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

In [2]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
bagOfWordsA = documentA.split(' ')
bagOfWordsA
```

Out[2]:
```
['Jupiter', 'is', 'the', 'largest', 'Planet']
```

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

Out[3]:
```
['Mars', 'is', 'the', 'fourth', 'planet', 'from', 'the', 'Sun']
```

In [4]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
uniqueWords
```

Out[4]:
```
{'Jupiter',
 'Mars',
 'Planet',
 'Sun',
 'fourth',
 'from',
 'is',
 'largest',
 'planet',
 'the'}
```

In [5]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
```

In [6]:
```
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 [12]:
```
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)
```

In [15]:
```
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])
idfs
```

Out[15]:
```
{'largest': 0.6931471805599453,
 'the': 0.0,
 'Mars': 0.6931471805599453,
 'is': 0.0,
 'Jupiter': 0.6931471805599453,
 'Planet': 0.6931471805599453,
 'fourth': 0.6931471805599453,
 'Sun': 0.6931471805599453,
 'from': 0.6931471805599453,
 'planet': 0.6931471805599453}
```

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

Out[16]:

	largest	the	Mars	is	Jupiter	Planet	fourth	Sun	from	planet
0	0.138629	0.0	0.000000	0.0	0.138629	0.138629	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.0	0.086643	0.0	0.000000	0.000000	0.086643	0.086643	0.086643	0.086643

In [ ]: