In [1]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [2]:
```
documentA = 'the man went out for a walk'
documentB = 'the children sat around the fire'
```

In [3]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

In [4]:
```
bagOfWordsA
```

Out[4]:
```
['the', 'man', 'went', 'out', 'for', 'a', 'walk']
```

In [5]:
```
bagOfWordsB
```

Out[5]:
```
['the', 'children', 'sat', 'around', 'the', 'fire']
```

In [6]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

In [7]:
```
uniqueWords
```

Out[7]:
```
{'a',
 'around',
 'children',
 'fire',
 'for',
 'man',
 'out',
 'sat',
 'the',
 'walk',
 'went'}
```

In [8]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)

for word in bagOfWordsA:
    numOfWordsA[word] += 1

numOfWordsB = dict.fromkeys(uniqueWords, 0)

for word in bagOfWordsB:
    numOfWordsB[word] += 1
```

In [10]:
```
from nltk.corpus import stopwords
#stopwords.words('english')
```

In [11]:
```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count / float(bagOfWordsCount)
    return tfDict
```

In [12]:
```
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

In [13]:
```
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
```

In [14]:
```
idfs = computeIDF([numOfWordsA, numOfWordsB])
```

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

In [16]:
```
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
```

In [17]:
```
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform([documentA, documentB])
feature_names = vectorizer.get_feature_names()
dense = vectors.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist, columns=feature_names)
```

In [18]:
```
df
```

Out[18]:

	around	children	fire	for	man	out	sat	the	walk	went
0	0.000000	0.000000	0.000000	0.42616	0.42616	0.42616	0.000000	0.303216	0.42616	0.42616
1	0.407401	0.407401	0.407401	0.00000	0.00000	0.00000	0.407401	0.579739	0.00000	0.00000

In [ ]: