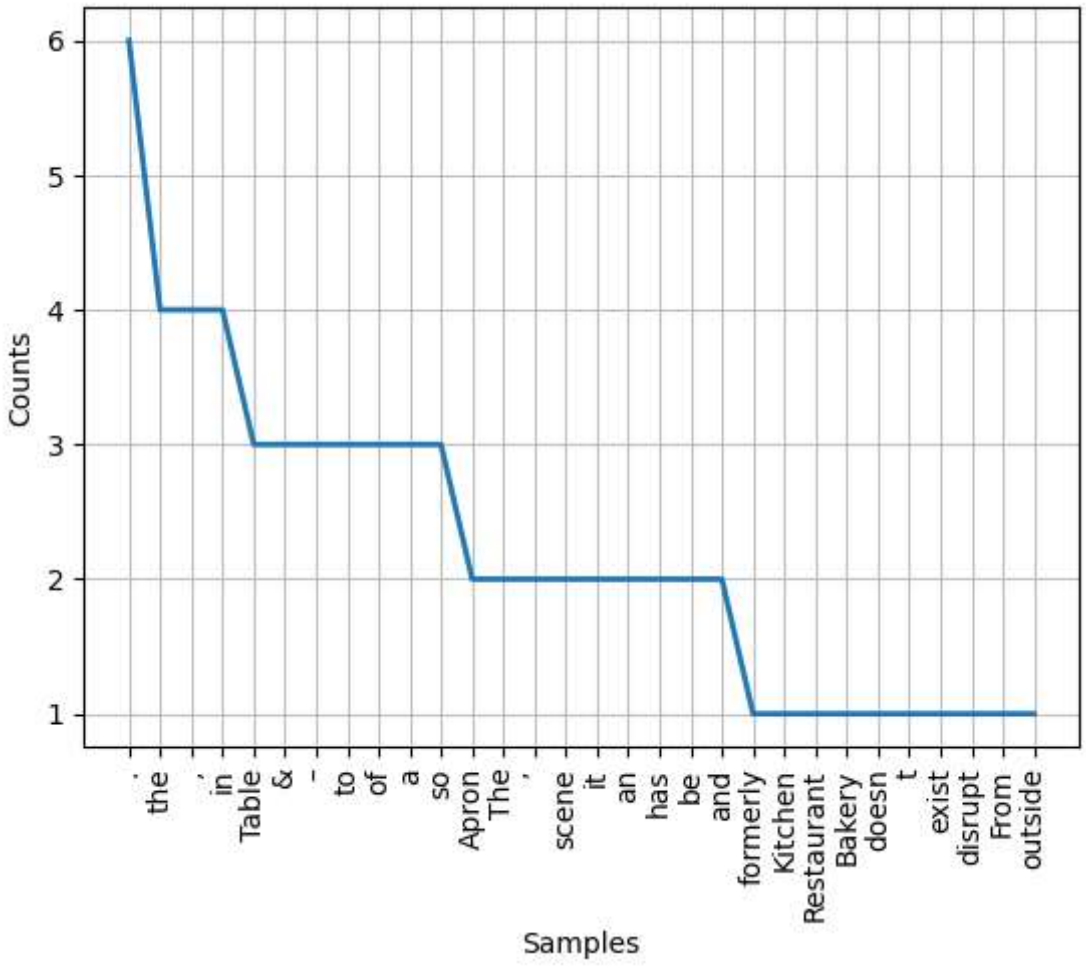
```
<FreqDist with 96 samples and 133 outcomes>
```

In [6]:
```
fdist.most_common(3)
```

Out[6]:
```
[('.', 6), ('the', 4), (',', 4)]
```

In [7]:
```
# Frequency Distribution Plot
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)

plt.show()
```



Stopwords

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

In [10]:
```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

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

Removing Stopwords from our paragraphs

In [12]:
```
tokenized_sent = words

filtered_sent=[]
for w in tokenized_sent:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenized_sent)
print()
print("Filtered Sentence:",filtered_sent)
```

```
Tokenized Sentence: ['Table', '&', 'Apron', '-', 'formerly', 'The', 'Kitc
hen', 'Table', 'Restaurant', '&', 'Bakery', '-', 'doesn', '’', 't', 'exis
t', 'to', 'disrupt', 'the', 'scene', '.', 'From', 'the', 'outside', ',',
'it', 'barely', 'stretches', 'the', 'boundaries', 'of', 'what', 'is', 'a
n', 'already', 'saturated', 'restaurant-cum-bakery', 'scene', '.', 'But',
'none', 'of', 'it', 'matters', '.', 'Because', 'right', 'from', 'its', 'b
irth', 'in', '2014', ',', 'Table', '&', 'Apron', 'has', 'proven', 'to',
'be', 'a', 'restaurant', 'that', 'has', 'in', 'spades', 'a', 'component',
'so', 'elementary', 'yet', 'so', 'rare', '-', 'heart', '.', 'Through', 'h
ard', 'work', ',', 'dedication', 'and', 'all', 'the', 'boring', 'old-fash
ioned', 'virtues', 'of', 'an', 'honest', 'operation', ',', 'owner', 'Marc
us', 'Low', 'and', 'his', 'team', 'have', 'carved', 'for', 'us', 'a', 'li
ttle', 'treasure', 'in', 'Damansara', 'Kim', '.', '(', 'Credit', 'must',
'also', 'be', 'given', 'to', 'former', 'co-owner', 'Mei', 'Wan', 'Tan',
'.', ')', 'The', 'narcissism', 'you', '’', 'll', 'find', 'in', 'so', 'man
y', 'KL']

Filtered Sentence: ['Table', '&', 'Apron', '-', 'formerly', 'The', 'Kitch
en', 'Table', 'Restaurant', '&', 'Bakery', '-', '’', 'exist', 'disrupt',
'scene', '.', 'From', 'outside', ',', 'barely', 'stretches', 'boundarie
s', 'already', 'saturated', 'restaurant-cum-bakery', 'scene', '.', 'But',
'none', 'matters', '.', 'Because', 'right', 'birth', '2014', ',', 'Tabl
e', '&', 'Apron', 'proven', 'restaurant', 'spades', 'component', 'element
ary', 'yet', 'rare', '-', 'heart', '.', 'Through', 'hard', 'work', ',',
'dedication', 'boring', 'old-fashioned', 'virtues', 'honest', 'operatio
n', ',', 'owner', 'Marcus', 'Low', 'team', 'carved', 'us', 'little', 'tre
asure', 'Damansara', 'Kim', '.', '(', 'Credit', 'must', 'also', 'given',
'former', 'co-owner', 'Mei', 'Wan', 'Tan', '.', ')', 'The', 'narcissism',
'’', 'find', 'many', 'KL']
```

Lexicon Normalization

Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

Lemmatization

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, The word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up.

In [14]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ACER\AppData\Roaming\nltk_data...
```

Out[14]:
```
True
```

In [15]:
```
#Lexicon Normalization
#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))
```

```
Lemmatized Word: fly
Stemmed Word: fli
```

Stemming

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".

In [19]:
```
# Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)
```

```
Filtered Sentence: ['Table', '&', 'Apron', '-', 'formerly', 'The', 'Kitch
en', 'Table', 'Restaurant', '&', 'Bakery', '-', '’', 'exist', 'disrupt',
'scene', '.', 'From', 'outside', ',', 'barely', 'stretches', 'boundarie
s', 'already', 'saturated', 'restaurant-cum-bakery', 'scene', '.', 'But',
'none', 'matters', '.', 'Because', 'right', 'birth', '2014', ',', 'Tabl
e', '&', 'Apron', 'proven', 'restaurant', 'spades', 'component', 'element
ary', 'yet', 'rare', '-', 'heart', '.', 'Through', 'hard', 'work', ',',
'dedication', 'boring', 'old-fashioned', 'virtues', 'honest', 'operatio
n', ',', 'owner', 'Marcus', 'Low', 'team', 'carved', 'us', 'little', 'tre
asure', 'Damansara', 'Kim', '.', '(', 'Credit', 'must', 'also', 'given',
'former', 'co-owner', 'Mei', 'Wan', 'Tan', '.', ')', 'The', 'narcissism',
'’', 'find', 'many', 'KL']
Stemmed Sentence: ['tabl', '&', 'apron', '-', 'formerli', 'the', 'kitche
n', 'tabl', 'restaur', '&', 'bakeri', '-', '’', 'exist', 'disrupt', 'scen
e', '.', 'from', 'outsid', ',', 'bare', 'stretch', 'boundari', 'alreadi',
'satur', 'restaurant-cum-bakeri', 'scene', '.', 'but', 'none', 'matter',
'.', 'becaus', 'right', 'birth', '2014', ',', 'tabl', '&', 'apron', 'prov
en', 'restaur', 'spade', 'compon', 'elementari', 'yet', 'rare', '-', 'hea
rt', '.', 'through', 'hard', 'work', ',', 'dedic', 'bore', 'old-fashion',
'virtu', 'honest', 'oper', ',', 'owner', 'marcu', 'low', 'team', 'carv',
'us', 'littl', 'treasur', 'damansara', 'kim', '.', '(', 'credit', 'must',
'also', 'given', 'former', 'co-own', 'mei', 'wan', 'tan', '.', ')', 'th
e', 'narciss', '’', 'find', 'mani', 'kl']
```

POS Tagging

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

Reference: https://pythonspot.com/nltk-speech-tagging/ (https://pythonspot.com/nltk-speech-tagging/)

In [17]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ACER\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[17]: True

In [18]:

```
from nltk import pos_tag
# https://cs.nyu.edu/grishman/jet/guide/PennPOS.html

# print(nltk.pos_tag(words))

print(nltk.pos_tag(words))
```

```
[('Table', 'NNP'), ('&', 'CC'), ('Apron', 'NNP'), ('-', 'NNP'), ('formerl
y', 'RB'), ('The', 'DT'), ('Kitchen', 'NNP'), ('Table', 'NNP'), ('Restaur
ant', 'NNP'), ('&', 'CC'), ('Bakery', 'NNP'), ('-', 'NNP'), ('doesn', 'N
N'), ('’', 'NNP'), ('t', 'VBZ'), ('exist', 'VBP'), ('to', 'TO'), ('disrup
t', 'VB'), ('the', 'DT'), ('scene', 'NN'), ('.', '.'), ('From', 'IN'),
('the', 'DT'), ('outside', 'NN'), (',', ','), ('it', 'PRP'), ('barely',
'RB'), ('stretches', 'VBZ'), ('the', 'DT'), ('boundaries', 'NNS'), ('of',
'IN'), ('what', 'WP'), ('is', 'VBZ'), ('an', 'DT'), ('already', 'RB'),
('saturated', 'VBN'), ('restaurant-cum-bakery', 'JJ'), ('scene', 'NN'),
('.', '.'), ('But', 'CC'), ('none', 'NN'), ('of', 'IN'), ('it', 'PRP'),
('matters', 'NNS'), ('.', '.'), ('Because', 'IN'), ('right', 'NN'), ('fro
m', 'IN'), ('its', 'PRP$'), ('birth', 'NN'), ('in', 'IN'), ('2014', 'C
D'), (',', ','), ('Table', 'NNP'), ('&', 'CC'), ('Apron', 'NNP'), ('has',
'VBZ'), ('proven', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('a', 'DT'), ('res
taurant', 'NN'), ('that', 'WDT'), ('has', 'VBZ'), ('in', 'IN'), ('spade
s', 'NNS'), ('a', 'DT'), ('component', 'NN'), ('so', 'RB'), ('elementar
y', 'JJ'), ('yet', 'RB'), ('so', 'RB'), ('rare', 'JJ'), ('-', 'JJ'), ('he
art', 'NN'), ('.', '.'), ('Through', 'IN'), ('hard', 'JJ'), ('work', 'N
N'), (',', ','), ('dedication', 'NN'), ('and', 'CC'), ('all', 'PDT'), ('t
he', 'DT'), ('boring', 'JJ'), ('old-fashioned', 'JJ'), ('virtues', 'NN
S'), ('of', 'IN'), ('an', 'DT'), ('honest', 'NN'), ('operation', 'NN'),
(',', ','), ('owner', 'NN'), ('Marcus', 'NNP'), ('Low', 'NNP'), ('and',
'CC'), ('his', 'PRP$'), ('team', 'NN'), ('have', 'VBP'), ('carved', 'VB
N'), ('for', 'IN'), ('us', 'PRP'), ('a', 'DT'), ('little', 'JJ'), ('treas
ure', 'NN'), ('in', 'IN'), ('Damansara', 'NNP'), ('Kim', 'NNP'), ('.',
'.'), ('(', '('), ('Credit', 'NNP'), ('must', 'MD'), ('also', 'RB'), ('b
e', 'VB'), ('given', 'VBN'), ('to', 'TO'), ('former', 'JJ'), ('co-owner',
'NN'), ('Mei', 'NNP'), ('Wan', 'NNP'), ('Tan', 'NNP'), ('.', '.'), (')',
')'), ('The', 'DT'), ('narcissism', 'NN'), ('you', 'PRP'), ('’', 'VBP'),
('ll', 'JJ'), ('find', 'VBP'), ('in', 'IN'), ('so', 'RB'), ('many', 'J
J'), ('KL', 'NNP')]
```

In [20]:

```
text = """ Dostoevsky was the son of a doctor.
His parents were very hard-working and deeply religious people,
but so poor that they lived with their five children in only
two rooms. The father and mother spent their evenings
in reading aloud to their children, generally from books of
a serious character."""

words = word_tokenize(text)

import nltk
tagged = nltk.pos_tag(words)
print(tagged)
```

```
[('Dostoevsky', 'NNP'), ('was', 'VBD'), ('the', 'DT'), ('son', 'NN'), ('o
f', 'IN'), ('a', 'DT'), ('doctor', 'NN'), ('.', '.'), ('His', 'PRP$'),
('parents', 'NNS'), ('were', 'VBD'), ('very', 'RB'), ('hard-working', 'J
J'), ('and', 'CC'), ('deeply', 'RB'), ('religious', 'JJ'), ('people', 'NN
S'), (',', ','), ('but', 'CC'), ('so', 'RB'), ('poor', 'JJ'), ('that', 'I
N'), ('they', 'PRP'), ('lived', 'VBD'), ('with', 'IN'), ('their', 'PRP
$'), ('five', 'CD'), ('children', 'NNS'), ('in', 'IN'), ('only', 'RB'),
('two', 'CD'), ('rooms', 'NNS'), ('.', '.'), ('The', 'DT'), ('father', 'N
N'), ('and', 'CC'), ('mother', 'NN'), ('spent', 'VBN'), ('their', 'PRP
$'), ('evenings', 'NNS'), ('in', 'IN'), ('reading', 'VBG'), ('aloud', 'N
N'), ('to', 'TO'), ('their', 'PRP$'), ('children', 'NNS'), (',', ','),
('generally', 'RB'), ('from', 'IN'), ('books', 'NNS'), ('of', 'IN'),
('a', 'DT'), ('serious', 'JJ'), ('character', 'NN'), ('.', '.')]
```

In [21]:
```
import re

stop_words = set(stopwords.words("english"))

text = """ Dostoevsky was the son of a doctor.
His parents were very hard-working and deeply religious people,
but so poor that they lived with their five children in only
two rooms. The father and mother spent their evenings
in reading aloud to their children, generally from books of
a serious character."""

words = word_tokenize(text)

def is_ok(token):
    return re.match('^[a-z]+$', token) and token not in stop_words

filtered = [word for word in word_tokenize(text.lower()) if is_ok(word)]

import nltk
tagged = nltk.pos_tag(filtered)
print(tagged)
```

```
[('dostoevsky', 'JJ'), ('son', 'NN'), ('doctor', 'NN'), ('parents', 'NN
S'), ('deeply', 'RB'), ('religious', 'JJ'), ('people', 'NNS'), ('poor',
'JJ'), ('lived', 'VBD'), ('five', 'CD'), ('children', 'NNS'), ('two', 'C
D'), ('rooms', 'NNS'), ('father', 'RB'), ('mother', 'RB'), ('spent', 'J
J'), ('evenings', 'NNS'), ('reading', 'VBG'), ('aloud', 'JJ'), ('childre
n', 'NNS'), ('generally', 'RB'), ('books', 'NNS'), ('serious', 'JJ'), ('c
haracter', 'NN')]
```

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

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

Out[23]: True

In [26]:
```
from nltk.corpus import brown
brown_tagged_sents = brown.tagged_sents(categories='news')
brown_sents = brown.sents(categories='news')

print(brown_tagged_sents)
```

```
[[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ('Grand', 'JJ-
TL'), ('Jury', 'NN-TL'), ('said', 'VBD'), ('Friday', 'NR'), ('an', 'AT'),
('investigation', 'NN'), ('of', 'IN'), ("Atlanta's", 'NP$'), ('recent',
'JJ'), ('primary', 'NN'), ('election', 'NN'), ('produced', 'VBD'), ('``',
'``'), ('no', 'AT'), ('evidence', 'NN'), ("''", "''"), ('that', 'CS'),
('any', 'DTI'), ('irregularities', 'NNS'), ('took', 'VBD'), ('place', 'N
N'), ('.', '.')], [('The', 'AT'), ('jury', 'NN'), ('further', 'RBR'), ('s
aid', 'VBD'), ('in', 'IN'), ('term-end', 'NN'), ('presentments', 'NNS'),
('that', 'CS'), ('the', 'AT'), ('City', 'NN-TL'), ('Executive', 'JJ-TL'),
('Committee', 'NN-TL'), (',', ','), ('which', 'WDT'), ('had', 'HVD'), ('o
ver-all', 'JJ'), ('charge', 'NN'), ('of', 'IN'), ('the', 'AT'), ('electio
n', 'NN'), (',', ','), ('``', '``'), ('deserves', 'VBZ'), ('the', 'AT'),
('praise', 'NN'), ('and', 'CC'), ('thanks', 'NNS'), ('of', 'IN'), ('the',
'AT'), ('City', 'NN-TL'), ('of', 'IN-TL'), ('Atlanta', 'NP-TL'), ("''",
"''"), ('for', 'IN'), ('the', 'AT'), ('manner', 'NN'), ('in', 'IN'), ('wh
ich', 'WDT'), ('the', 'AT'), ('election', 'NN'), ('was', 'BEDZ'), ('condu
cted', 'VBN'), ('.', '.')], ...]
```

In [29]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk import chunk

text = "The white dog fight with a black cat"
words = word_tokenize(text)
tagged = nltk.pos_tag(words)

grammar = "NP: {<DT>?<JJ>*<NN>}"
cp = nltk.RegexpParser(grammar)
result = cp.parse(tagged)
print(result)
result.draw()
```

```
(S
  (NP The/DT white/JJ dog/NN)
  (NP fight/NN)
  with/IN
  (NP a/DT black/JJ cat/NN))
```

In [54]:

```
from nltk import ne_chunk

def entities(text):
    return ne_chunk(
        pos_tag(
            word_tokenize(text)
        )
    )

tree = entities (text)

tree.pprint()
```

```
  (ORGANIZATION Kitchen/NNP Table/NNP Restaurant/NNP)
  &/CC
  (PERSON Bakery/NNP)
  -/NNP
  doesn/NN
  ’/NNP
  t/VBZ
  exist/VBP
  to/TO
  disrupt/VB
  the/DT
  scene/NN
  ./.
  From/IN
  the/DT
  outside/NN
  ,/,
  it/PRP
  barely/RB
  stretches/VBZ
```

In [56]:
```
#tree.draw

#from nltk.corpus import treebank
#tree = treebank.parsed_sents('wsj_0001.mrg')[0]
tree.draw
```

```
ly', 'RB'), ('The', 'DT'), Tree('ORGANIZATION', [('Kitchen', 'NNP'),
('Table', 'NNP'), ('Restaurant', 'NNP')]), ('&', 'CC'), Tree('PERSON',
[('Bakery', 'NNP')]), ('-', 'NNP'), ('doesn', 'NN'), ('’', 'NNP'),
('t', 'VBZ'), ('exist', 'VBP'), ('to', 'TO'), ('disrupt', 'VB'), ('th
e', 'DT'), ('scene', 'NN'), ('.', '.'), ('From', 'IN'), ('the', 'DT'),
('outside', 'NN'), (',', ','), ('it', 'PRP'), ('barely', 'RB'), ('stret
ches', 'VBZ'), ('the', 'DT'), ('boundaries', 'NNS'), ('of', 'IN'), ('wh
at', 'WP'), ('is', 'VBZ'), ('an', 'DT'), ('already', 'RB'), ('saturate
d', 'VBN'), ('restaurant-cum-bakery', 'JJ'), ('scene', 'NN'), ('.',
'.'), ('But', 'CC'), ('none', 'NN'), ('of', 'IN'), ('it', 'PRP'), ('mat
ters', 'NNS'), ('.', '.'), ('Because', 'IN'), ('right', 'NN'), ('from',
'IN'), ('its', 'PRP$'), ('birth', 'NN'), ('in', 'IN'), ('2014', 'CD'),
(',', ','), Tree('PERSON', [('Table', 'NNP')]), ('&', 'CC'), Tree('PERS
ON', [('Apron', 'NNP')]), ('has', 'VBZ'), ('proven', 'VBN'), ('to', 'T
O'), ('be', 'VB'), ('a', 'DT'), ('restaurant', 'NN'), ('that', 'WDT'),
('has', 'VBZ'), ('in', 'IN'), ('spades', 'NNS'), ('a', 'DT'), ('compone
nt', 'NN'), ('so', 'RB'), ('elementary', 'JJ'), ('yet', 'RB'), ('so',
'RB'), ('rare', 'JJ'), ('-', 'JJ'), ('heart', 'NN'), ('.', '.'), ('Thro
ugh', 'IN'), ('hard', 'JJ'), ('work', 'NN'), (',', ','), ('dedication',
```

In [35]:
```
nltk.download('names')
```

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

Out[35]: True

In [36]:
```
# Step 1: Load Data
from nltk.corpus import names
labeled_names = ([(name, 'male') for name in names.words('male.txt')]
    + [(name, 'female') for name in names.words('female.txt')])

import random
random.shuffle(labeled_names)
#print(labeled_names[:10])

# Step 2: Extract last letter of a name as the feature and form feature set
def feature_extractor(name):
    return {'last_letter': name[-1]}

featureset = [(feature_extractor(name), gender) for (name, gender) in label
# print(featureset[:10])

# Step 3: Split the feature set to training/testing datasets

train_set, test_set = featureset[500:], featureset[:500]

# Step 4/5: Load the classifier and perform training
import nltk
classifier = nltk.NaiveBayesClassifier.train(train_set)

# Step 6: Prediction/Evaluation
print(classifier.classify(feature_extractor('Danny')))
print(nltk.classify.accuracy(classifier, test_set))
```

```
female
0.772
```

In [38]:
```
nltk.download('movie_reviews')
```

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

Out[38]:
```
True
```

In [39]:

```
import nltk

# Step 1: Load Data
from nltk.corpus import movie_reviews
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

import random
random.shuffle(documents)

# Step 2: Extract Feature
all_words = []
for w in movie_reviews.words():
    all_words.append(w.lower())
all_words = nltk.FreqDist(all_words)
word_features = list(all_words.keys())[:3000]

def feature_extractor(review):
    words = set(review)
    features = {}
    for w in word_features:
        features[w] = (w in words)

    return features

featureset = [(feature_extractor(review), sentiment) for (review, sentiment

# Step 3: Split the feature set to training/testing datasets
training_set, testing_set = featureset[:1900],featureset[1900:]

# Step 4/5: Load the classifier and perform training
classifier = nltk.NaiveBayesClassifier.train(training_set)

# Step 6: Prediction/Evaluation
print("Classifier accuracy:",nltk.classify.accuracy(classifier, testing_set
classifier.show_most_informative_features(15)
```

```
Classifier accuracy: 0.77
Most Informative Features
                   regard = True              pos : neg    =     10.9 : 1.
0
                    sucks = True              neg : pos    =     10.3 : 1.
0
                   annual = True              pos : neg    =      9.6 : 1.
0
                      ugh = True              neg : pos    =      9.1 : 1.
0
              silverstone = True              neg : pos    =      7.7 : 1.
0
                  idiotic = True              neg : pos    =      7.1 : 1.
0
                     mena = True              neg : pos    =      7.1 : 1.
0
                   shoddy = True              neg : pos    =      7.1 : 1.
0
                   suvari = True              neg : pos    =      7.1 : 1.
0
            unimaginative = True              neg : pos    =      7.1 : 1.
0
                atrocious = True              neg : pos    =      6.7 : 1.
0
               schumacher = True              neg : pos    =      6.7 : 1.
0
                  jumbled = True              neg : pos    =      6.4 : 1.
0
                   turkey = True              neg : pos    =      6.4 : 1.
0
                  cunning = True              pos : neg    =      6.3 : 1.
0
```

TextBlob

TextBlob does not work in 2023 and beyond anymore

Installation

conda install -c conda-forge textblob

Difference between TextBlob and NLTK

Reference: https://www.quora.com/What-is-the-use-of-NLTK-and-TextBlob-What-is-the-difference-between-both-And-for-text-analysis-which-tool-is-better#:~:text=I%20als-,NLTK%20and%20TextBlob%20are%20both%20excellent%20libraries%2((https://www.quora.com/What-is-the-use-of-NLTK-and-TextBlob-What-is-the-difference-between-both-And-for-text-analysis-which-tool-is-better#:~:text=I%20als-,NLTK%20and%20TextBlob%20are%20both%20excellent%20libraries%2(

In [15]:
```
from textblob import TextBlob

blob = TextBlob(text)

print(blob.noun_phrases)
```

```
['apron', 'kitchen', 'table restaurant', 'bakery', '- doesn ’ t', 'restau
rant-cum-bakery scene', 'apron', 'rare - heart', 'hard work', 'honest ope
ration', 'marcus low', 'damansara kim', 'credit', 'mei wan tan', '’ ll',
'kl']
```

Reference: https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk (https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk)

Quiz

Exploring Features of NLTK:

a. Open the text file for processing: First, we are going to open and read the file which we want to analyze eg. The fishing documentation in txt file (page2). Reference: https://huntfish.mdc.mo.gov/sites/default/files/downloads/page/IntroToFishing_2017_v2.pdf (https://huntfish.mdc.mo.gov/sites/default/files/downloads/page/IntroToFishing_2017_v2.pdf)

b. Import required libraries: For various data processing cases in NLP, we need to import some libraries. In this case, we are going to use NLTK for Natural Language Processing. We will use it to perform various operations on the text.

c. Sentence tokenizing: By tokenizing the text with sent_tokenize( ), we can get the text as sentences.

d. Word tokenizing: By tokenizing the text with word_tokenize( ), we can get the text as words.

e. Find the frequency distribution: Let’s find out the frequency of words in our text.

f. Plot the frequency graph: Let’s plot a graph to visualize the word distribution in our text.

g. Remove punctuation marks: Next, we are going to remove the punctuation marks as they are not very useful for us. We are going to use isalpha( ) method to separate the punctuation marks from the actual text. Also, we are going to make a new list called words_no_punc, which will store the words in lower case but exclude the punctuation marks.

h. Plotting graph without punctuation marks:

i. List of stopwords:

j. Removing stopwords:

k. Final frequency distribution: the final graph has many useful words that help us understand what our sample data is about, showing how essential it is to perform data cleaning on NLP.

Hint: https://medium.com/towards-artificial-intelligence/natural-language-processing-nlp-with-python-tutorial-for-beginners-1f54e610a1a0 (https://medium.com/towards-artificial-

In [1]:

```
import PyPDF2
import numpy as np
import matplotlib.pyplot as plt
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
from nltk.probability import FreqDist
from nltk.corpus import stopwords

with open('IntroToFishing_2017_v2.pdf','rb') as pdf_file, open('IntroToFish
    read_pdf = PyPDF2.PdfFileReader(pdf_file)
    number_of_pages = read_pdf.getNumPages()
    for page_number in range(number_of_pages):
        page = read_pdf.getPage(page_number)
        page_content = page.extractText()
        text_file.write(page_content)

text = open('./IntroToFishing_2017_v2.txt', encoding='utf-8').read()
text = text.replace ('  ',' ')

sentences = nltk.sent_tokenize(text)
print("Tokenized sentences:", sentences)
print()

words = word_tokenize(text)
print("Tokenized words:", words)
print()

fdist = FreqDist(words)
print("Frequent words:", fdist.most_common())
print()

fdist.plot(30,cumulative=False)
plt.show()


# Without punctuations and stop words
words2 = nltk.wordpunct_tokenize(text)
words_no_punc = []
stop_words = set(stopwords.words("english"))
print("Stopwords:", stop_words)
print()

for w in words2:
    if w not in stop_words and w.isalpha():
        words_no_punc.append(w.lower())

fdist2 = FreqDist(words_no_punc)
print("Frequent words:", fdist2.most_common())
print()

fdist2.plot(30,cumulative=False)
```

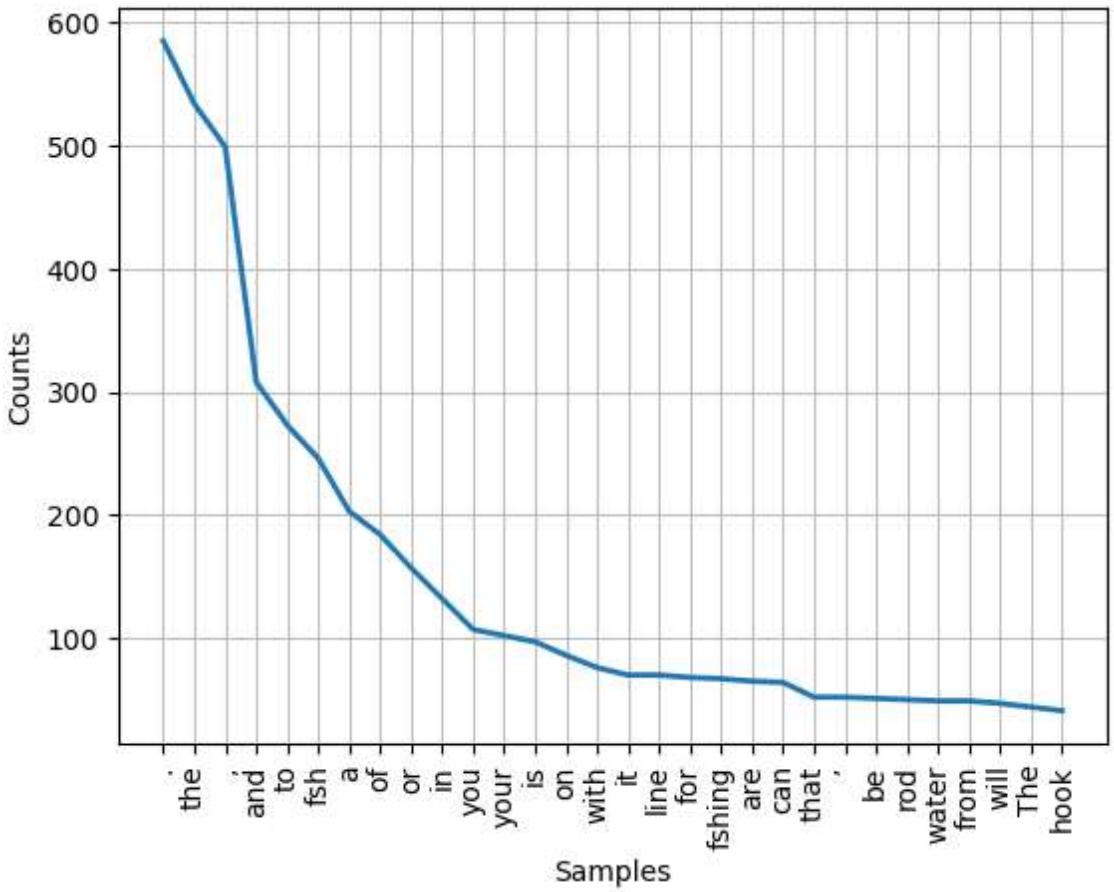
```
plt.show()
```

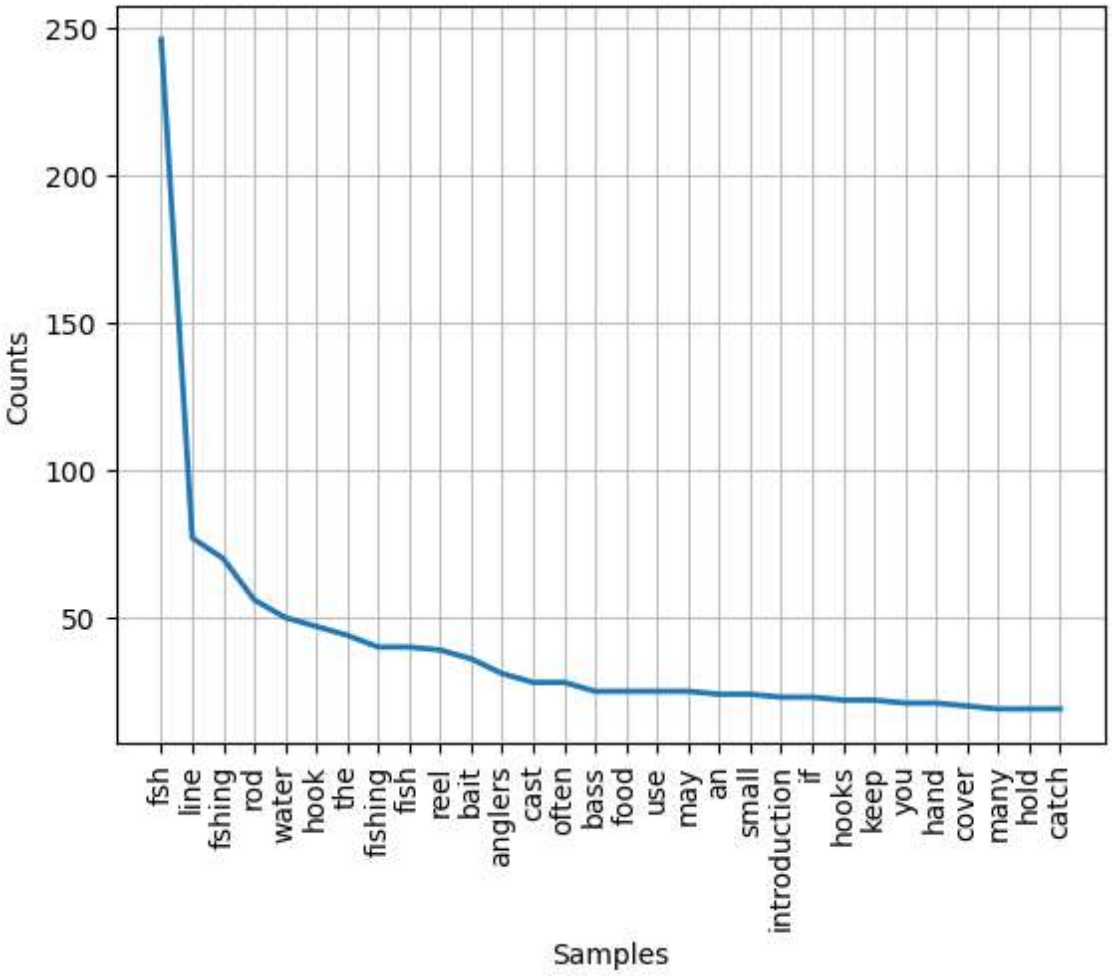
```
Tokenized sentences: ['An Introduction to FISHING                    \nFi
shing \n“God never did make a more calm, quiet, innocent recreation
....” Izaak Walton \n2 | An Introduction to Fishing Fishing is a great
way to spend a day.', 'You can take a lunch and picnic as you fsh.', 'Y
ou can camp near a lake.', 'You can hike or boat to a fshing spot.', 'M
any people build their vacations around fshing.', 'Be sure to take your
family or your friends along, for there is no more sociable activity.',
'Missouri has more than 800,000 acres of surface water, and most of it
provides great fshing.', 'Our waters hold ancient paddlefsh, wary large
mouth bass, and tasty bluegill—more than 200 diferent species.', 'About
40 of those fsh species are the targets of anglers.', 'Some Missourians
fsh for sport or relaxation, while others fsh only for food.', 'Regardl
ess of motivation, the majority of anglers reap all the benefts of fshi
ng.', 'They spend quality time on the water and then return home to a s
atisfying meal of fried or grilled fsh they have caught themselves.',
'One of the joys of fshing is that it can be fun and productive at any
skill level.', 'You can complicate the sport with jargon and sophistica
ted equipment, but the whole sport can be pared down to some basic equi
pment and techniques.', 'This publication presents those basics to yo
u.', 'It would take many volumes to introduce you to all the species of
```

In [2]:

```
# Graph 1
fdist.plot(30,cumulative=False)
plt.show()


# Graph 2 - Without punctuations and stop words
fdist2.plot(30,cumulative=False)
plt.show()
```




250
200
150
100
50
Counts
fsh
line
fshing
rod
water
hook
the
fishing
fish
reel
bait
anglers
cast
often
bass
food
use
may
an
small
introduction
if
hooks
keep
you
hand
cover
many
hold
catch
Samples