TFIDF

In [1]:
```
docA = "i had a good day my teacher was very happy"
docB = "today i went to theatre i enjoyed a lot"
docQ="today i went to park  i enjoyed a lot" ##given query
```

In [2]:
```
bowA = docA.split(" ")
bowB = docB.split(" ")
bowQ=docQ.split(" ")
```

In [3]:
```
bowA
```

Out[3]:
```
['i', 'had', 'a', 'good', 'day', 'my', 'teacher', 'was', 'very', 'happy']
```

In [4]:
```
wordSet = set(bowA).union(set(bowB).union(set(bowQ)))
```

In [5]:
```
wordSet
```

Out[5]:
```
{'',
 'a',
 'day',
 'enjoyed',
 'good',
 'had',
 'happy',
 'i',
 'lot',
 'my',
 'park',
 'teacher',
 'theatre',
 'to',
 'today',
 'very',
 'was',
 'went'}
```

In [6]:
```
wordDictA = dict.fromkeys(wordSet, 0)
wordDictB = dict.fromkeys(wordSet, 0)
wordDictQ=dict.fromkeys(wordSet,0)
```

In [7]:
```
for word in bowA:
    wordDictA[word]+=1

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

In [8]:
```
for word in bowQ:
    wordDictQ[word]+=1
```

In [9]:
```
import pandas as pd
pd.DataFrame([wordDictA, wordDictB, wordDictQ])
```

Out[9]:

		i	very	park	a	lot	teacher	good	today	to	went	had	was	enjoyed	theatre	day	happy	my
0	0	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	1	1
1	0	2	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0
2	1	2	0	1	1	1	0	0	1	1	1	0	0	1	0	0	0	0

TF

TERM FREQUENCY= word(i)/Total count

In [10]:
```
def computeTF(wordDict, bow):
    tfDict = {}
    bowCount = len(bow)
    for word, count in wordDict.items():
        tfDict[word] = count/float(bowCount)
    return tfDict
```

In [11]:
```
tfBowA = computeTF(wordDictA, bowA)
tfBowB = computeTF(wordDictB, bowB)
tfBowQ=computeTF(wordDictQ, bowQ)
```

In [12]:
```
pd.DataFrame([tfBowA,tfBowB,tfBowQ])
```

Out[12]:

		i	very	park	a	lot	teacher	good	today	to	went	had	was	enjoyed	theatre	day	happy	
0	0.0	0.100000	0.1	0.0	0.100000	0.000000	0.1	0.1	0.000000	0.000000	0.000000	0.1	0.1	0.000000	0.000000	0.1	0.1	
1	0.0	0.222222	0.0	0.0	0.111111	0.111111	0.0	0.0	0.111111	0.111111	0.111111	0.0	0.0	0.111111	0.111111	0.0	0.0	
2	0.1	0.200000	0.0	0.1	0.100000	0.100000	0.0	0.0	0.100000	0.100000	0.100000	0.0	0.0	0.100000	0.000000	0.0	0.0	

IDF

Inverse Document Frequncy T= Log(1+d/Total number of Document) Idf is calculated for each unique term in document where d is number of documents in which term T has appeared

In [13]:
```
def computeIDF(docList):
    import math
    idfDict = {}
    N = len(docList)

    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for doc in docList:
        for word, val in doc.items():
            if val > 0:
                idfDict[word] += 1

    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / float(val))

    return idfDict
```

In [14]:
```
idfs = computeIDF([wordDictA, wordDictB,wordDictQ])
import pandas as pd
IDF=pd.DataFrame([idfs])
```

TFIDF

In [15]:
```
def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return tfidf
```

In [16]:
```
tfidfBowA = computeTFIDF(tfBowA, idfs)
tfidfBowB = computeTFIDF(tfBowB, idfs)
tfidfBowQ = computeTFIDF(tfBowQ, idfs)
```

In [17]:
```
pd.DataFrame([tfidfBowA, tfidfBowB, tfidfBowQ])
```

Out[17]:

		i	very	park	a	lot	teacher	good	today	to	went	had	was	enjoyed	theatre
0	0.000000	0.0	0.047712	0.000000	0.0	0.000000	0.047712	0.047712	0.000000	0.000000	0.000000	0.047712	0.047712	0.000000	0.000000
1	0.000000	0.0	0.000000	0.000000	0.0	0.019566	0.000000	0.000000	0.019566	0.019566	0.019566	0.000000	0.000000	0.019566	0.053013
2	0.047712	0.0	0.000000	0.047712	0.0	0.017609	0.000000	0.000000	0.017609	0.017609	0.017609	0.000000	0.000000	0.017609	0.000000

Cosine Similarity

In [18]:
```
import re, math
from collections import Counter

WORD = re.compile(r'\w+')
def get_cosine(vec1, vec2):
    intersection = set(vec1.keys()) & set(vec2.keys())
    numerator = sum([vec1[x] * vec2[x] for x in intersection])

    sum1 = sum([vec1[x]**2 for x in vec1.keys()])
    sum2 = sum([vec2[x]**2 for x in vec2.keys()])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominator:
        return 0.0
    else:
        return float(numerator) / denominator

def text_to_vector(text):
    words = WORD.findall(text)
    return Counter(words)

vector1 = text_to_vector(docA)
vector2 = text_to_vector(docB)
vectorQ= text_to_vector(docQ)

cosine = get_cosine(vector1, vectorQ)
print ('Cosine:',cosine)
```

```
Cosine: 0.28603877677367767
```

In [19]:
```
cosine=get_cosine(vector2, vectorQ)
print('Cosine:',cosine)
```

```
Cosine: 0.9090909090909091
```

In [0]:
```
```