Name: Aadarsha Chapagain

Student Id: c0825975

In [4]:
```
from sklearn.feature_extraction.text import CountVectorizer
text = "Hello everyone. Welcome to Word Tokenization"
vectorizer = CountVectorizer()
tokenizer = vectorizer.build_tokenizer()
tokens = tokenizer(text)
print(tokens)
```

```
['Hello', 'everyone', 'Welcome', 'to', 'Word', 'Tokenization']
```

In [5]:
```
from sklearn.feature_extraction import _stop_words
StopWords = list(_stop_words.ENGLISH_STOP_WORDS)
print(StopWords)
```

```
['empty', 'whereafter', 'one', 'over', 'etc', 'system', 'ie', 'bottom', 'front', 'anywhere', 'myself', 'nevertheless', 'a
lthough', 'amongst', 'with', 'so', 'whither', 'bill', 'everyone', 'made', 'serious', 'hereafter', 'somehow', 'the', 'ther
e', 'thereupon', 'hence', 'anyway', 'hers', 'both', 'where', 'since', 'fill', 'not', 'most', 'others', 'eight', 'but', 'i
tself', 'has', 'almost', 'you', 'name', 'at', 'thus', 'under', 'whatever', 'beside', 'found', 'fifty', 'hasnt', 'do', 'd
e', 'couldnt', 'until', 'again', 'two', 'as', 'down', 'thick', 'meanwhile', 'through', 'everything', 'else', 'less', 'ge
t', 'seem', 'was', 'sometimes', 'should', 'her', 'part', 'indeed', 'upon', 'all', 'already', 'seeming', 'those', 'somewhe
re', 'our', 'sixty', 'nine', 'thereby', 'and', 'third', 'therein', 'mostly', 'what', 'for', 'per', 'throughout', 'becaus
e', 'than', 'must', 'from', 'please', 'nowhere', 'cannot', 'further', 'that', 'your', 'afterwards', 'anything', 'or', 'ca
n', 'together', 'full', 'his', 'nobody', 'even', 'latterly', 'next', 'wherein', 'it', 'go', 'thence', 'ever', 'find', 'n
o', 'among', 'have', 'someone', 'we', 'never', 'herself', 'by', 'detail', 'its', 'put', 'to', 'on', 'back', 'an', 'former
ly', 'between', 'neither', 'last', 'still', 'none', 'anyhow', 'a', 'becomes', 'however', 'during', 'many', 'then', 'excep
t', 'though', 'she', 'once', 'onto', 'interest', 'fire', 'every', 'inc', 'keep', 'thin', 'also', 'another', 'mill', 'no
w', 'while', 'were', 'mine', 'few', 'my', 'five', 'behind', 'former', 'ltd', 'within', 'nothing', 'toward', 'themselves',
'side', 'these', 'become', 'whereby', 'became', 'four', 'about', 'eleven', 'they', 'either', 'any', 'would', 'sometime',
'whose', 'above', 'done', 'call', 'show', 'below', 'here', 'therefore', 'is', 'only', 'twenty', 'seems', 'me', 'seemed',
'wherever', 'beforehand', 'being', 'top', 'out', 'along', 'too', 'something', 'eg', 'if', 'forty', 'hereby', 'yet', 'whet
her', 'ours', 'twelve', 'before', 'perhaps', 'noone', 'nor', 'besides', 'will', 'be', 'off', 'he', 'whence', 'of', 'migh
t', 'give', 'otherwise', 'thereafter', 'why', 'are', 'had', 'been', 'well', 'namely', 'fifteen', 'rather', 'across', 'thr
ee', 'whenever', 'yourself', 'himself', 'see', 'ten', 'other', 'more', 'due', 'around', 'each', 'latter', 'amount', 'alon
e', 'sincere', 'such', 'same', 'several', 'them', 'whereupon', 're', 'who', 'yourselves', 'enough', 'very', 'could', 'any
one', 'move', 'co', 'him', 'up', 'often', 'i', 'whoever', 'amoungst', 'always', 'whole', 'may', 'whom', 'some', 'hereupo
n', 'into', 'moreover', 'whereas', 'cant', 'beyond', 'herein', 'much', 'how', 'six', 'elsewhere', 'describe', 'after', 't
```

```
owards', 'un', 'cry', 'take', 'becoming', 'hundred', 'ourselves', 'own', 'least', 'via', 'us', 'first', 'in', 'thru', 'wi
thout', 'con', 'everywhere', 'their', 'this', 'when', 'yours', 'which', 'am', 'against']
```

In [6]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import _stop_words
text = "There was no chance for any character development they were busy"
StopWords = list(_stop_words.ENGLISH_STOP_WORDS)
vectorizer = CountVectorizer()
tokenizer = vectorizer.build_tokenizer()
words = tokenizer(text)
result = []
for w in words:
    if w not in StopWords:
        result.append(w)
print(words)
print(result)
```

```
['There', 'was', 'no', 'chance', 'for', 'any', 'character', 'development', 'they', 'were', 'busy']
['There', 'chance', 'character', 'development', 'busy']
```

In [7]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import _stop_words
text = ["this movie made it into one of my top ten most awful movies",
        "there wasn't a continuous minute where there wasn't a fight with one monster or another",
        "there was no chance for any character development they were too busy running from one sword fight to another",
        "i had no emotional attachment except to the big bad machine that wanted to destroy them"]
StopWords = list(_stop_words.ENGLISH_STOP_WORDS)
vectorizer = CountVectorizer(stop_words=StopWords)
vectorizer.fit(text)
print(len(vectorizer.vocabulary_))
print(vectorizer.vocabulary_)
vectors = vectorizer.transform(text)
print(vectors.toarray())
```

```
21
{'movie': 15, 'awful': 1, 'movies': 16, 'wasn': 20, 'continuous': 7, 'minute': 13, 'fight': 11, 'monster': 14, 'chance':
5, 'character': 6, 'development': 9, 'busy': 4, 'running': 17, 'sword': 18, 'emotional': 10, 'attachment': 0, 'big': 3,
'bad': 2, 'machine': 12, 'wanted': 19, 'destroy': 8}
[[0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 2]
 [0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 1 1 0 0]
 [1 0 1 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0]]
```

In [8]:

```
# import required module
from sklearn.feature_extraction.text import TfidfVectorizer
data = ['Natural Language Processing','Natural data','NLP','Data Processing']
tfidf = TfidfVectorizer()
# get tf-df values
result = tfidf.fit_transform(data)
# get idf values
print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names(), tfidf.idf_):
    print(ele1, ':', ele2)
```

```
idf values:
data : 1.5108256237659907
language : 1.916290731874155
natural : 1.5108256237659907
nlp : 1.916290731874155
processing : 1.5108256237659907
```

In [9]:

```
# get indexing
print('\nWord indexes:')
print(tfidf.vocabulary_)

print('\ntf-idf values in matrix form:')
print(result.toarray())
```

```
Word indexes:
{'natural': 2, 'language': 1, 'processing': 4, 'data': 0, 'nlp': 3}

tf-idf values in matrix form:
[[0.         0.66767854 0.52640543 0.         0.52640543]
 [0.70710678 0.         0.70710678 0.         0.        ]
 [0.         0.         0.         1.         0.        ]
 [0.70710678 0.         0.         0.         0.70710678]]
```

In [10]:

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
ps =  PorterStemmer()
ls = LancasterStemmer()
words = ["program", "programs", "programmer", "programming", "destabilize"]
print('Porter Stemmer : ')
for w in words:
  print(w," : ",ps.stem(w))
print('\n Lancaster Stemmer : ')
```

```
for w in words:
  print(w," : ",ls.stem(w))
```

```
Porter Stemmer :
program  :  program
programs  :  program
programmer  :  programm
programming  :  program
destabilize  :  destabil

 Lancaster Stemmer :
program  :  program
programs  :  program
programmer  :  program
programming  :  program
destabilize  :  dest
```

In [11]:

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
analyzer = CountVectorizer().build_analyzer()
words = ["program", "programs", "programmer", "programming", "destabilize"]

def stemmed_words(doc):
    return (stemmer.stem(w) for w in analyzer(doc))

stem_vectorizer = CountVectorizer(analyzer=stemmed_words)
stem_vectorizer.fit_transform(words)
print(stem_vectorizer.get_feature_names())
```

```
['destabil', 'program', 'programm']
```

In [ ]: