Que1. Text Analysis using Python

Install the libraries.

```
!pip install nltk

     Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
     Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
     Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
     Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.12.25)
     Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.2)
```

Import NLTK and Download the required data.

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

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

Tokenize text

```
from nltk.tokenize import word_tokenize


text = "This is a sample text for our text analysis tool."
tokens= word_tokenize(text)
print(tokens)

     ['This', 'is', 'a', 'sample', 'text', 'for', 'our', 'text', 'analysis', 'tool', '.']
```

Remove the stopwords.

Stopwords are the words that do not carry much meaning and can be removed to just reduce the size of the data and reduce the computation on the unecessary words.

```
from nltk.corpus import stopwords


stop_words= set(stopwords.words('english'))
filtered_tokens= [token for token in tokens if token.lower() not in stop_words]
print(filtered_tokens)

     ['sample', 'text', 'text', 'analysis', 'tool', '.']
```

Perform the Stemming and Lemmatization

```
from nltk.stem import PorterStemmer, WordNetLemmatizer


stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
```

```
stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
lemmatized_tokens= [lemmatizer.lemmatize(token) for token in filtered_tokens]


print("Stemmed tokens : ",stemmed_tokens )
print("Lemmatized tokens : ",lemmatized_tokens )
```

```
Stemmed tokens :  ['sampl', 'text', 'text', 'analysi', 'tool', '.']
Lemmatized tokens :  ['sample', 'text', 'text', 'analysis', 'tool', '.']
```

Analyze the text

```
from nltk.probability import FreqDist


freq_dist = FreqDist(lemmatized_tokens)
print(freq_dist.most_common())
```

```
[('text', 2), ('sample', 1), ('analysis', 1), ('tool', 1), ('.', 1)]
```

2.Paragraph = sentences + words + perfrom the parts of speech tagging and seperate the text into n phrases.

Seperate the text into words

```
from nltk.tokenize import word_tokenize


text = "The dog sits on the porch."


#perform the tokenization
tokenized_text= word_tokenize(text)


print(tokenized_text)
```

```
['The', 'dog', 'sits', 'on', 'the', 'porch', '.']
```

```
#Import the tokenizer
from nltk.tokenize import RegexpTokenizer


# define the tokenizer parameters
tokenizer = RegexpTokenizer("[\w']+")
# this is to seperate using the empty-spaces(blank-spaces)


tokenized_text= tokenizer.tokenize(text)
print(tokenized_text)
```

```
['The', 'dog', 'sits', 'on', 'the', 'porch']
```

How to seperate the text into sentences

```
from nltk.tokenize import sent_tokenize


# Define example text data
sentences = """Machine Learning can usually be divided into classic Machine Learning and Deep Learning. Classic Machine Learning is rel



tokenized_sentences = sent_tokenize(sentences)
print(tokenized_sentences)
```

```
['Machine Learning can usually be divided into classic Machine Learning and Deep Learning.', 'Classic Machine Learning is relatively
```

Seperate text into phrases

```
from nltk import ngrams


n=2

bigrams = ngrams(tokenized_text,n)

bigram_list= []
for phrase in bigrams:
  bigram_list.append(phrase)


print(bigram_list)
```

```
[('The', 'dog'), ('dog', 'sits'), ('sits', 'on'), ('on', 'the'), ('the', 'porch')]
```

How to perform the syntactical anlaysis using nltk.

```
from nltk import pos_tag


# Create an example list of strings
# that represents a tokenized sentence
text_to_tag = ['Life', 'is', 'what', 'happens', 'when', 'you', 'are', 'busy', 'making', 'other', 'plans']



nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
# Perform POS-tagging
tagged_text = pos_tag(text_to_tag)



print(tagged_text)
```

```
[('Life', 'NNP'), ('is', 'VBZ'), ('what', 'WP'), ('happens', 'VBZ'), ('when', 'WRB'), ('you', 'PRP'), ('are', 'VBP'), ('busy', 'JJ'
```

Perform the Morphological Analysis using nltk.

```
# Import the stemmers
from nltk.stem import PorterStemmer
from nltk.stem.snowball import SnowballStemmer



# Define the stemmers we will use
porter_stemmer = PorterStemmer()
snowball_stemmer = SnowballStemmer(language='english')



# Create an empty list we will populate with words
# stemmed with the PorterStemmer algorithm
stemmed_text_porter_stemmer = []

# Stem the words using the PorterStemmer algorithm
for word in text_to_tag:
    stemmed_word = porter_stemmer.stem(word)
    stemmed_text_porter_stemmer.append(stemmed_word)



print(stemmed_word)
```

```
plan
```

```
print(stemmed_text_porter_stemmer)
```

```
['life', 'is', 'what', 'happen', 'when', 'you', 'are', 'busi', 'make', 'other', 'plan']
```

```
# Create an empty list we will populate with words
# stemmed with the SnowballStemmer algorithm
stemmed_text_snowball_stemmer = []

# Stem the words using the SnowballStemmer algorithm
for word in text_to_tag:
    stemmed_word = snowball_stemmer.stem(word)
    stemmed_text_snowball_stemmer.append(stemmed_word)
```

How to do Lemmatization

The way lemmatization works is actually quite simple. Instead of removing parts from words, it will take a look at the word and its POS tag and, based on that information, pick a version of that word from the WordNet dictionary. These dictionary word forms are called lemmas. In practice, this means that, as long as we perform POS tagging before lemmatization, the results we get from lemmatization will be better than the results we'd get from stemming. Theoretically, we can do lemmatization without performing POS tagging, though this is not recommended. If we don't supply the lemmatizer with a POS tag for a word, its default behavior is to treat the word as a noun. This means that lemmatizing a word without entering its POS tag will often lead to a word being lemmatized incorrectly. Let's demonstrate how we lemmatize text in NLTK.

```
# Import what we need
from nltk.corpus import wordnet

# Create function that converts tags
# from the Treebank format to the WordNet format
def convert_pos_tags(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

```
# Import what we need
from nltk.stem import WordNetLemmatizer

# Create the lemmatizer
lemmatizer = WordNetLemmatizer()
```

```
# Create an empty list where we will store lemmatized words
lemmatized_text_wordnet = []

# Convert POS tags
# and lemmatize using the WordNet algorithm
for word, tag in tagged_text:
    wordnet_tag = convert_pos_tags(tag)
    if wordnet_tag is None:
        lemmatized_word = lemmatizer.lemmatize(word)
    else:
        lemmatized_word = lemmatizer.lemmatize(word, pos=wordnet_tag)
    lemmatized_text_wordnet.append(lemmatized_word)
```

```
lemmatized_text_wordnet
```

```
['Life',
 'be',
 'what',
 'happen',
 'when',
 'you',
 'be',
 'busy',
 'make',
 'other',
 'plan']
```