Lab FAT, June 1, Winter 2020-21, L15+L16

WEB MINING

by Sharadindu Adhikari, 19BCE2105

QUESTION (**r** file):

QUESTION -1

Write a python program to find the important words from the text using TFIDF.

Use any 5 documents in the course page location(Samples given)

a) https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/lec1.pdf
b) https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/lec2.pdf
c) https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/lec4.pdf
d) D & E can be taken of your choice..

ANSWER:

Input data:

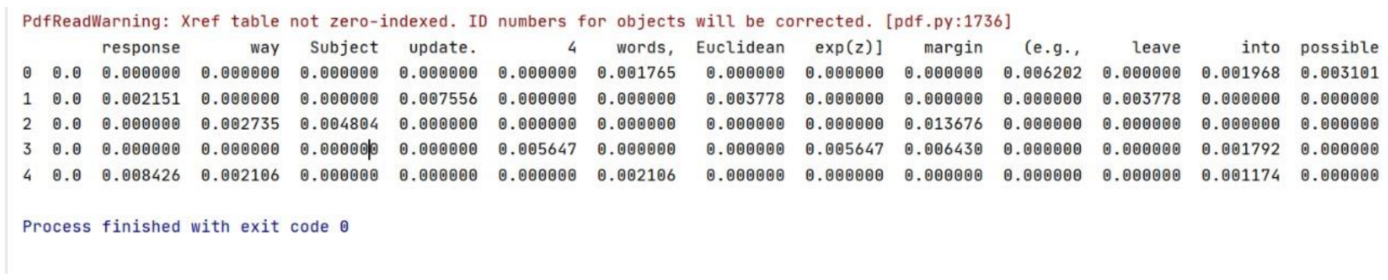
The given 4 link pdfs and I've chosen d) as lec3.pdf (mit.edu) and e) as lec5.pdf (mit.edu)

Code:

```python
import pandas as pd
import PyPDF2

pdf_file = open('F:\\lec1.pdf', 'rb')
read_pdf = PyPDF2.PdfFileReader(pdf_file)
number_of_pagesA = read_pdf.getNumPages()
page = read_pdf.getPage(0)
page_content = page.extractText()
documentA = page_content.encode('utf-8')
bagOfWordsA = page_content.split(' ')

pdf_file = open('D:\lec2.pdf', 'rb')
read_pdf = PyPDF2.PdfFileReader(pdf_file)
number_of_pagesB = read_pdf.getNumPages()
page = read_pdf.getPage(0)
page_content = page.extractText()
documentB = page_content.encode('utf-8')
bagOfWordsB = page_content.split(' ')

pdf_file = open('D:\lec3.pdf', 'rb')
read_pdf = PyPDF2.PdfFileReader(pdf_file)
number_of_pagesC = read_pdf.getNumPages()
page = read_pdf.getPage(0)
page_content = page.extractText()
documentC = page_content.encode('utf-8')
bagOfWordsC = page_content.split(' ')

pdf_file = open('D:\lec4.pdf', 'rb')
read_pdf = PyPDF2.PdfFileReader(pdf_file)
number_of_pagesD = read_pdf.getNumPages()
page = read_pdf.getPage(0)
page_content = page.extractText()
documentD = page_content.encode('utf-8')
bagOfWordsD = page_content.split(' ')

pdf_file = open('D:\lec5.pdf', 'rb')
read_pdf = PyPDF2.PdfFileReader(pdf_file)
number_of_pagesE = read_pdf.getNumPages()
page = read_pdf.getPage(0)
page_content = page.extractText()
documentE = page_content.encode('utf-8')
bagOfWordsE = page_content.split(' ')
```

```
uniqueWords =
set(bagOfWordsA).union(set(bagOfWordsB)).union(set(bagOfWordsC)).union(set(bagOfWordsD
)).union(set(bagOfWordsE))
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
    numOfWordsA[word] += 1
numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
    numOfWordsB[word] += 1
numOfWordsC = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsC:
    numOfWordsC[word] += 1
numOfWordsD = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsD:
    numOfWordsD[word] += 1
numOfWordsE = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsE:
    numOfWordsE[word] += 1


def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict


tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
tfC = computeTF(numOfWordsC, bagOfWordsC)
tfD = computeTF(numOfWordsD, bagOfWordsD)
tfE = computeTF(numOfWordsE, bagOfWordsE)


def computeIDF(documents):
    import math
    N = len(documents)
    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1

    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict


idfs = computeIDF([numOfWordsA, numOfWordsB, numOfWordsC, numOfWordsD, numOfWordsE])


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


tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
tfidfC = computeTFIDF(tfC, idfs)
tfidfD = computeTFIDF(tfD, idfs)
tfidfE = computeTFIDF(tfE, idfs)

df = pd.DataFrame([tfidfA, tfidfB, tfidfC, tfidfD, tfidfE])
print(
    df.to_string())  # Here the '0' row represents the TF-IDF of Document 1 and '1'
row represents the TF-IDF of Document 2.
```

Results:



```
PdfReadWarning: Xref table not zero-indexed. ID numbers for objects will be corrected. [pdf.py:1736]
         response       way   Subject   update.         4    words,  Euclidean   exp(z)]    margin     (e.g.,     leave      into  possible
0  0.0  0.000000  0.000000  0.000000  0.000000  0.000000  0.001765   0.000000  0.000000  0.000000  0.006202  0.000000  0.001968  0.003101
1  0.0  0.002151  0.000000  0.000000  0.007556  0.000000  0.000000   0.003778  0.000000  0.000000  0.000000  0.003778  0.000000  0.000000
2  0.0  0.000000  0.002735  0.004804  0.000000  0.000000  0.000000   0.000000  0.000000  0.013676  0.000000  0.000000  0.000000  0.000000
3  0.0  0.000000  0.000000  0.000000  0.000000  0.005647  0.000000   0.000000  0.005647  0.006430  0.000000  0.000000  0.001792  0.000000
4  0.0  0.008426  0.002106  0.000000  0.000000  0.000000  0.002106   0.000000  0.000000  0.000000  0.000000  0.000000  0.001174  0.000000

Process finished with exit code 0
```