Lab 3 - Tokenization

Tokenize the following sentences

In [2]:
```
sentence = "How to change payment method and payment frequency"
```

In [3]:
```
#Enter you code here
sentence = "How to change payment method and payment frequency"
tokens = sentence.split()
print(tokens)
```

```
['How', 'to', 'change', 'payment', 'method', 'and', 'payment', 'frequency']
```

In [6]:
```
sentence = "Who would have thought that computer programs would be analyzing human
```

In [8]:
```
#Enter you code here
sentence = "Who would have thought that computer programs would be analyzing human
tokens = sentence.split()
print(tokens)
```

```
['Who', 'would', 'have', 'thought', 'that', 'computer', 'programs', 'would', 'be',
'analyzing', 'human', 'sentiments']
```

import nltk and import word_tokenize from nltk.tokenize

In [12]:
```
import nltk
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\William\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\William\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[12]: True

import nltk and import word_tokenize from nltk.tokenize

In [14]:
```
import nltk
from nltk.tokenize import word_tokenize
```

Using word_tokenize, tokenize the following text.
hint: assing the result to a variable called tokens, then print(tokens)

In [19]:
```
text = "Who would have thought that computer programs would be analyzing human sent
```

In [103…
```
text = "Who would have thought that computer programs would be analyzing human sent


tokens = word_tokenize(text)


print(tokens)
```

```
['Who', 'would', 'have', 'thought', 'that', 'computer', 'programs', 'would', 'be',
'analyzing', 'human', 'sentiments']
```

import stopwords from nltk.corpus

In [101…
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


text = "Who would have thought that computer programs would be analyzing human sent


tokens = word_tokenize(text)


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


filtered_tokens = [word for word in tokens if word.lower() not in stop_words]


print(filtered_tokens)
```

```
['would', 'thought', 'computer', 'programs', 'would', 'analyzing', 'human', 'sentime
nts']
```

download the Englist stopwords from nltk and assign them to the variable called: stopwords

In [99]:
```
import nltk
from nltk.corpus import stopwords


stopwords = set(stopwords.words('english'))
```

print the stopwords

In [45]:
```
print(stopwords)
```

```
{"it's", 'we', "mustn't", 'being', 'she', 'they', 'having', 'for', 'shan', 'how', 'b
efore', 'has', "wasn't", 'y', 'in', "don't", 'was', 'am', 'as', 'now', "she's", 't
o', 'ain', 'here', 'can', "you'll", 'this', 'i', 'because', 'so', "mightn't", 'won',
'did', 'under', "didn't", "wouldn't", 'shouldn', 'very', 'than', 'doesn', 'when', 'i
ts', 've', 'themselves', 'over', 'are', 'most', 'against', 's', 'further', 'll', 'sh
ould', "that'll", 'only', 'ourselves', 'these', 'of', 'is', 'where', 'down', 'thos
e', 'by', 'below', 'doing', 'your', "aren't", 't', 'and', 'why', 'while', 'whom', 'y
ours', 'he', 'them', 'm', 'or', "you're", 'her', 'each', 'be', 'few', 'couldn', 'm
e', 'aren', 'no', 'you', 'herself', 'been', 'were', 'during', 'their', 'do', 'some',
'a', 'own', 'just', 'yourself', 'again', "won't", 'theirs', "haven't", 'out', "does
n't", 'there', 'nor', 'hers', 'don', "hasn't", 'from', 'such', 'which', 'his', 'othe
r', 'itself', 'himself', 'once', 'the', 'more', 'but', 'into', 'above', 'both', 'a
n', 'weren', 'o', 'didn', 'd', "shouldn't", 'all', "you've", 'hasn', 'isn', 'on', "n
eedn't", 're', "isn't", 'haven', 'about', 'ours', 'it', 'our', 'with', 'ma', 'him',
'will', 'same', "should've", 'off', 'have', 'needn', 'then', 'who', 'up', 'any', 'th
rough', "couldn't", 'hadn', 'mightn', 'if', 'until', 'what', 'between', 'after', "ha
dn't", 'myself', 'my', 'yourselves', 'had', 'wasn', 'wouldn', "shan't", 'that', 'doe
s', "you'd", "weren't", 'at', 'mustn', 'too', 'not'}
```

print the content of tokens again

In [97]:
```
import nltk
from nltk.tokenize import word_tokenize


text = "Who would have thought that computer programs would be analyzing human sent


tokens = word_tokenize(text)


print(tokens)
```

```
['Who', 'would', 'have', 'thought', 'that', 'computer', 'programs', 'would', 'be',
'analyzing', 'human', 'sentiments']
```

print only the words that are not tokens

In [95]:
```
import nltk
from nltk.tokenize import word_tokenize


text = "Who would have thought that computer programs would be analyzing human sent


tokens = word_tokenize(text)


words_only = [word for word in tokens if word.isalpha()]


print(words_only)
```

```
['Who', 'would', 'have', 'thought', 'that', 'computer', 'programs', 'would', 'be',
'analyzing', 'human', 'sentiments']
```

import * from nltk.stem.porter (from nltk.stem.porter import *)

In [55]:
```
from nltk.stem.porter import PorterStemmer
```

import WordNetLemmatizer from from nltk.stem

In [57]:
```
from nltk.stem import WordNetLemmatizer
```

complete the block of code below - (one line of code)

In [59]:
```
text = "Who would have thought that computer programs would be analyzing human sent

### tokenize the text and assign the result to tokens. hint: use word_tokenize

# your code goes here (one line)
tokens = word_tokenize(text)
# end of your code

lemmatizer = WordNetLemmatizer()
tokens=[lemmatizer.lemmatize(word) for word in tokens]
```

print tokens

In [61]:
```
print(tokens)
```

```
['Who', 'would', 'have', 'thought', 'that', 'computer', 'program', 'would', 'be', 'a
nalyzing', 'human', 'sentiment']
```

import RegexpTokenizer from nltk.tokenize

In [63]:
```
from nltk.tokenize import RegexpTokenizer
```

tokenize the followign text using RegexpTokenizer
include every word, number, symbols, and white spaces

In [66]:
```
text = "Who would have thought that computer programs would be analyzing human sent

## your code goes here (1 line of code)
tokenizer = RegexpTokenizer(r'\S+')
## end of your code
tokenizer.tokenize(text)
```

Out[66]:
```
['Who',
 'would',
 'have',
 'thought',
 'that',
 'computer',
 'programs',
 'would',
 'be',
 'analyzing',
 'human',
 'sentiments?',
 "Isn't",
 'is',
 'amazin?']
```

import BlanklineTokenizer from nltk.tokenize

In [68]:
```
from nltk.tokenize import BlanklineTokenizer
```

tokenize the followign text using BlanklineTokenizer

In [71]:
```
text = "Who would have thought that computer programs would be analyzing human sent

## your code goes here (1 line of code)
segments = text.split('\n\n')
## end of your code

tokenizer.tokenize(text)
```

Out[71]:
```
['Who',
 'would',
 'have',
 'thought',
 'that',
 'computer',
 'programs',
 'would',
 'be',
 'analyzing',
 'human',
 'sentiments?',
 "Isn't",
 'is',
 'amazin?',
 'I',
 'hope',
 'you',
 'are',
 'enjoying',
 'this',
 'class.',
 'The',
 'price',
 'of',
 'our',
 'book',
 'is',
 '$42']
```

import WordPunctTokenizer from nltk.tokenize

In [73]:
```
from nltk.tokenize import WordPunctTokenizer
```

tokenize the followign text using WordPunctTokenizer

In [75]:
```
text = "Who would have thought that computer programs would be analyzing human sent

## your code goes here (1 line of code)
tokenizer = WordPunctTokenizer()
## end of your code

tokenizer.tokenize(text)
```

Out[75]:
```
['Who',
 'would',
 'have',
 'thought',
 'that',
 'computer',
 'programs',
 'would',
 'be',
 'analyzing',
 'human',
 'sentiments',
 '?',
 'Isn',
 "'",
 't',
 'is',
 'amazin',
 '?',
 'I',
 'hope',
 'you',
 'are',
 'enjoying',
 'this',
 'class',
 '.',
 'The',
 'price',
 'of',
 'our',
 'book',
 'is',
 '$',
 '42']
```

Import some libraries and remove warnings:

gensim, pandas, matplotlib
from gensim.models.phrases import Phraser, Phrases
from gensim.models.word2vec import Word2Vec
add %matplotlib inline
remove warnings

In [93]:
```
import warnings
warnings.filterwarnings('ignore')

import gensim
import pandas as pd
import matplotlib.pyplot as plt
from gensim.models.phrases import Phraser, Phrases
from gensim.models.word2vec import Word2Vec
```

Add some libraries for Natural Language Preprocessing and Word Vectors

In [91]:
```
import nltk
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import *
nltk.download('gutenberg')
nltk.download('punkt')
nltk.download('stopwords')
import string
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\William\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\William\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\William\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

import gutenberg from nltk.corpus

In [83]:
```
from nltk.corpus import gutenberg
```

print the length of the gutenberg.fileids

In [89]:
```
from nltk.corpus import gutenberg


file_ids = gutenberg.fileids()


print(len(file_ids))
```

```
18
```

display the fileids with extension txt

In [109…
```
txt_file_ids = [file_id for file_id in file_ids if file_id.endswith('.txt')]
```

how many words are in gutenberg?

In [111…
```
total_words = sum(len(gutenberg.words(file_id)) for file_id in file_ids)

print(f"Total number of words in the Gutenberg corpus: {total_words}")
```

```
Total number of words in the Gutenberg corpus: 2621613
```

In [113…
```
gberg_sent_tokens = sent_tokenize(gutenberg.raw())
```

Display the tokenized text from 10 to 15

In [115…
```
from nltk.corpus import gutenberg
from nltk.tokenize import sent_tokenize


gberg_raw_text = gutenberg.raw()
gberg_sent_tokens = sent_tokenize(gberg_raw_text)


for i in range(10, 16):
    print(f"Sentence {i}: {gberg_sent_tokens[i]}")
```

```
Sentence 10: It was on the wedding-day
of this beloved friend that Emma first sat in mournful thought
of any continuance.
Sentence 11: The wedding over, and the bride-people gone,
her father and herself were left to dine together, with no prospect
of a third to cheer a long evening.
Sentence 12: Her father composed himself
to sleep after dinner, as usual, and she had then only to sit
and think of what she had lost.
Sentence 13: The event had every promise of happiness for her friend.
Sentence 14: Mr. Weston
was a man of unexceptionable character, easy fortune, suitable age,
and pleasant manners; and there was some satisfaction in considering
with what self-denying, generous friendship she had always wished
and promoted the match; but it was a black morning's work for her.
Sentence 15: The want of Miss Taylor would be felt every hour of every day.
```

Display the first sentence

In [118…
```
gberg_sent_tokens[1]
```

Out[118…
```
"She was the youngest of the two daughters of a most affectionate,\nindulgent fath
er; and had, in consequence of her sister's marriage,\nbeen mistress of his house
from a very early period."
```

Tokenize the first sentence. Hint: use word_tokenize

In [125…
```
gberg_raw_text = gutenberg.raw()
gberg_sent_tokens = sent_tokenize(gberg_raw_text)


first_sentence = gberg_sent_tokens[0]


first_sentence_tokens = word_tokenize(first_sentence)


print(first_sentence_tokens)
```

```
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', 'VOLUME', 'I', 'CHAPTER', 'I', 'E
mma', 'Woodhouse', ',', 'handsome', ',', 'clever', ',', 'and', 'rich', ',', 'with',
'a', 'comfortable', 'home', 'and', 'happy', 'disposition', ',', 'seemed', 'to', 'uni
te', 'some', 'of', 'the', 'best', 'blessings', 'of', 'existence', ';', 'and', 'had',
'lived', 'nearly', 'twenty-one', 'years', 'in', 'the', 'world', 'with', 'very', 'lit
tle', 'to', 'distress', 'or', 'vex', 'her', '.']
```

diplay the 20th tokenized word of the first sentence

In [130…

```
first_sentence = gberg_sent_tokens[0]


first_sentence_tokens = word_tokenize(first_sentence)

if len(first_sentence_tokens) >= 20:
    print(f"The 20th tokenized word is: {first_sentence_tokens[19]}")
else:
    print("The first sentence does not have 20 words.")
```

```
The 20th tokenized word is: rich
```

Create a new model using gensim.models.Word2Vec.load and the file: 'clean_gutenberg_model.w2v'

In [132…

```
from gensim.models import Word2Vec


model = Word2Vec.load('clean_gutenberg_model.w2v')


print(f"Model vocabulary size: {len(model.wv.index_to_key)}")
print(f"Example words in the model: {model.wv.index_to_key[:10]}")
```

```
Model vocabulary size: 10329
Example words in the model: ['the', 'and', 'of', 'to', 'a', 'in', 'i', 'that', 'he',
'it']
```

Explore the model for the word 'father' hint: use model.vw

In [140…

```
if 'father' in model.wv:

    father_vector = model.wv['father']


    print(f"Vector representation of 'father':\n{father_vector}")


    similar_words = model.wv.most_similar('father', topn=10)


    print("\nMost similar words to 'father':")
    for word, similarity in similar_words:
        print(f"{word}: {similarity}")
```

```
else:
    print("'father' is not in the model's vocabulary.")
```

```
Vector representation of 'father':
[ 0.4562079   0.08332714 -0.50139236  0.6157958  -0.09423487  0.11964868
 -0.3001857  -0.00971249  0.0105763  -0.18398604  0.13505384  0.11026111
  0.430888    0.35198018  0.09762941  0.4074535   0.07121532  0.09248074
  0.18866564 -0.23999684 -0.12835982  0.27435288  0.15070201 -0.35054013
 -0.1263971   0.62390506  0.02754452 -0.20921491  0.305323   -0.8932203
  0.00809613  0.04333829 -0.11906347 -0.13334717  0.35294238  0.10896434
 -0.06387077  0.11086144  0.00103075  0.07404359  0.3645398  -0.30586603
  0.11004212 -0.11104885 -0.22725084 -0.3267144  -0.27682877 -0.05846506
 -0.31954205  0.3055387   0.18186492 -0.34653825 -0.00137501  0.23123048
  0.22988743  0.38822395  0.37932292 -0.08853725 -0.4913139  -0.31676406
  0.12621118  0.6862584   0.06878584  0.51259506]

Most similar words to 'father':
mother: 0.8257375359535217
brother: 0.7275018692016602
sister: 0.7177823781967163
curseth: 0.7117124795913696
wife: 0.7076548337936401
dinah: 0.687444806098938
dearly: 0.682124137878418
uncle: 0.6820593476295471
daughter: 0.68001389503479
master: 0.6691791415214539
```

display five similar words to the word 'father'

In [144…

```
if 'father' in model.wv:

    similar_words = model.wv.most_similar('father', topn=5)


    print("Five most similar words to 'father':")
    for word, similarity in similar_words:
        print(f"{word}: {similarity}")
else:
    print("'father' is not in the model's vocabulary.")
```

```
Five most similar words to 'father':
mother: 0.8257375359535217
brother: 0.7275018692016602
sister: 0.7177823781967163
curseth: 0.7117124795913696
wife: 0.7076548337936401
```

Do you agree with the result?

Yes, I do agree with the result.

using your model and the "doesn_match" function, find out which of the following words doesn't match

"sister brother child cousing aunt"

In [148…
```
words = ["sister", "brother", "child", "cousin", "aunt"]

odd_one_out = model.wv.doesnt_match(words)

print(f"The word that doesn't match is: {odd_one_out}")
```

```
The word that doesn't match is: child
```

Do you agree with the result and why?

Yes, and the reason why is the result of the doesnt_match function should reflect the word that is least contextually related to the others based on the model's training data. For instance, "cousin" might be less related to immediate family terms like "sister" and "brother." If the result aligns with expected semantic relationships, it's likely accurate; otherwise, it may reflect the model's training data nuances.

Visualize

Using pandas, load the file clean_gutenberg_tsne.csv to a dataframe called coords_df

In [153…
```
import pandas as pd

coords_df = pd.read_csv('clean_gutenberg_tsne.csv')
```

Display the head

In [155…
```
print(coords_df.head())
```

```
           x          y   token
0  62.494060   8.023034    emma
1   8.142986  33.342200      by
2  62.507140  10.078477    jane
3  12.477635  17.998343  volume
4  25.736960  30.876250       i
```

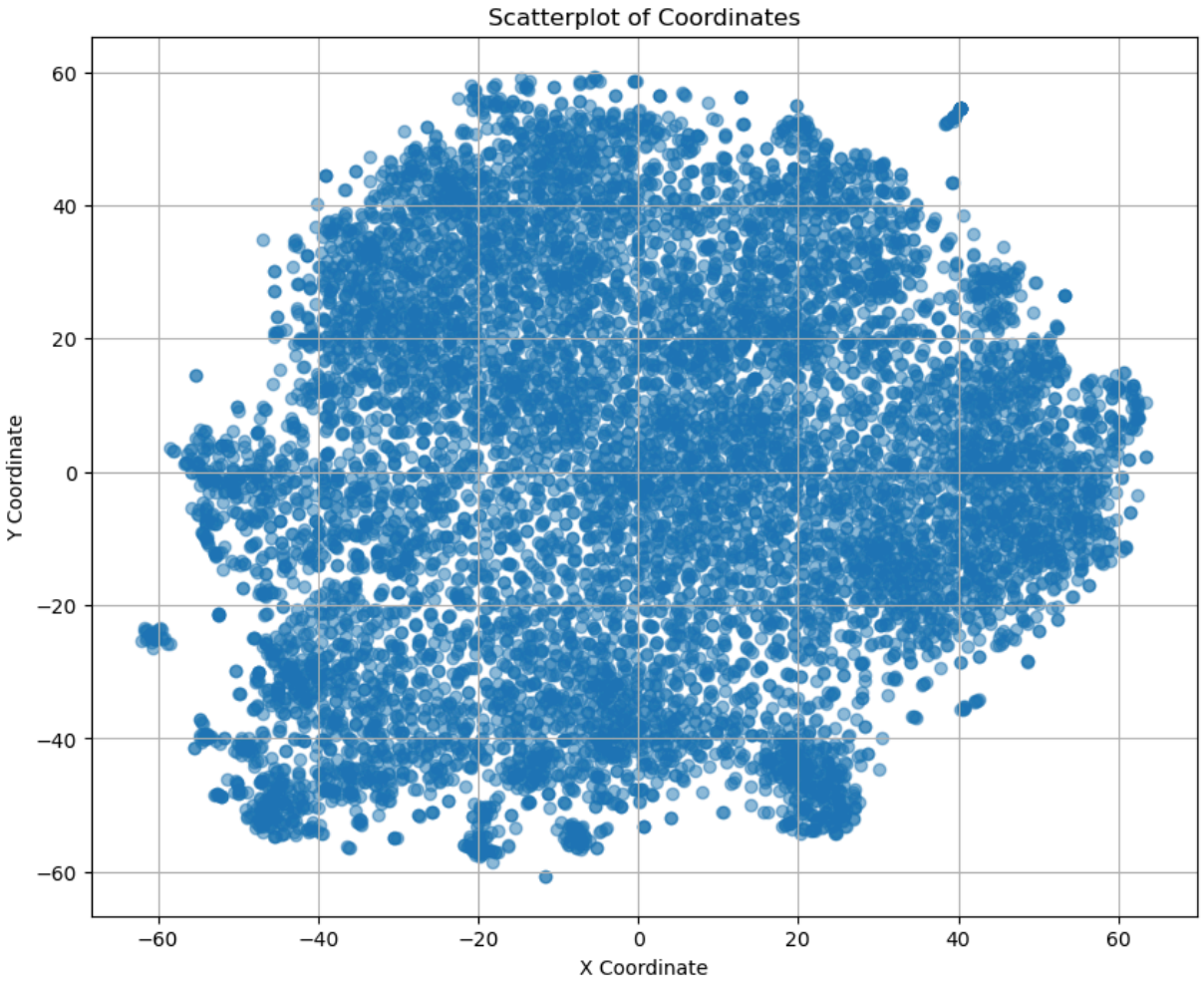
Create a scatterplot of coords_df

In [157…
```
import pandas as pd
import matplotlib.pyplot as plt


coords_df = pd.read_csv('clean_gutenberg_tsne.csv')


plt.figure(figsize=(10, 8))
plt.scatter(coords_df['x'], coords_df['y'], alpha=0.5)
plt.title('Scatterplot of Coordinates')
```

```
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')
plt.grid(True)
plt.show()
```



In [ ]: