In [1]:
```
pip install PyPDF2
```

```
Collecting PyPDF2
  Downloading https://files.pythonhosted.org/packages/b4/01/68fcc0d43daf4c6bdbc6b33cc3
f77bda531c86b174cac56ef0ffdb96faab/PyPDF2-1.26.0.tar.gz (https://files.pythonhosted.or
g/packages/b4/01/68fcc0d43daf4c6bdbc6b33cc3f77bda531c86b174cac56ef0ffdb96faab/PyPDF2-
1.26.0.tar.gz) (77kB)
Building wheels for collected packages: PyPDF2
  Building wheel for PyPDF2 (setup.py): started
  Building wheel for PyPDF2 (setup.py): finished with status 'done'
  Created wheel for PyPDF2: filename=PyPDF2-1.26.0-cp37-none-any.whl size=61091 sha256
=0d4e38a2d84a2eb596926071afcfcc9af7ee84e4a96279e01fcc241d05e7915c
  Stored in directory: C:\Users\HP\AppData\Local\pip\Cache\wheels\53\84\19\35bc977c8bf
5f0c23a8a011aa958acd4da4bbd7a229315c1b7
Successfully built PyPDF2
Installing collected packages: PyPDF2
Successfully installed PyPDF2-1.26.0
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install python-docx
```

```
Collecting python-docx
  Downloading https://files.pythonhosted.org/packages/8b/a0/52729ce4aa026f31b74cc877be
1d11e4ddeaa361dc7aebec148171644b33/python-docx-0.8.11.tar.gz (https://files.pythonhost
ed.org/packages/8b/a0/52729ce4aa026f31b74cc877be1d11e4ddeaa361dc7aebec148171644b33/pyt
hon-docx-0.8.11.tar.gz) (5.6MB)
Requirement already satisfied: lxml>=2.3.2 in c:\users\hp\anaconda3\lib\site-packages
 (from python-docx) (4.4.1)
Building wheels for collected packages: python-docx
  Building wheel for python-docx (setup.py): started
  Building wheel for python-docx (setup.py): finished with status 'done'
  Created wheel for python-docx: filename=python_docx-0.8.11-cp37-none-any.whl size=18
4607 sha256=4e01ad30c11a4aa6b83f7f4ed6a7d696a2fb4d55f2dfe2cef0f0f8cae90d846e
  Stored in directory: C:\Users\HP\AppData\Local\pip\Cache\wheels\a6\90\f1\a7cb70b3863
3ae04e7fb963b1c70f63fd6fc01c075b8230adc
Successfully built python-docx
Installing collected packages: python-docx
Successfully installed python-docx-0.8.11
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```python
# importing required modules
import PyPDF2

# creating a pdf file object
pdfFileObj = open(r"D:\College\TE\SEM-2\Practical\DSBDA\7\sample1.pdf", 'rb')

# creating a pdf reader object
pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

# printing number of pages in pdf file
print(pdfReader.numPages)

# creating a page object
pageObj = pdfReader.getPage(0)

# extracting text from page
print(pageObj.extractText())

# closing the pdf file object
pdfFileObj.close()
```

```
1
Welcome to Smallpdf
Digital DocumentsŠAll In One Place
Access Files Anytime, Anywhere
Enhance Documents in One Click
Collaborate With Others
With the new Smallpdf experience, you can
freely upload, organize, and share digital
documents. When you enable the
,Storage™
option


your computer, phone, or tablet. We™ll also

Smallpdf Mobile App to our
online portal

you with an array of options to convert,
compress, or modify it.
Forget mundane administrative tasks. With
Smallpdf, you can request e-signatures, send

Smallpdf G Suite
App for your entire organization.
Ready to take document management to the next level?
```

In [9]:
```python
# import docx NOT python-docx
import docx

# create an instance of a word document
doc = docx.Document()

# add a heading of level 0 (largest heading)
doc.add_heading('Heading for the document', 0)

# add a paragraph and store
# the object in a variable
doc_para = doc.add_paragraph('Your paragraph goes here, ')

# add a run i.e, style like
# bold, italic, underline, etc.
doc_para.add_run('hey there, bold here').bold = True
doc_para.add_run(', and ')
doc_para.add_run('these words are italic').italic = True

# add a page break to start a new page
doc.add_page_break()

# add a heading of level 2
doc.add_heading('Heading level 2', 2)

# pictures can also be added to our word document
# width is optional
doc.add_picture(r"D:\College\TE\SEM-2\Practical\DSBDA\7\index.jpg")

# now save the document to a location
doc.save('new_doc')
```

In [10]:
```python
pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\hp\anaconda3\lib\site-packages (3.4.5)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages (from nl
tk) (1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

In [11]:
```python
import nltk
nltk.download()
nltk.download('punkt')
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml (http
s://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)

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

Out[11]:
```
True
```

In [12]:
```
#Sentence Tokenization

sentence_data = "The First sentence is about Python. The Second: about Django. You can
nltk_tokens = nltk.sent_tokenize(sentence_data)
print (nltk_tokens)
```

```
['The First sentence is about Python.', 'The Second: about Django.', 'You can learn Py
thon,Django and Data Ananlysis here.']
```

In [13]:
```
#Non English language Tokenization

german_tokenizer = nltk.data.load('tokenizers/punkt/german.pickle')
german_tokens=german_tokenizer.tokenize('Wie geht es Ihnen?  Gut, danke.')
print(german_tokens)
```

```
['Wie geht es Ihnen?', 'Gut, danke.']
```

In [14]:
```
#Word Tokenization

word_data = "It originated from the idea that there are readers who prefer learning new
nltk_tokens = nltk.word_tokenize(word_data)
print (nltk_tokens)
```

```
['It', 'originated', 'from', 'the', 'idea', 'that', 'there', 'are', 'readers', 'who',
'prefer', 'learning', 'new', 'skills', 'from', 'the', 'comforts', 'of', 'their', 'draw
ing', 'rooms']
```

In [15]:
```
#Word Tokenization

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

#Dummy text
txt = "He is a boy. "\
    "She is a girl"

word_tokens = word_tokenize(txt)

print(word_tokens)
```

```
['He', 'is', 'a', 'boy', '.', 'She', 'is', 'a', 'girl']
```

In [16]:
```
#Part of Speech (POS) tagging

import nltk
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import word_tokenize
text = word_tokenize("Hello welcome to the world of to learn Categorizing and POS Taggi
nltk.pos_tag(text)
```

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

Out[16]:
```
[('Hello', 'NNP'),
 ('welcome', 'NN'),
 ('to', 'TO'),
 ('the', 'DT'),
 ('world', 'NN'),
 ('of', 'IN'),
 ('to', 'TO'),
 ('learn', 'VB'),
 ('Categorizing', 'NNP'),
 ('and', 'CC'),
 ('POS', 'NNP'),
 ('Tagging', 'NNP'),
 ('with', 'IN'),
 ('NLTK', 'NNP'),
 ('and', 'CC'),
 ('Python', 'NNP')]
```

In [17]:
```
import nltk
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
```

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

Out[17]:
```
True
```

In [18]:
```
from nltk.corpus import stopwords
print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', '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', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', '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', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [19]:
```
#Stopwords removal from sentence

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = """This is a sample sentence,
                  showing off the stop words filtration."""

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

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print("Tokenized:", word_tokens)
print("Stop Words Removed:", filtered_sentence)
```

```
Tokenized: ['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']
Stop Words Removed: ['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']
```

In [21]:

```python
#Stopwords from input file

import io
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# word_tokenize accepts
# a string as an input, not a file.
stop_words = set(stopwords.words('english'))
file1 = open(r"D:\College\TE\SEM-2\Practical\DSBDA\7\text.txt")

# Use this to read file content as a stream:
line = file1.read()
words = line.split()
for r in words:
    if not r in stop_words:
        appendFile = open('filteredtext.txt','a')
        appendFile.write(" "+r)
        appendFile.close()
```

In [22]:

```python
#Stemming

import nltk
from nltk.stem.porter import PorterStemmer
porter_stemmer = PorterStemmer()

word_data = "It vijaying meeting better vijayed vijays eats skills originated from the
# First Word tokenization
nltk_tokens = nltk.word_tokenize(word_data)
#Next find the roots of the word
for w in nltk_tokens:
       print("Actual: %s  Stem: %s"  % (w,porter_stemmer.stem(w)))
```

```
Actual: It  Stem: It
Actual: vijaying  Stem: vijay
Actual: meeting  Stem: meet
Actual: better  Stem: better
Actual: vijayed  Stem: vijay
Actual: vijays  Stem: vijay
Actual: eats  Stem: eat
Actual: skills  Stem: skill
Actual: originated  Stem: origin
Actual: from  Stem: from
Actual: the  Stem: the
Actual: idea  Stem: idea
Actual: that  Stem: that
Actual: there  Stem: there
Actual: are  Stem: are
Actual: readers  Stem: reader
Actual: who  Stem: who
Actual: prefer  Stem: prefer
Actual: learning  Stem: learn
Actual: new  Stem: new
Actual: skills  Stem: skill
Actual: from  Stem: from
Actual: the  Stem: the
Actual: comforts  Stem: comfort
Actual: of  Stem: of
Actual: their  Stem: their
Actual: drawing  Stem: draw
Actual: rooms  Stem: room
```

In [23]:
```
#Lemmatization

import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

word_data = "It studies densely is better  meeting studying vijaying vijayed vijays ski
nltk_tokens = nltk.word_tokenize(word_data)
for w in nltk_tokens:
        print("Actual: %s  Lemma: %s"  % (w,wordnet_lemmatizer.lemmatize(w)))
```

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

Actual: It  Lemma: It
Actual: studies  Lemma: study
Actual: densely  Lemma: densely
Actual: is  Lemma: is
Actual: better  Lemma: better
Actual: meeting  Lemma: meeting
Actual: studying  Lemma: studying
Actual: vijaying  Lemma: vijaying
Actual: vijayed  Lemma: vijayed
Actual: vijays  Lemma: vijays
Actual: skills  Lemma: skill
Actual: originated  Lemma: originated
Actual: from  Lemma: from
Actual: the  Lemma: the
Actual: idea  Lemma: idea
Actual: that  Lemma: that
Actual: there  Lemma: there
Actual: are  Lemma: are
Actual: readers  Lemma: reader
Actual: who  Lemma: who
Actual: prefer  Lemma: prefer
Actual: learning  Lemma: learning
Actual: new  Lemma: new
Actual: skills  Lemma: skill
Actual: from  Lemma: from
Actual: the  Lemma: the
Actual: comforts  Lemma: comfort
Actual: of  Lemma: of
Actual: their  Lemma: their
Actual: drawing  Lemma: drawing
Actual: rooms  Lemma: room
```

In [24]:
```
#Expt.No.7 2nd Operation

import pandas as pd
import sklearn as sk
import math
```

In [25]:
```
first_sentence = "Data Science is the best 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 words
total= set(first_sentence).union(set(second_sentence))
print(total)
```

```
{'century', 'Data', 'of', 'for', '21st', 'job', 'learning', 'the', 'Science', 'is', 'k
ey', 'Machine', 'science', 'best', 'data'}
```

In [26]:
```
#count the words

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

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

Out[26]:

	century	Data	of	for	21st	job	learning	the	Science	is	key	Machine	science	best	data
0	1	1	1	0	1	1	0	2	1	1	0	0	0	1	0
1	0	0	0	1	0	0	1	1	0	1	1	1	1	0	1

In [27]:
```
#Compute Term Frequency(TF)

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
pd.DataFrame([tfFirst, tfSecond])
```

Out[27]:

	century	Data	of	for	21st	job	learning	the	Science	is	key	Machine	science	best	d
0	0.1	0.1	0.1	0.000	0.1	0.1	0.000	0.200	0.1	0.100	0.000	0.000	0.000	0.1	0.0
1	0.0	0.0	0.0	0.125	0.0	0.0	0.125	0.125	0.0	0.125	0.125	0.125	0.125	0.0	0.

In [28]:
```
#Compute Inverse Document Frequency(IDF)

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 [29]:
```
#Compute Term Frequency(TF) - Inverse Document Frequency(IDF)

def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return(tfidf)

#running our two sentences through the IDF:
idfFirst = computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)

#putting it in a dataframe
pd.DataFrame([idfFirst, idfSecond])
```

Out[29]:

	century	Data	of	for	21st	job	learning	the	Science	is	
0	0.030103	0.030103	0.030103	0.000000	0.030103	0.030103	0.000000	0.060206	0.030103	0.030103	0.0
1	0.000000	0.000000	0.000000	0.037629	0.000000	0.000000	0.037629	0.037629	0.000000	0.037629	0.0

In [30]:

```
#Compute TF-IDF

#first step is to import the library
from sklearn.feature_extraction.text import TfidfVectorizer

#for the sentence, make sure all words are lowercase or you will run #into error. for s
firstV= "Data Science is the sexiest job of the 21st century"
secondV= "machine learning is the key for data science"

#calling the TfidfVectorizer
vectorize= TfidfVectorizer()

#fitting the model and passing our sentences right away:
response= vectorize.fit_transform([firstV, secondV])

print(response)
```

```
  (0, 1)        0.34211869506421816
  (0, 0)        0.34211869506421816
  (0, 9)        0.34211869506421816
  (0, 5)        0.34211869506421816
  (0, 11)       0.34211869506421816
  (0, 12)       0.48684053853849035
  (0, 4)        0.24342026926924518
  (0, 10)       0.24342026926924518
  (0, 2)        0.24342026926924518
  (1, 3)        0.40740123733358447
  (1, 6)        0.40740123733358447
  (1, 7)        0.40740123733358447
  (1, 8)        0.40740123733358447
  (1, 12)       0.28986933576883284
  (1, 4)        0.28986933576883284
  (1, 10)       0.28986933576883284
  (1, 2)        0.28986933576883284
```

In [ ]: