Bag of words , tf-idf , TfidfVectorizer , cosine similarity, NLP basics tutorial

RSREETech

	Doc1	Doc2	Doc3	Doc4
Doc1	1	0.492416	0.492416	0.277687
Doc2	0.492416	1	0.75419	0.215926
Doc3	0.492416	0.75419	1	0.215926
Doc4	0.277687	0.215926	0.215926	1

Definitions

- Corpus

  Is a collection of documents

- Vocabulary

  Collection of all words present in a corpus

- Bag of words

  In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity

Bag of words(BoW) example

- Doc 1 : "Sachin is a great Indian cricket player”
- Doc 2: "Virat is the captain of the Indian cricket team

Corpus ->{Doc1,Doc2}

BoW(Doc1) ->{ Sachin: 1, is:1, a:1, great:1,Indian:1 ,cricket:1,player:1}

BoW(Doc2) ->{Virat:1, is:1, the:2, captain:1, of:1, Indian:1, cricket:1,
team:1}

Bag of words(BoW) example

Vector representation

Doc	Sachin	is	a	great	Indian	cricket	player	Virat	the	team	of	captain
Doc1	1	1	1	1	1	1	1	0	0	0	0	0
Doc2	0	1	0	0	1	1	0	1	2	1	1	1

Definitions

- Term Frequency(tf)

  Is the number of times a word occurs in a document

  tf(t,d) = f (t,d)

  t -> Term, d-> Document

- Inverse Document Frequency(idf)

  Number of documents that contain the word.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

N -> Total number of documents

t -> Term, d-> Document

Term frequency–Inverse document frequency

$$tf\text{-}idf(t,d,C) = tf(t,d) \cdot Idf(t,C)$$

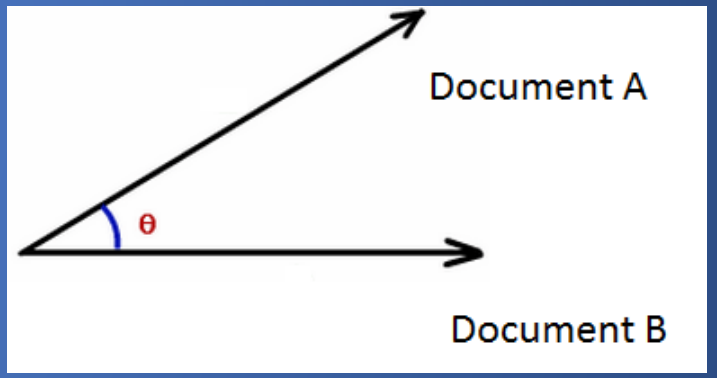
Uncommon words in a document-> Higher tf-idf

More common words across documents -> Almost zero tf-idf

Doc ->{ tf-idf(w1) , tf-idf(w2), …,tf-idf(wn) }

Cosine Similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$



Text Matching :

- Vectors A and B - > tf-idf vectors of the documents.

Cosine similarity :

- Method of normalizing document length during comparison.

Information retrieval:

- Cosine similarity of two documents ->[0,1]
- As tf-idf values are not negative

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Rithesh

C:\Users\Rithesh\Documents\Python Scripts\Tdidf.py

Tdidf.py*

```
# -*- coding: utf-8 -*-
"""
Created on Tue May 19 11:42:08 2020

@author: Rithesh Sreenivasan
"""


from nltk.tokenize import word_tokenize

import numpy as np
import string
import pandas as pd


docs=[ "Sachin is considered to be one of the greatest cricket players",
          "Federer is considered one of the greatest tennis players",
          "Nadal is considered one of the greatest tennis players",
          "Virat is the captain of the  Indian cricket team"

         ]

def createVocab( docList):
    vocab = {}
    for doc in docList:
        print(doc)
        doc= doc.translate(str.maketrans('', '', string.punctuation))

        words= word_tokenize(doc.lower())
        for word in words:
            if(word in vocab.keys()):
                vocab[word] = vocab[word] +1
```

Name Type Size Value

Variable explorer Help Plots Files

Console 6/A

```
In [20]:
```

IPython console History

Kite: ready conda: base (Python 3.7.6) Line 25, Col 24 UTF-8 CRLF RW Mem 64%

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Rithesh\Documents\Python Scripts\Tdidf.py

Tdidf.py*

```
40
41  termDict={}
42
43
44  docsTFMat = np.zeros((len(docs),len(vocab)))
45
46  docsIdfMat = np.zeros((len(vocab),len(docs)))
47
48  docTermDf = pd.DataFrame(docsTFMat ,columns=sorted(vocab.keys()))
49  docCount=0
50  for doc in docs:
51      doc= doc.translate(str.maketrans('', '', string.punctuation))
52      words= word_tokenize(doc.lower())
53      for word in words:
54          if(word in vocab.keys()):
55            docTermDf[word][docCount] = docTermDf[word][docCount] +1
56
57      docCount = docCount +1
58
59
60  #Computed idf for each word in vocab
61  idfDict={}
62
63  for column in docTermDf.columns:
64      idfDict[column]= np.log((len(docs) +1 )/(1+ (docTermDf[column] != 0).sum()))+1
65
66  #compute tf.idf matrix
67  docsTfIdfMat = np.zeros((len(docs),len(vocab)))
68  docTfIdfDf = pd.DataFrame(docsTfIdfMat ,columns=sorted(vocab.keys()))
69
70
71
```

Name	Type	Size	Value
docs	list	4	['Sachin is considered to be one of the greatest cricket players', 'Fe ...
vocab	dict	18	{'sachin':1, 'is':4, 'considered':3, 'to':1 'be':1, 'one':3, 'of':4, ...

Variable explorer | Help | Plots | Files

Console 6/A

```
    ...:             if(word in vocab.keys()):
    ...:                 vocab[word] = vocab[word]
+1
    ...:             else:
    ...:                 vocab[word] =1
    ...:     return vocab
    ...:
    ...: vocab = createVocab(docs)
Sachin is considered to be one of the greatest
cricket players
Federer is considered one of the greatest tennis
players
Nadal is considered one of the greatest tennis
players
Virat is the captain of the  Indian cricket team

In [23]:
```

IPython console | History

Kite: ready | conda: base (Python 3.7.6) | Line 58, Col 10 | UTF-8 | CRLF | RW | Mem 57%

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Rithesh\Documents\Python Scripts\Tdidf.py

Tdidf.py

```
58
59
60  #Computed idf for each word in vocab
61  idfDict={}
62
63  for column in docTermDf.columns:
64      idfDict[column]= np.log((len(docs) +1 )/(1+ (docTermDf[column] != 0).sum()))+1
65
66  #compute tf.idf matrix
67  docsTfIdfMat = np.zeros((len(docs),len(vocab)))
68  docTfIdfDf = pd.DataFrame(docsTfIdfMat ,columns=sorted(vocab.keys()))
69
70
71
72  docCount = 0
73  for doc in docs:
74      for key in idfDict.keys():
75          docTfIdfDf[key][docCount] = docTermDf[key][docCount] * idfDict[key]
76      docCount = docCount +1
77
78  print(docTfIdfDf)
79
80  ## Use TfidfVectorizer to perfom the same
81
82
83  from sklearn.feature_extraction.text import TfidfVectorizer
84  from sklearn.metrics.pairwise import cosine_similarity
85
86  vectorizer = TfidfVectorizer(analyzer='word',norm=None, use_idf=True,smooth_idf=True)
87  tfIdfMat  = vectorizer.fit_transform(docs)
88
89  feature_names = sorted(vectorizer.get_feature_names())
```

C:\Users\Rithesh

Name	Type	Size	Value
doc	str	1	Virat is the captain…
docCount	int	1	4
docTermDf	DataFrame	(4, 18)	Column names: be, ca…
docs	list	4	['Sachin is consider…
docsIdfMat	Array of float64	(18, 4)	[[0. 0. 0. 0.] [0. 0. 0. 0.]
docsTFMat	Array of float64	(4, 18)	[[1. 0. 1. ... 1. 1.… [0. 0. 1. ... 1. 0.…
termDict	dict	0	{}
vocab	dict	18	{'sachin':1, 'is':4,…

Variable explorer | Help | Plots | Files

Console 6/A

```
np.zeros((len(vocab),len(docs)))
    ...:
    ...: docTermDf = pd.DataFrame(docsTFMat
,columns=sorted(vocab.keys()))
    ...: docCount=0
    ...: for doc in docs:
    ...:     doc= doc.translate(str.maketrans('',
'', string.punctuation))
    ...:     words= word_tokenize(doc.lower())
    ...:     for word in words:
    ...:         if(word in vocab.keys()):
    ...:           docTermDf[word][docCount] =
docTermDf[word][docCount] +1
    ...:
    ...:     docCount = docCount +1

In [24]:
```

IPython console | History

Kite: ready | conda: base (Python 3.7.6) | Line 60, Col 37 | UTF-8 | CRLF | RW | Mem 58%

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Rithesh\Documents\Python Scripts\Tdidf.py

Tdidf.py*

```
64      idfDict[column]= np.log((len(docs) +1 )/(1+ (docTermDf[column] != 0).sum()))+1
65
66  #compute tf.idf matrix
67  docsTfIdfMat = np.zeros((len(docs),len(vocab)))
68  docTfIdfDf = pd.DataFrame(docsTfIdfMat ,columns=sorted(vocab.keys()))
69
70
71
72  docCount = 0
73  for doc in docs:
74      for key in idfDict.keys():
75          docTfIdfDf[key][docCount] = docTermDf[key][docCount] * idfDict[key]
76      docCount = docCount +1
77
78  print(docTfIdfDf)
79
80  ## Use TfidfVectorizer to perfom the same
81
82
83  from sklearn.feature_extraction.text import TfidfVectorizer
84  from sklearn.metrics.pairwise import cosine_similarity
85
86  vectorizer = TfidfVectorizer(analyzer='word',norm=None, use_idf=True,smooth_idf=True)
87  tfIdfMat  = vectorizer.fit_transform(docs)
88
89  feature_names = sorted(vectorizer.get_feature_names())
90
91
92  skDocsTfIdfdf = pd.DataFrame(tfIdfMat.todense(),  columns=feature_names)
93  print(skDocsTfIdfdf)
94
95
```

Name	Type	Size	Value
column	str	1	virat
doc	str	1	Virat is the captain…
docCount	int	1	4
docTermDf	DataFrame	(4, 18)	Column names: be, ca…
docs	list	4	['Sachin is consider…
docsIdfMat	Array of float64	(18, 4)	[[0. 0. 0. 0.] [0. 0. 0. 0.]
docsTFMat	Array of float64	(4, 18)	[[1. 0. 1. ... 1. 1.… [0. 0. 1. ... 1. 0.…
idfDict	dict	18	{'be':1.916290731874…

Variable explorer | Help | Plots | Files

Console 6/A

```
...:     words= word_tokenize(doc.lower())
...:     for word in words:
...:         if(word in vocab.keys()):
...:           docTermDf[word][docCount] =
docTermDf[word][docCount] +1
...:
...:     docCount = docCount +1
...:
...:
...: #Computed idf for each word in vocab
...: idfDict={}
...:
...: for column in docTermDf.columns:
...:     idfDict[column]= np.log((len(docs) +1 )/
(1+ (docTermDf[column] != 0).sum()))+1

In [6]:
```

IPython console | History

Kite: ready | conda: base (Python 3.7.6) | Line 70, Col 1 | UTF-8 | CRLF | RW | Mem 62%

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Rithesh\Documents\Python Scripts\Tdidf.py

Tdidf.py

```
78  print(docTfIdfDf)
79
80  ## Use TfidfVectorizer to perfom the same
81
82
83  from sklearn.feature_extraction.text import TfidfVectorizer
84  from sklearn.metrics.pairwise import cosine_similarity
85
86  vectorizer = TfidfVectorizer(analyzer='word',norm=None, use_idf=True,smooth_idf=True)
87  tfIdfMat  = vectorizer.fit_transform(docs)
88
89  feature_names = sorted(vectorizer.get_feature_names())
90
91  docList=['Doc 1','Doc 2','Doc 3','Doc 4']
92  skDocsTfIdfdf = pd.DataFrame(tfIdfMat.todense(),index=sorted(docList),  columns=feature_
93  print(skDocsTfIdfdf)
94
95
96  #compute cosine similarity
97  csim = cosine_similarity(tfIdfMat,tfIdfMat)
98
99  csimDf = pd.DataFrame(csim,index=sorted(docList),columns=sorted(docList))
100
101
102
103
104
105
106
107
108
```

C:\Users\Rithesh

Name	Type	Siz
DocList	list	4
column	str	1
csim	Array of float64	(4, 4
csimDf	DataFrame	(4, 4
doc	str	1
docCount	int	1
docList	list	4
docTermDf	DataFrame	(4, 1

Variable explorer Help Plots Files

Console 6/A

```
In [48]:
```

IPython console History

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Rithesh\Documents\Python Scripts\Tdidf.py

Tdidf.py

```
78  print(docTfIdfDf)
79
80  ## Use TfidfVectorizer to perfom the same
81
82
83  from sklearn.feature_extraction.text import TfidfVectorizer
84  from sklearn.metrics.pairwise import cosine_similarity
85
86  vectorizer = TfidfVectorizer(analyzer='word',norm=None, use_idf=True,smooth_idf=True)
87  tfIdfMat   = vectorizer.fit_transform(docs)
88
89  feature_names = sorted(vectorizer.get_feature_names())
90
91  docList=['Doc 1','Doc 2','Doc 3','Doc 4']
92  skDocsTfIdfdf = pd.DataFrame(tfIdfMat.todense(),index=sorted(docList),  columns=feature_
93  print(skDocsTfIdfdf)
94
95
96  #compute cosine similarity
97  csim = cosine_similarity(tfIdfMat,tfIdfMat)
98
99  csimDf = pd.DataFrame(csim,index=sorted(docList),columns=sorted(docList))
100
101
102
103
104
105
106
107
108
```

Name	Type	Siz
csimDf	DataFrame	(4, 4
doc	str	1
docCount	int	1
docList	list	4
docTermDf	DataFrame	(4, 1
docs	list	4
docsIdfMat	Array of float64	(18,
docsTFMat	Array of float64	(4, 1

Variable explorer | Help | Plots | Files

Console 6/A

```
1.916291  0.000000
Doc 2  0.000000  0.000000    1.223144  ...  1.0
0.000000  0.000000
Doc 3  0.000000  0.000000    1.223144  ...  1.0
0.000000  0.000000
Doc 4  0.000000  1.916291    0.000000  ...  2.0
0.000000  1.916291

[4 rows x 18 columns]

In [50]: csim = cosine_similarity(tfIdfMat,tfIdfMat)
    ...:
    ...: csimDf =
pd.DataFrame(csim,index=sorted(docList),columns=sort
ed(docList))

In [51]:
```

IPython console | History

Kite: ready | conda: base (Python 3.7.6) | Line 99, Col 74 | UTF-8 | CRLF | RW | Mem 53%

docs - List (4 elements)

Index	Type	Size	Value
0	str	1	Sachin is considered to be one of the greatest cricket players
1	str	1	Federer is considered one of the greatest tennis players
2	str	1	Nadal is considered one of the greatest tennis players
3	str	1	Virat is the captain of the Indian cricket team

Save and Close | Close

csimDf - DataFrame

Index	Doc 1	Doc 2	Doc 3	Doc 4
Doc 1	1	0.492416	0.492416	0.277687
Doc 2	0.492416	1	0.75419	0.215926
Doc 3	0.492416	0.75419	1	0.215926
Doc 4	0.277687	0.215926	0.215926	1

https://github.com/rsreetech/Tf-idf