In [1]:

```
# Extract Sample document and apply following document preprocessing methods:
# Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
```

In [2]:

```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.8/site-p
ackages (3.7)
Requirement already satisfied: click in /usr/local/lib/python3.8/site-
packages (from nltk) (8.1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/site-p
ackages (from nltk) (4.64.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib64/pyt
hon3.8/site-packages (from nltk) (2022.4.24)
Requirement already satisfied: joblib in /usr/local/lib/python3.8/site
-packages (from nltk) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```
import nltk
nltk.download('punkt')
from nltk import word_tokenize, sent_tokenize
import pandas as pd
import sklearn as sk
import math
```

```
[nltk_data] Downloading package punkt to /home/TE/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [4]:

```
sent = "GeeksforGeeks is a great learning platform . It is one of the best for Comp
words = (word_tokenize(sent))
sentences = (sent_tokenize(sent))
```

In [5]:

```
print(words)
print(sentences)
```

```
['GeeksforGeeks', 'is', 'a', 'great', 'learning', 'platform', '.', 'I
t', 'is', 'one', 'of', 'the', 'best', 'for', 'Computer', 'Science', 's
tudents', '.', 'play', 'playing', 'played', 'communication']
['GeeksforGeeks is a great learning platform .', 'It is one of the bes
t for Computer Science students.', 'play playing played communicatio
n']
```

In [6]:

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

In [7]:

```
ps = PorterStemmer()
stem = []
for i in words:
    stem_word = ps.stem(i);
    stem.append(stem_word);

print(stem)
```

```
['geeksforgeek', 'is', 'a', 'great', 'learn', 'platform', '.', 'it',
'is', 'one', 'of', 'the', 'best', 'for', 'comput', 'scienc', 'studen
t', '.', 'play', 'play', 'play', 'commun']
```

In [8]:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lem = []
for i in words:
    lem_word = lemmatizer.lemmatize(i,'v');
    lem.append(lem_word);

print(lem)
```

```
['GeeksforGeeks', 'be', 'a', 'great', 'learn', 'platform', '.', 'It',
'be', 'one', 'of', 'the', 'best', 'for', 'Computer', 'Science', 'stude
nts', '.', 'play', 'play', 'play', 'communication']
```

In [9]:

```
from nltk import pos_tag
```

In [10]:

```
from nltk.corpus import stopwords
```

In [11]:

```
tags = pos_tag(words)
```

In [12]:

```
print(tags)
sw_nltk = stopwords.words('english')
```

```
[('GeeksforGeeks', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('great', 'J
J'), ('learning', 'JJ'), ('platform', 'NN'), ('.', '.'), ('It', 'PR
P'), ('is', 'VBZ'), ('one', 'CD'), ('of', 'IN'), ('the', 'DT'), ('bes
t', 'JJS'), ('for', 'IN'), ('Computer', 'NNP'), ('Science', 'NNP'),
('students', 'NNS'), ('.', '.'), ('play', 'VB'), ('playing', 'VBG'),
('played', 'JJ'), ('communication', 'NN')]
```

In [13]:

```
sw_nltk = stopwords.words('english')
print(sw_nltk)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'y
ourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'her
s', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
t', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'b
e', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'di
d', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'a
s', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'again
st', 'between', 'into', 'through', 'during', 'before', 'after', 'abov
e', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'ov
er', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'wh
en', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'mor
e', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'ow
n', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'jus
t', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o',
're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'did
n', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't",
'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'must
n', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

In [14]:

```
wordAbs = [word for word in words if word.lower() not in sw_nltk]
new_text = " ".join(wordAbs)
print(new_text)
```

```
GeeksforGeeks great learning platform . one best Computer Science stud
ents . play playing played communication
```

In [15]:

```
# Create representation of document by calculating Term Frequency and Inverse Docum
# Frequency.
```

In [16]:

```
first_sentence = "Data Science is the sexiest job of the 21st century"
second_sentence = "machine learning is the key for data science"
#split so each word have their own string
first_sentence = first_sentence.split(" ")
second_sentence = second_sentence.split(" ")#join them to remove common duplicate w
total= set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'Science', 'is', 'data', 'century', 'job', 'the', 'Data', 'of', '21s
t', 'sexiest', 'machine', 'key', 'science', 'learning', 'for'}
```

In [17]:

```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1
```

In [18]:

```
pd.DataFrame([wordDictA, wordDictB])
```

Out[18]:

	Science	is	data	century	job	the	Data	of	21st	sexiest	machine	key	science	learni
0	1	1	0	1	1	2	1	1	1	1	0	0	0	
1	0	1	1	0	0	1	0	0	0	0	1	1	1	

In [19]:

```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)
    for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
    return(tfDict)
#running our sentences through the tf function:
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)
#Converting to dataframe for visualization
tf = pd.DataFrame([tfFirst, tfSecond])
```

In [20]:

```
tf
```

Out[20]:

	Science	is	data	century	job	the	Data	of	21st	sexiest	machine	key	scienc
0	0.1	0.100	0.000	0.1	0.1	0.200	0.1	0.1	0.1	0.1	0.000	0.000	0.0(
1	0.0	0.125	0.125	0.0	0.0	0.125	0.0	0.0	0.0	0.0	0.125	0.125	0.12

In [21]:

```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)

    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

    return(idfDict)
#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB])
```

In [22]:

```
idfs
```

Out[22]:

```
{'Science': 0.3010299956639812,
 'is': 0.3010299956639812,
 'data': 0.3010299956639812,
 'century': 0.3010299956639812,
 'job': 0.3010299956639812,
 'the': 0.3010299956639812,
 'Data': 0.3010299956639812,
 'of': 0.3010299956639812,
 '21st': 0.3010299956639812,
 'sexiest': 0.3010299956639812,
 'machine': 0.3010299956639812,
 'key': 0.3010299956639812,
 'science': 0.3010299956639812,
 'learning': 0.3010299956639812,
 'for': 0.3010299956639812}
```

In [ ]: