1. Tokenization

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
nltk.download('punkt')

text = "I believe this would help the reader understand how tokenization \
        works. as well as realize its importance."
print("Sentence tokenization: ")
sents = (sent_tokenize(text))
print(sents)

print ("Word tokenization: ", word_tokenize(text))


print("Word tokenization with list of list of each word: ")
words = [word_tokenize(sent) for sent in sents]
print(words)
```

```
Sentence tokenization:
['I believe this would help the reader understand how tokenization         works.', 'as well as realize its importance.']
Word tokenization:  ['I', 'believe', 'this', 'would', 'help', 'the', 'reader', 'understand', 'how', 'tokenization', 'works', '.', 'as',
Word tokenization with list of list of each word:
[['I', 'believe', 'this', 'would', 'help', 'the', 'reader', 'understand', 'how', 'tokenization', 'works', '.'], ['as', 'well', 'as', 're
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

2. Stop word removal

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from string import punctuation

text = "I believe this would help the reader understand how tokenization \
        works. as well as realize its importance (text) ."

custom_list = set(stopwords.words('english')+list(punctuation))

word_list = [word for word in word_tokenize(text) if word not in custom_list]
print(word_list)
```

```
['I', 'believe', 'would', 'help', 'reader', 'understand', 'tokenization', 'works', 'well', 'realize', 'importance', 'text']
```

3. N-Gram

```
from nltk.collocations import BigramCollocationFinder

word_list = ['I', 'believe', 'would', 'help', 'reader', 'understand', \
             'tokenization', 'works', 'well', 'realize', 'importance', 'text']

finde = BigramCollocationFinder.from_words(word_list)
print(finde.ngram_fd.items())
```

```
dict_items([(('I', 'believe'), 1), (('believe', 'would'), 1), (('would', 'help'), 1), (('help', 'reader'), 1), (('reader', 'understand')
```

4. Word Sense Disambiguation(WSD)

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
from nltk.corpus import wordnet
for ss in wordnet.synsets('mouse'):
    print(ss, ss.definition())

from nltk.wsd import lesk
from nltk.tokenize import word_tokenize

print("------------WSD---------------")
print("For bass: ")
context_1 = lesk(word_tokenize("Sing in a lower tone, along with the bass"), "bass")
print(context_1, context_1.definition())

context_2 = lesk(word_tokenize("The sea bass really very hard to catch"), "bass")
print(context_2, context_2.definition())

print("For mouse: ")
context_3 = lesk(word_tokenize("My mouse is not working, need to change it"), "mouse")
print(context_3, context_3.definition())
```

```
Synset('mouse.n.01') any of numerous small rodents typically resembling diminutive rats having pointed snouts and small ears on elongate
Synset('shiner.n.01') a swollen bruise caused by a blow to the eye
Synset('mouse.n.03') person who is quiet or timid
Synset('mouse.n.04') a hand-operated electronic device that controls the coordinates of a cursor on your computer screen as you move it
Synset('sneak.v.01') to go stealthily or furtively
Synset('mouse.v.02') manipulate the mouse of a computer
------------WSD---------------
For bass:
Synset('bass.n.07') the member with the lowest range of a family of musical instruments
Synset('sea_bass.n.01') the lean flesh of a saltwater fish of the family Serranidae
For mouse:
Synset('mouse.n.04') a hand-operated electronic device that controls the coordinates of a cursor on your computer screen as you move it
```

5. Stemming

```
from nltk.tokenize import word_tokenize
from nltk.stem.lancaster import LancasterStemmer

new_text = "It is important to by very pythonly while you are pythoning\
             with python. All pythoners have pythoned poorly at least once."

l_s =  LancasterStemmer()
stem_lan =  [l_s.stem(word) for word in word_tokenize(new_text)]
print(stem_lan)
```

```
['it', 'is', 'import', 'to', 'by', 'very', 'python', 'whil', 'you', 'ar', 'python', 'with', 'python', '.', 'al', 'python', 'hav', 'pytho
```

6. Count Vectorizer

```
import pandas as pd
corpus = [
     'This is the first document from heaven',
     'but the second document is from mars',
     'And this is the third one from nowhere',
     'Is this the first document from nowhere?',
]

df = pd.DataFrame({'Text':corpus})
print(df)

from sklearn.feature_extraction.text import CountVectorizer
```

```
count_v = CountVectorizer()
X = count_v.fit_transform(df.Text).toarray()
print(count_v.get_feature_names())

print(X)
print(count_v.vocabulary_)

count_v = CountVectorizer(stop_words=['this','is'])
X = count_v.fit_transform(df.Text).toarray()

print(X)
```

```
                                       Text
0    This is the first document from heaven
1      but the second document is from mars
2    And this is the third one from nowhere
3  Is this the first document from nowhere?
['and', 'but', 'document', 'first', 'from', 'heaven', 'is', 'mars', 'nowhere', 'one', 'second', 'the', 'third', 'this']
[[0 0 1 1 1 1 1 0 0 0 0 1 0 1]
 [0 1 1 0 1 0 1 1 0 0 1 1 0 0]
 [1 0 0 0 1 0 1 0 1 1 0 1 1 1]
 [0 0 1 1 1 0 1 0 1 0 0 1 0 1]]
{'this': 13, 'is': 6, 'the': 11, 'first': 3, 'document': 2, 'from': 4, 'heaven': 5, 'but': 1, 'second': 10, 'mars': 7, 'and': 0, 'third
[[0 0 1 1 1 1 0 0 0 0 1 0]
 [0 1 1 0 1 0 1 0 0 1 1 0]
 [1 0 0 0 1 0 0 1 1 0 1 1]
 [0 0 1 1 1 0 0 1 0 0 1 0]]
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_fea
  warnings.warn(msg, category=FutureWarning)
```

7. TF-IDF Vectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [
     'This is the first document from heaven',
     'but the second document is from mars',
     'And this is the third one from nowhere',
     'Is this the first document from nowhere?',
]

vectorizer = TfidfVectorizer()
vectorizer.fit(corpus)
print(vectorizer.vocabulary_)
print(vectorizer.idf_)
```

```
{'this': 13, 'is': 6, 'the': 11, 'first': 3, 'document': 2, 'from': 4, 'heaven': 5, 'but': 1, 'second': 10, 'mars': 7, 'and': 0, 'third
[1.91629073 1.91629073 1.22314355 1.51082562 1.         1.91629073
 1.         1.91629073 1.51082562 1.91629073 1.91629073 1.
 1.91629073 1.22314355]
```

8. Hashing

```
from sklearn.feature_extraction.text import HashingVectorizer
import pandas as pd
corpus = [
     'This is the first document from heaven',
     'but the second document is from mars',
     'And this is the third one from nowhere',
     'Is this the first document from nowhere?',
]

df = pd.DataFrame({'Text':corpus})

hash_v = HashingVectorizer(n_features=15, norm=None, alternate_sign=True)
print(hash_v.fit_transform(df.Text).toarray())
```

```
[[ 0.  0. -1.  0.  0.  0.  0.  0.  0.  0. -1.  1.  0.  0.]
 [ 0.  0.  0. -1.  0. -1.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0. -1.  1.  0.  0.  0.  0.  0.  2.  0. -1.]
 [ 0.  0. -1. -1.  0.  0.  0.  0.  0.  0.  0. -1.  2.  0.  0.]]
```

✓ 0s completed at 8:35 PM