AI Basics

NLTK Guide & Cheatsheet

+++ 20 Code Examples +++

```
## NLP ## NLU
from nltk import *
```

#Python

/sumitkhanna

Python

Comprehensive Guide to NLTK Framework

The Natural Language Toolkit (NLTK) is a powerful Python library used for working with human language data (text). It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK is widely used in research and industry for tasks such as text analysis, sentiment analysis, and natural language processing (NLP).

Cheat Sheet of Useful NLTK Methods

Method Name	Definition
`nltk.download`	Download NLTK datasets and models
`word_tokenize`	Tokenize a string to words
`sent_tokenize`	Tokenize a string to sentences
`pos_tag`	Part-of-speech tagging for tokens
`ne_chunk`	Named Entity Recognition
`FreqDist`	Frequency distribution of words
`ConditionalFreqDist`	Conditional frequency distribution of words
`Text`	Create a Text object for analysis
`concordance`	Find concordance of words in text
`similar`	Find words similar to a given word
`common_contexts`	Find common contexts shared by two words
`dispersion_plot`	Display a lexical dispersion plot
`generate`	Generate random text based on a language model
`bigrams`	Generate bigrams from text
`ngrams`	Generate n-grams from text

Method Name	Definition
`wordnet.synsets`	Get synsets for a word using WordNet
`wordnet.lemmas`	Get lemmas for a synset using WordNet
`wordnet.synset`	Get a specific synset using WordNet
`wordnet.morphy`	Find the base form of a word using WordNet
`wordnet.synset.lemma_names`	Get lemma names of a synset
`wordnet.synset.definition`	Get the definition of a synset
`wordnet.synset.examples`	Get example sentences for a synset
`wordnet.synset.hypernyms`	Get hypernyms of a synset
`wordnet.synset.hyponyms`	Get hyponyms of a synset
`wordnet.synset.member_holonyms`	Get member holonyms of a synset
`wordnet.synset.part_meronyms`	Get part meronyms of a synset
`nltk.corpus.words.words`	Get a list of words from the Words corpus
`nltk.corpus.stopwords.words`	Get a list of stopwords for a language
`nltk.corpus.gutenberg.raw`	Get raw text from the Gutenberg corpus
`nltk.corpus.brown.words`	Get words from the Brown corpus
`nltk.corpus.reuters.words`	Get words from the Reuters corpus
`nltk.corpus.inaugural.words`	Get words from the Inaugural Address corpus
`nltk.corpus.webtext.words`	Get words from the Web Text corpus
`nltk.corpus.treebank.parsed_sents`	Get parsed sentences from the Treebank corpus
`nltk.corpus.semcor.tagged_sents`	Get tagged sentences from the SemCor corpus
`nltk.corpus.names.words`	Get names from the Names corpus
`nltk.corpus.sentiwordnet.senti_synset`	Get a SentiSynset from SentiWordNet
`nltk.corpus.wordnet_ic.ic`	Get information content from WordNet IC corpus
`nltk.corpus.nps_chat.tagged_posts`	Get tagged posts from the NPS Chat corpus
`nltk.corpus.movie_reviews.words`	Get words from the Movie Reviews corpus
`nltk.corpus.twitter_samples.strings`	Get strings from the Twitter Samples corpus
`nltk.classify.NaiveBayesClassifier`	A Naive Bayes classifier for text classification

Method Name	Definition
`nltk.classify.DecisionTreeClassifier`	A Decision Tree classifier for text classification
`nltk.tag.PerceptronTagger`	A part-of-speech tagger using the Averaged Perceptron algorithm
`nltk.tag.HMMTagger`	A Hidden Markov Model part-of-speech tagger
`nltk.chunk.RegexpParser`	A regular expression based chunk parser
`nltk.translate.bleu_score`	Calculate BLEU score for machine translation evaluation
`nltk.stem.PorterStemmer`	Porter Stemmer for stemming words
`nltk.stem.LancasterStemmer`	Lancaster Stemmer for stemming words
`nltk.stem.SnowballStemmer`	Snowball Stemmer for stemming words
`nltk.stem.WordNetLemmatizer`	WordNet Lemmatizer for lemmatizing words

Detailed Explanation and Usage of Each Method

`nltk.download`

This method is used to download the necessary NLTK datasets and models. It's essential for setting up your environment with the required resources.

```
import nltk

# Download the NLTK datasets and models
nltk.download()
```

`word_tokenize`

Tokenizes a given string into individual words.

```
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
print(tokens)
```

`sent_tokenize`

Tokenizes a given string into sentences.

```
from nltk.tokenize import sent_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. I
sentences = sent_tokenize(text)
print(sentences)
```

`pos_tag`

Performs part-of-speech tagging for a list of tokens.

```
from nltk import pos_tag
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
tagged_tokens = pos_tag(tokens)
print(tagged_tokens)
```

`ne_chunk`

Performs named entity recognition on a list of tagged tokens.

```
from nltk import ne_chunk, pos_tag
from nltk.tokenize import word_tokenize

text = "Barack Obama was born in Hawaii."
tokens = word_tokenize(text)
tagged_tokens = pos_tag(tokens)
named_entities = ne_chunk(tagged_tokens)
print(named_entities)
```

`FreqDist`

Calculates the frequency distribution of words in a text.

```
from nltk.probability import FreqDist
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. I
tokens = word_tokenize(text)
fdist = FreqDist(tokens)
print(fdist.most_common(5))
```

ConditionalFreqDist

Calculates the conditional frequency distribution of words.

```
from nltk.probability import ConditionalFreqDist
from nltk.corpus import brown

cfd = ConditionalFreqDist(
    (genre, word)
    for genre in brown.categories()
    for word in brown.words(categories=genre)
)

print(cfd['news'].most_common(10))
```

Text

Creates a Text object for text analysis.

```
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.concordance('NLTK')
```

concordance

Finds the concordance of a word in a text.

```
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.concordance('NLTK')
```

`similar`

Finds words similar to a given word.

```
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.similar('NLTK')
```

`common_contexts`

Find

s common contexts shared by two words.

```
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.common_contexts(['NLTK', 'platform'])
```

`dispersion_plot`

Displays a lexical dispersion plot.

```
from nltk.draw.dispersion import dispersion_plot
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
dispersion_plot(tokens, ['NLTK', 'platform'])
```

generate

Generates random text based on a language model.

```
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.generate()
```

bigrams

Generates bigrams from text.

```
from nltk import bigrams
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
bigrams_list = list(bigrams(tokens))
print(bigrams_list)
```

ngrams

Generates n-grams from text.

```
from nltk import ngrams
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
trigrams_list = list(ngrams(tokens, 3))
print(trigrams_list)
```

`wordnet.synsets`

Gets synsets for a word using WordNet.

```
from nltk.corpus import wordnet

synsets = wordnet.synsets('dog')
print(synsets)
```

`wordnet.lemmas`

Gets lemmas for a synset using WordNet.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
lemmas = synset.lemmas()
print(lemmas)
```

`wordnet.synset`

Gets a specific synset using WordNet.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
print(synset)
```

`wordnet.morphy`

Finds the base form of a word using WordNet.

```
from nltk.corpus import wordnet

base_form = wordnet.morphy('running')
print(base_form)
```

`wordnet.synset.lemma_names`

Gets lemma names of a synset.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
lemma_names = synset.lemma_names()
print(lemma_names)
```

`wordnet.synset.definition`

Gets the definition of a synset.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
definition = synset.definition()
print(definition)
```

`wordnet.synset.examples`

Gets example sentences for a synset.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
examples = synset.examples()
print(examples)
```

`wordnet.synset.hypernyms`

Gets hypernyms of a synset.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
hypernyms = synset.hypernyms()
print(hypernyms)
```

wordnet.synset.hyponyms

Gets hyponyms of a synset.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
hyponyms = synset.hyponyms()
print(hyponyms)
```

wordnet.synset.member_holonyms

Gets member holonyms of a synset.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
member_holonyms = synset.member_holonyms()
print(member_holonyms)
```

wordnet.synset.part_meronyms

Gets part meronyms of a synset.

```
from nltk.corpus import wordnet

synset = wordnet.synset('dog.n.01')
part_meronyms = synset.part_meronyms()
print(part_meronyms)
```

nltk.corpus.words.words

Gets a list of words from the Words corpus.

```
from nltk.corpus import words

word_list = words.words()
print(word_list[:10])
```

`nltk.corpus.stopwords.words`

Gets a list of stopwords for a language.

```
from nltk.corpus import stopwords

stopword_list = stopwords.words('english')
print(stopword_list)
```

`nltk.corpus.gutenberg.raw`

Gets raw text from the Gutenberg corpus.

```
from nltk.corpus import gutenberg

raw_text = gutenberg.raw('austen-emma.txt')
print(raw_text[:1000])
```

`nltk.corpus.brown.words`

Gets words from the Brown corpus.

```
from nltk.corpus import brown

brown_words = brown.words()
print(brown_words[:10])
```

`nltk.corpus.reuters.words`

Gets words from the Reuters corpus.

```
from nltk.corpus import reuters

reuters_words = reuters.words()
print(reuters_words[:10])
```

`nltk.corpus.inaugural.words`

Gets words from the Inaugural Address corpus.

```
from nltk.corpus import inaugural

inaugural_words = inaugural.words()
print(inaugural_words[:10])
```

`nltk.corpus.webtext.words`

Gets words from the Web Text corpus.

```
from nltk.corpus import webtext

webtext_words = webtext.words()
print(webtext_words[:10])
```

`nltk.corpus.treebank.parsed_sents`

Gets parsed sentences from the Treebank corpus.

```
from nltk.corpus import treebank

parsed_sents = treebank.parsed_sents()
print(parsed_sents[:1])
```

`nltk.corpus.semcor.tagged_sents`

Gets tagged sentences from the SemCor corpus.

```
from nltk.corpus import semcor

tagged_sents = semcor.tagged_sents()
print(tagged_sents[:1])
```

`nltk.corpus.names.words`

Gets names from the Names corpus.

```
from nltk.corpus import names

names_list = names.words()
print(names_list[:10])
```

nltk.corpus.sentiwordnet.senti_synset

Gets a SentiSynset from SentiWordNet.

```
from nltk.corpus import sentiwordnet as swn

senti_synset = swn.senti_synset('dog.n.01')
print(senti_synset)
```

nltk.corpus.wordnet_ic.ic

Gets information content from the WordNet IC corpus.

```
from nltk.corpus import wordnet_ic

ic = wordnet_ic.ic('ic-brown.dat')
print(ic)
```

nltk.corpus.nps_chat.tagged_posts

Gets tagged posts from the NPS Chat corpus.

```
from nltk.corpus import nps_chat

tagged_posts = nps_chat.tagged_posts()
print(tagged_posts[:1])
```

nltk.corpus.movie_reviews.words

Gets words from the Movie Reviews corpus.

```
from nltk.corpus import movie_reviews

movie_reviews_words = movie_reviews.words()
print(movie_reviews_words[:10])
```

`nltk.corpus.twitter_samples.strings`

Gets strings from the Twitter Samples corpus.

```
from nltk.corpus import twitter_samples

tweets = twitter_samples.strings()
print(tweets[:1])
```

`nltk.classify.NaiveBayesClassifier`

A Naive Bayes classifier for text classification.

```
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews
import random

documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features

featuresets = [(document_features(d), c) for (d, c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]
classifier = NaiveBayesClassifier.train(train_set)

print(nltk.classify.accuracy(classifier, test_set))
classifier.show_most_informative_features(5)
```

`nltk.classify.DecisionTreeClassifier`

A Decision Tree classifier for text classification.

```
from nltk.classify import DecisionTreeClassifier
from nltk.corpus import movie_reviews
import random

documents = [(list(movie_reviews.words(fileid

)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features

featuresets = [(document_features(d), c) for (d, c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]
classifier = DecisionTreeClassifier.train(train_set)

print(nltk.classify.accuracy(classifier, test_set))
```

`nltk.tag.PerceptronTagger`

A part-of-speech tagger using the Averaged Perceptron algorithm.

```
from nltk.tag import PerceptronTagger
from nltk.tokenize import word_tokenize

tagger = PerceptronTagger()
text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
tagged_tokens = tagger.tag(tokens)
print(tagged_tokens)
```

`nltk.tag.HMMTagger`

A Hidden Markov Model part-of-speech tagger.

```
from nltk.tag import hmm
from nltk.corpus import treebank

trainer = hmm.HiddenMarkovModelTrainer()
tagged_sents = treebank.tagged_sents()
tagger = trainer.train(tagged_sents)

text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
tagged_tokens = tagger.tag(tokens)
print(tagged_tokens)
```

`nltk.chunk.RegexpParser`

A regular expression based chunk parser.

```
from nltk.chunk import RegexpParser
from nltk.tokenize import word_tokenize
from nltk import pos_tag

text = "The quick brown fox jumps over the lazy dog."
tokens = word_tokenize(text)
tagged_tokens = pos_tag(tokens)

grammar = "NP: {<DT>?<JJ>*<NN>}"
chunk_parser = RegexpParser(grammar)
tree = chunk_parser.parse(tagged_tokens)
print(tree)
```

`nltk.translate.bleu_score`

Calculates BLEU score for machine translation evaluation.

```
from nltk.translate.bleu_score import sentence_bleu

reference = [['this', 'is', 'a', 'test']]
candidate = ['this', 'is', 'test']
score = sentence_bleu(reference, candidate)
print(score)
```

`nltk.stem.PorterStemmer`

Porter Stemmer for stemming words.

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
word = "running"
stemmed_word = stemmer.stem(word)
print(stemmed_word)
```

nltk.stem.LancasterStemmer

Lancaster Stemmer for stemming words.

```
from nltk.stem import LancasterStemmer

stemmer = LancasterStemmer()
word = "running"
stemmed_word = stemmer.stem(word)
print(stemmed_word)
```

nltk.stem.SnowballStemmer

Snowball Stemmer for stemming words.

```
from nltk.stem import SnowballStemmer

stemmer = SnowballStemmer("english")
word = "running"
stemmed_word = stemmer.stem(word)
print(stemmed_word)
```

nltk.stem.WordNetLemmatizer

WordNet Lemmatizer for lemmatizing words.

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
word = "running"
lemmatized_word = lemmatizer.lemmatize(word, pos='v')
print(lemmatized_word)
```

Code Examples Using NLTK for Useful Use Cases

1. Sentiment Analysis Using Naive Bayes Classifier

```
import nltk
from nltk.corpus import movie_reviews
import random
from nltk.classify import NaiveBayesClassifier
from nltk.classify.util import accuracy as nltk_accuracy

# Load the dataset
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]
random.shuffle(documents)

# Define a feature extractor function
all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features

# Create feature sets
featuresets = [(document_features(d), c) for (d, c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]

# Train a Naive Bayes classifier
classifier = NaiveBayesClassifier.train(train_set)

# Evaluate the classifier
print('Accuracy:', nltk_accuracy(classifier, test_set))

# Show the most informative features
classifier.show_most_informative_features(5)
```

2. Named Entity Recognition

```
import nltk
from nltk import ne_chunk, pos_tag
from nltk.tokenize import word_tokenize

text = "Barack Obama was born in Hawaii. He was elected president in 2008."
tokens = word_tokenize(text)
tagged_tokens = pos_tag(tokens)
named_entities = ne_chunk(tagged_tokens)
print(named_entities)
```

3. Part-of-Speech Tagging with PerceptronTagger

```
import nltk
from nltk.tag import PerceptronTagger
from nltk.tokenize import word_tokenize

tagger = PerceptronTagger()
text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
tagged_tokens = tagger.tag(tokens)
print(tagged_tokens)
```

4. Building a Language Model

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.lm import MLE
from nltk.lm.preprocessing import padded_everygram_pipeline

# Sample text
text = "NLTK is a leading platform for building Python programs to work with human language data. I

# Tokenize the text
tokens = word_tokenize(text.lower())

# Prepare the data for language modeling
n = 3
train_data, padded_vocab = padded_everygram_pipeline(n, [tokens])

# Train the language model
model = MLE(n)
model.fit(train_data, padded_vocab)

# Generate text
context = ('nltk', 'is')
print('Generated text:', ' '.join(model.generate(10, text_seed=context)))
```

5. Synonym Extraction Using WordNet

```
import nltk
from nltk.corpus import wordnet

word = "happy"
synonyms = []
for syn in wordnet.synsets(word):
    for lemma in syn.lemmas():
        synonyms.append(lemma.name())

print(set(synonyms))
```

6. Word Sense Disambiguation

```
import nltk
from nltk.corpus import wordnet
from nltk.wsd import lesk

sentence = "I went to the bank to deposit money"
ambiguous_word = "bank"
context = sentence.split()

# Get the correct sense
sense = lesk(context, ambiguous_word)
print(sense, sense.definition())
```

7. Chunking Using Regular Expressions

```
import nltk
from nltk.chunk import RegexpParser
from nltk.tokenize import word_tokenize
from nltk import pos_tag

text = "The quick brown fox jumps over the lazy dog."
tokens = word_tokenize(text)
tagged_tokens = pos_tag(tokens)

grammar = "NP: {<DT>?<JJ>*<NN>}"
chunk_parser = RegexpParser(grammar)
tree = chunk_parser.parse(tagged_tokens)
print(tree)
```

8. Calculating BLEU Score

```
import nltk
from nltk.translate.bleu_score import sentence_bleu

reference = [['this', 'is', 'a', 'test']]
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate)
print(score)
```

9. Text Classification Using DecisionTreeClassifier

```
import nltk
from nltk.classify import DecisionTreeClassifier
from nltk.corpus import movie_reviews
import random

documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

random.shuffle(documents)

all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
word_features = list(all_words)[:2000]

def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features

featuresets = [(document_features(d), c) for (d, c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]
classifier = DecisionTreeClassifier.train(train_set)

print(nltk.classify.accuracy(classifier, test_set))
```

10. Stemming with PorterStemmer

```
import nltk
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
words = ["running", "jumps", "easily", "fairly"]
stemmed

_words = [stemmer.stem(word) for word in words]
print(stemmed_words)
```

11. Lemmatization with WordNetLemmatizer

```
import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
words = ["running", "jumps", "easily", "fairly"]
lemmatized_words = [lemmatizer.lemmatize(word, pos='v') for word in words]
print(lemmatized_words)
```

12. Frequency Distribution of Words

```
import nltk
from nltk.probability import FreqDist
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. I
tokens = word_tokenize(text)
fdist = FreqDist(tokens)
print(fdist.most_common(5))
```

13. Conditional Frequency Distribution

```
import nltk
from nltk.probability import ConditionalFreqDist
from nltk.corpus import brown

cfd = ConditionalFreqDist(
    (genre, word)
    for genre in brown.categories()
    for word in brown.words(categories=genre)
)

print(cfd['news'].most_common(10))
```

14. Finding Synonyms with WordNet

```
import nltk
from nltk.corpus import wordnet

word = "good"
synonyms = []
for syn in wordnet.synsets(word):
    for lemma in syn.lemmas():
        synonyms.append(lemma.name())

print(set(synonyms))
```

15. Part-of-Speech Tagging

```
import nltk
from nltk import pos_tag
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
tagged_tokens = pos_tag(tokens)
print(tagged_tokens)
```

16. Named Entity Recognition

```
import nltk
from nltk import ne_chunk, pos_tag
from nltk.tokenize import word_tokenize

text = "Barack Obama was born in Hawaii."
tokens = word_tokenize(text)
tagged_tokens = pos_tag(tokens)
named_entities = ne_chunk(tagged_tokens)
print(named_entities)
```

17. Concordance

```
import nltk
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.concordance('NLTK')
```

18. Word Similarity

```
import nltk
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.similar('NLTK')
```

19. Common Contexts

```
import nltk
from nltk.text import Text
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
text_obj = Text(tokens)
text_obj.common_contexts(['NLTK', 'platform'])
```

20. Dispersion Plot

```
import nltk
from nltk.draw.dispersion import dispersion_plot
from nltk.tokenize import word_tokenize

text = "NLTK is a leading platform for building Python programs to work with human language data. N
tokens = word_tokenize(text)
dispersion_plot(tokens, ['NLTK', 'platform'])
```

Follow me on

Sumit Khanna for more updates