Text Analytics
1. Extract Sample document and apply following document preprocessing methods:
Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
2. Create representation of document by calculating Term Frequency and Inverse Document
Frequency.

In [6]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk
from nltk.tokenize import *
from nltk.corpus import *
from nltk.stem import *
import re
```

In [8]:
```
file_path = 'doc.txt'
with open(file_path, 'r', encoding='utf-8') as f:
    raw_text = f.read()
    print(raw_text)
```

```
Text analytics assignment. This is a sample document to check freq
uency of text in document. Frequency determines the importance of
a word based on its occurence in the document. Words for frequency
testing: System, system, systeM , system. File, file, file, file.
docs docs. check,check,check.
```

In [10]:
```
nltk.download('all')
var1 = sent_tokenize(raw_text)
```

```
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /home/admin1/nltk_data...
[nltk_data]    |   Unzipping corpora/europarl_raw.zip.
[nltk_data]    | Downloading package extended_omw to
[nltk_data]    |     /home/admin1/nltk_data...
[nltk_data]    | Downloading package floresta to
[nltk_data]    |     /home/admin1/nltk_data...
[nltk_data]    |   Unzipping corpora/floresta.zip.
[nltk_data]    | Downloading package framenet_v15 to
[nltk_data]    |     /home/admin1/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v15.zip.
[nltk_data]    | Downloading package framenet_v17 to
[nltk_data]    |     /home/admin1/nltk_data...
[nltk_data]    |   Unzipping corpora/framenet_v17.zip.
[nltk_data]    | Downloading package gazetteers to
[nltk_data]    |     /home/admin1/nltk_data...
[nltk_data]    |   Unzipping corpora/gazetteers.zip.
[nltk_data]    | Downloading package genesis to
[nltk_data]    |     /home/admin1/nltk_data...
[nltk_data]    |   Unzipping corpora/genesis.zip.
```

In [11]:
```
print(var1)
```

```
['Text analytics assignment.', 'This is a sample document to check
frequency of text in document.', 'Frequency determines the importa
nce of a word based on its occurence in the document.', 'Words for
frequency testing: System, system, systeM , system.', 'File, file,
file, file.', 'docs docs.', 'check,check,check.']
```

In [12]:
```
var2= word_tokenize(raw_text)
```

In [14]:
```
print(var2)
```

```
['Text', 'analytics', 'assignment', '.', 'This', 'is', 'a', 'sampl
e', 'document', 'to', 'check', 'frequency', 'of', 'text', 'in', 'd
ocument', '.', 'Frequency', 'determines', 'the', 'importance', 'o
f', 'a', 'word', 'based', 'on', 'its', 'occurence', 'in', 'the',
'document', '.', 'Words', 'for', 'frequency', 'testing', ':', 'Sys
tem', ',', 'system', ',', 'systeM', ',', 'system', '.', 'File',
',', 'file', ',', 'file', ',', 'file', '.', 'docs', 'docs', '.',
'check', ',', 'check', ',', 'check', '.']
```

In [15]:
```
var3= set(stopwords.words('english'))
print(var3)
```

```
{"they'll", 'couldn', 'before', 'been', 'out', 'very', 'down', 'i
t', 'off', "we'd", 'who', "you've", 'does', 'or', 'there', 'then',
'up', 'no', 'any', 'itself', "shouldn't", 'few', 'over', 'each',
're', 'some', 'to', 'whom', 'yourselves', 'after', 'nor', 'here',
'needn', "they'd", 'aren', 'do', 'am', 'have', 'wouldn', "we'll",
'now', 'at', "they've", 'this', 'won', 'from', "i'd", 'was', 'wit
h', 'for', 'is', "aren't", "isn't", "it'll", 'are', 'ma', 'me', 't
hrough', "we're", 'hasn', "it's", 'themselves', 'below', 'furthe
r', 'didn', 'why', 'we', 'i', 'just', 's', 'be', 'how', "he'd", 'w
here', 'own', 'more', 'o', 'of', "should've", 'if', 'himself', 'in
to', 'having', 'shan', 'can', 'when', 'by', 'being', "you'll", 'wh
at', 'herself', "mightn't", 'those', 'not', "shan't", 'both', "the
y're", "hadn't", 'll', "mustn't", 'other', "that'll", 'you', 'as',
"haven't", 'had', "it'd", 'myself', 'because', "he'll", 'which',
"she's", 'hers', 'but', 'yours', 'under', "won't", 'him', 'weren',
'in', 'doing', 'between', 'mightn', 'her', 'his', 'same', "has
n't", 'he', 'ain', 'your', "couldn't", 'during', "didn't", 'an',
'isn', 'than', 'doesn', 'that', 'their', "i'm", 'its', 'shouldn',
've', 'on', "you'd", 'such', 'once', "wasn't", "weren't", 't', 'di
d', 'ourselves', 'd', 'these', "she'll", 'all', "doesn't", 'unti
l', 'so', 'a', "she'd", "don't", 'while', 'about', 'wasn', 'most',
'my', "wouldn't", "i'll", "you're", 'she', 'don', 'y', 'above', 'h
aven', 'only', 'and', 'the', 'has', "i've", 'them', "needn't", "h
e's", 'hadn', 'too', 'mustn', 'theirs', 'they', 'm', 'should', 'wi
ll', 'against', 'were', "we've", 'our', 'again', 'ours', 'yoursel
f'}
```

In [19]:
```
filtered_text=[]
tokens= word_tokenize(raw_text.lower())
for word in tokens:
  if word not in var3:
        filtered_text.append(word)
print(filtered_text)
```

```
['text', 'analytics', 'assignment', '.', 'sample', 'document', 'ch
eck', 'frequency', 'text', 'document', '.', 'frequency', 'determin
es', 'importance', 'word', 'based', 'occurence', 'document', '.',
'words', 'frequency', 'testing', ':', 'system', ',', 'system',
',', 'system', ',', 'system', '.', 'file', ',', 'file', ',', 'fil
e', ',', 'file', '.', 'docs', 'docs', '.', 'check', ',', 'check',
',', 'check', '.']
```

In [20]:
```
var = ["do", "did", "doing", "done"]
ps = PorterStemmer() # brings word to its root form
for w in var:
  root_word = ps.stem(w)
  print(root_word)
```

```
do
did
do
done
```

In [34]:
```
#POS Tagging
text = "doing reading swimming views viewing performance types typi

tokens = nltk.word_tokenize(text)

pos_tags = nltk.pos_tag(tokens)

print(pos_tags)
```

```
[('doing', 'VBG'), ('reading', 'VBG'), ('swimming', 'VBG'), ('view
s', 'NNS'), ('viewing', 'VBG'), ('performance', 'NN'), ('types',
'NNS'), ('typing', 'VBG')]
```

In [31]:
```
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

In [33]:
```
#Lemmetization
from nltk.corpus import wordnet

lemmatizer = WordNetLemmatizer()

for word, tag in pos_tags:
    wn_tag = get_wordnet_pos(tag)
    lemma = lemmatizer.lemmatize(word, pos=wn_tag)
    print(f"{word} ({tag}) → {lemma}")
```

```
doing (VBG) → do
reading (VBG) → read
swimming (VBG) → swim
views (NNS) → view
viewing (VBG) → view
performance (NN) → performance
types (NNS) → type
typing (VBG) → type
```

In [35]:
```
#TF , IDF
def calculate_tfIdf(raw_text):
    tokenizer = TfidfVectorizer()
    tf_matrix = tokenizer.fit_transform(raw_text)
    features_names = tokenizer.get_feature_names_out()
    return tf_matrix, features_names

document = [raw_text]
tf_matrix, feature_names = calculate_tfIdf(document)

print('TF-IDF')
print(feature_names, tf_matrix.toarray())
```

```
TF-IDF
['analytics' 'assignment' 'based' 'check' 'determines' 'docs' 'doc
ument'
 'file' 'for' 'frequency' 'importance' 'in' 'is' 'its' 'occurence'
'of'
 'on' 'sample' 'system' 'testing' 'text' 'the' 'this' 'to' 'word'
'words'] [[0.09901475 0.09901475 0.09901475 0.39605902 0.09901475
0.19802951
  0.29704426 0.39605902 0.09901475 0.29704426 0.09901475 0.1980295
1
  0.09901475 0.09901475 0.09901475 0.19802951 0.09901475 0.0990147
5
  0.39605902 0.09901475 0.19802951 0.19802951 0.09901475 0.0990147
5
  0.09901475 0.09901475]]
```