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)](https://files.pythonhosted.org/packages/b4/01/68fcc0d43daf4c6bdbc6b33cc3f77bda531c86b174cac56ef0ffdb96faab/PyPDF2-1.26.0.tar.gz)

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)](https://files.pythonhosted.org/packages/8b/a0/52729ce4aa026f31b74cc877be1d11e4ddeaa361dc7aebec148171644b33/python-docx-0.8.11.tar.gz)

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]:

*# 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]:

*# 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]:

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]:

**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)](https://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)

In [14]:

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

|  |  |  |
| --- | --- | --- |
| *#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'v

e", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',

'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 't

hey', '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', 'th

e', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'fo

r', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'aft

er', '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', 'ca

n', '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", 'doe

sn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is

n't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "sha

n't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'w ouldn', "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', 'st op', 'words', 'filtration', '.']

Stop Words Removed: ['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'f iltration', '.']

In [21]:

*#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]:

|  |  |  |
| --- | --- | --- |
| *#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)

*#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])

{'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]: |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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. |
|  | 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 [ ]: |  |  |  |