In [ ]:

```
#Q.1
```

In [1]:

```
from nltk.data import load
from nltk.tokenize.treebank import TreebankWordTokenizer
```

In [5]:

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

In [11]:

```
import re
import pandas as pd
import nltk
from nltk.tokenize import WordPunctTokenizer
nltk.download('stopwords')
from nltk.corpus import stopwords
# needed for nltk.pos_tag function nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\I5262\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\I5262\AppData\Roaming\nltk_data...
```

In [13]:

```
# load text
filename = 'novel.txt'
file = open(filename, 'rt')
text = file.read()
print(text)
file.close()
```

```
Gregor then turned to look out the window at the dull weather.
Drops of rain could be heard hitting the pane, which made him feel
quite sad.  "How about if I sleep a little bit longer and forget all
this nonsense", he thought, but that was something he was unable to
do because he was used to sleeping on his right, and in his present
state couldn't get into that position.  However hard he threw
himself onto his right, he always rolled back to where he was.  He
must have tried it a hundred times, shut his eyes so that he
wouldn't have to look at the floundering legs, and only stopped when
he began to feel a mild, dull pain there that he had never felt
before.

"Oh, God", he thought, "what a strenuous career it is that I've
chosen! Travelling day in and day out.  Doing business like this
takes much more effort than doing your own business at home, and on
top of that there's the curse of travelling, worries about making
train connections, bad and irregular food, contact with different
people all the time so that you can never get to know anyone or
become friendly with them.  It can all go to Hell!"  He felt a
slight itch up on his belly; pushed himself slowly up on his back
```

In [14]:

```
word_punct_token = WordPunctTokenizer().tokenize(text)
```

In [15]:

```
clean_token=[]
for token in word_punct_token:
    token = token.lower()
    # remove any value that are not alphabetical
    new_token = re.sub(r'[^a-zA-Z]+', '', token)
    # remove empty value and single character value
    if new_token != "" and len(new_token) >= 2:
        vowels=len([v for v in new_token if v in "aeiou"])
        if vowels != 0: # remove line that only contains consonants
            clean_token.append(new_token)
```

In [16]:

```
# Get the list of stop words
stop_words = stopwords.words('english')
# add new stopwords to the list
stop_words.extend(["could","though","would","also","many",'much'])
print(stop_words)
# Remove the stopwords from the list of tokens
tokens = [x for x in clean_token if x not in stop_words]
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "yo
u're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yoursel
ves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'hers
elf', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs',
'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'bein
g', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while',
'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from',
'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'furthe
r', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al
l', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'suc
h', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'ver
y', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should'v
e", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't",
'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "had
n't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'might
n', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't",
'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "wo
n't", 'wouldn', "wouldn't", 'could', 'though', 'would', 'also', 'many',
'much']
```

In [19]:

```
from nltk.tokenize import sent_tokenize, word_tokenize
```

In [20]:

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

In [21]:

```
filtered_list = []
```

In [23]:

```
worf_quote = "Sir, I protest. I am not a merry man!"
```

In [24]:

```
words_in_quote = word_tokenize(worf_quote)
words_in_quote
```

Out[24]:

```
['Sir', ',', 'I', 'protest', '.', 'I', 'am', 'not', 'a', 'merry', 'man',
'!']
```

In [25]:

```
for word in words_in_quote:
    if word.casefold() not in stop_words:
        filtered_list.append(word)
```

In [26]:

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

In [29]:

```
filtered_list = [
word for word in words_in_quote if word.casefold() not in stop_words
]
```

In [30]:

```
filtered_list
```

Out[30]:

```
['Sir', ',', 'protest', '.', 'merry', 'man', '!']
```

In [31]:

```
from nltk.stem import PorterStemmer
```

In [32]:

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

In [33]:

```
stemmer = PorterStemmer()
```

In [34]:

```
string_for_stemming = """
The crew of the USS Discovery discovered many discoveries.
Discovering is what explorers do."""
```

In [35]:

```
words = word_tokenize(string_for_stemming)
```

In [36]:

```
words
```

Out[36]:

```
['The',
 'crew',
 'of',
 'the',
 'USS',
 'Discovery',
 'discovered',
 'many',
 'discoveries',
 '.',
 'Discovering',
 'is',
 'what',
 'explorers',
 'do',
 '.']
```

In [37]:

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

In [38]:

```
stemmed_words
```

Out[38]:

```
['the',
 'crew',
 'of',
 'the',
 'uss',
 'discoveri',
 'discov',
 'mani',
 'discoveri',
 '.',
 'discov',
 'is',
 'what',
 'explor',
 'do',
 '.']
```

In [39]:

```
#Q.2
```

In [42]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [
          'Umapavan Submitted this Assignment',
          '3rd Subject on data Science  Courses for data Certification',
          'Parallel Processing Large File in Python',
          'NLP is Having Good Raise You Must Know For data science',
 ]

vectorizer = TfidfVectorizer()

# TD-IDF Matrix
X = vectorizer.fit_transform(corpus)

# extracting feature names
tfidf_tokens = vectorizer.get_feature_names_out()
```

In [43]:

```
import pandas as pd

result = pd.DataFrame(
    data=X.toarray(),
    index=["Doc1", "Doc2", "Doc3", "Doc4"],
    columns=tfidf_tokens
)

result
```

Out[43]:

	3rd	assignment	certification	courses	data	file	for	good
Doc1	0.000000	0.5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Doc2	0.338457	0.0	0.338457	0.338457	0.533687	0.000000	0.266844	0.000000
Doc3	0.000000	0.0	0.000000	0.000000	0.000000	0.408248	0.000000	0.000000
Doc4	0.000000	0.0	0.000000	0.000000	0.251021	0.000000	0.251021	0.318388

4 rows × 26 columns