In [2]:
```python
# Task 1
import re
import numpy as np

# load up all of the 19997 documents in the corpus
corpus = sc.textFile(
    "s3://chrisjermainebucket/comp330_A6/20_news_same_line.txt")

# each entry in validLines will be a line from the text file
validLines = corpus.filter(lambda x: 'id' in x)

# now we transform it into a bunch of (docID, text) pairs
keyAndText = validLines.map(
    lambda x: (x[x.index('id="') + 4:x.index('" url=')], x[x.index('">') + 2:]))

# now we split the text in each (docID, text) pair into a list of words
# after this, we have a data set with (docID, ["word1", "word2", "word3", ...])
# we have a bit of fancy regular expression stuff here to make sure that we do not
# die on some of the documents
regex = re.compile('[^a-zA-Z]')
keyAndListOfWords = keyAndText.map(
    lambda x: (str(x[0]), regex.sub(' ', x[1]).lower().split()))

# now get the top 20,000 words... first change (docID, ["word1", "word2", "word3", ...])
# to ("word1", 1) ("word2", 1)...
allWords = keyAndListOfWords.flatMap(lambda x: ((j, 1) for j in x[1]))
# now, count all of the words, giving us ("word1", 1433), ("word2", 3423423), etc.
allCounts = allWords.reduceByKey(lambda a, b: a + b)

# and get the top 20,000 words in a local array
# each entry is a ("word1", count) pair
topWords = allCounts.top(20000, lambda x: x[1])

# and we'll create a RDD that has a bunch of (word, dictNum) pairs
# start by creating an RDD that has the number 0 thru 20000
# 20000 is the number of words that will be in our dictionary
twentyK = sc.parallelize(range(20000))

# now, we transform (0), (1), (2), ... to ("mostcommonword", 1) ("nextmostcommon", 2), ...
# the number will be the spot in the dictionary used to tell us where the word is located
# A bunch of (word, posInDictionary) pairs
# dictionary.top(3) => [('zz', 6505), ('zyxel', 13838), ('zyeh', 18666)]
dictionary = twentyK.map(lambda x: (topWords[x][0], x))
# A bunch of (word, docID) pairs
# wordDictPair.top(3) => [('zzzzzzt', '20_newsgroups/rec.sport.baseball/104569'), ('zzzzzz', '20_newsgr
oups/rec.sport.hockey/53841'), ('zzzzzz', '20_newsgroups/rec.sport.baseball/105004')]
wordDictPair = keyAndListOfWords.flatMap(lambda x: ((j, x[0]) for j in x[1]))

# Join the two RDDs, you'll have a bunch of (word, (docID, posInDictionary)) pairs
# wordPair.top(3) => [('zz', ('20_newsgroups/talk.politics.guns/54380', 6505)), ('zz', ('20_newsgroups/
talk.politics.guns/54380', 6505)), ('zz', ('20_newsgroups/talk.politics.guns/54380', 6505))]
wordPair = wordDictPair.join(dictionary)

# Get a bunch of (docid, (listOfAllDictonaryPos)) pairs
# docIDIdxPair.top(3) => [('20_newsgroups/talk.religion.misc/84570', 19590), ('20_newsgroups/talk.relig
ion.misc/84570', 16881), ('20_newsgroups/talk.religion.misc/84570', 16475)]
# docIDAllIdxPair.top(2) => [('20_newsgroups/talk.religion.misc/84570', <pyspark.resultiterable.ResultI
terable object at 0x7fe93fa07190>), ('20_newsgroups/talk.religion.misc/84569', <pyspark.resultiterable.
ResultIterable object at 0x7fe93fa07d50>)]
docIDIdxPair = wordPair.map(lambda x: (x[1][0], x[1][1]))
docIDAllIdxPair = docIDIdxPair.groupByKey()
docIDAllIdxList = docIDAllIdxPair.map(lambda x: (x[0], list(x[1])))
docIDAllIdxList.top(1)
```

```
[('20_newsgroups/talk.religion.misc/84570', [1, 12, 12, 12, 12, 12, 12, 12, 12, 14, 201, 317, 1305, 1
2209, 4409, 15, 33, 78, 4002, 38, 38, 46, 46, 53, 123, 357, 812, 812, 2498, 4195, 9336, 92, 143, 143,
950, 930, 7271, 4722, 13, 26, 1817, 1958, 6, 6, 23, 34, 17, 45, 45, 2293, 4402, 6020, 9265, 164, 176,
1660, 2, 200, 36, 360, 210, 2153, 1654, 31, 367, 1262, 19904, 9552, 16847, 0, 0, 18, 18, 28, 28, 28,
28, 1237, 6102, 16467, 2403, 20, 20, 20, 25, 25, 49, 154, 415, 1573, 5794, 2269, 3, 4, 4, 4, 4, 32, 5
9, 59, 59, 59, 59, 184, 4989, 2003, 2218, 5, 5, 57, 87, 98, 117, 117, 148, 473, 6273, 6274, 24, 37, 4
7, 54, 61, 133, 1096, 1458, 1574, 4786, 1950])]
```

In [3]:
```python
# Then finally, you will write a map () that will take that RDD and convert into the listOfAllDictonary
Pos values to a NumPy array.
def transformNP(idxList):
    array = np.zeros(20000)
    for i in idxList:
        array[i] += 1
    return array


result = docIDAllIdxList.map(lambda x: (x[0], transformNP(x[1])))
```

In [4]:
```python
arr37261 = np.array(result.lookup("20_newsgroups/comp.graphics/37261"))
arr37261[arr37261.nonzero()]
```

```
array([ 8.,  2.,  6.,  3., 12.,  4.,  3.,  6.,  2.,  1.,  1.,  5.,  2.,
        2.,  2.,  3.,  1.,  1.,  1.,  1.,  3.,  1.,  1.,  2.,  3.,  4.,
        1.,  1.,  1.,  1.,  1.,  3.,  1.,  1.,  1.,  2.,  1.,  1.,  1.,
        2.,  1.,  1.,  2.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  2.,
        1.,  1.,  2.,  2.,  1.,  2.,  1.,  1.,  1.,  3.,  4.,  1.,  1.,
        1.,  1.,  2.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  2.,  1.,
        1.,  1.,  1.,  5.,  2.,  2.,  1.,  1.,  5.,  1.,  4.,  1.,  1.,
        1.,  2.,  1.,  2.,  1., 11.,  1.,  1.,  1.,  1.,  2.,  2.,  2.,
        5.,  1.,  2.,  1.,  1.,  1.,  1.,  2.,  1.,  2.,  2.,  4.,  1.,
        1.,  1.,  5.,  1.,  1.,  1.,  1.,  2.,  4.,  1.,  1.,  1.,  3.,
        1.,  1.,  1.,  1.,  3.,  2.,  2.,  1.,  1.,  6.,  1.,  6.,  1.,
        1.,  3.,  1.,  1.,  2.,  1.,  1.,  1.,  1.,  2.,  7.,  1.,  1.,
        1.,  1.,  1.])
```

In [5]:
```python
arr75944 = np.array(result.lookup("20_newsgroups/talk.politics.mideast/75944"))
arr75944[arr75944.nonzero()]
```

```
array([135.,  37.,  71.,  28.,  49.,  19.,  46.,  16.,  13.,  22.,   9.,
        22.,  11.,   7.,   7.,   6.,   4.,   6.,  12.,  11.,  10.,   3.,
        10.,   4.,   2.,  21.,   5.,   4.,   2.,   2.,   1.,   1.,   1.,
         5.,   1.,  23.,   5.,   2.,   1.,   6.,   8.,   4.,   7.,   3.,
         3.,   2.,   1.,   1.,   6.,   4.,   4.,   7.,   1.,   8.,   7.,
        13.,   4.,   4.,  10.,   3.,   3.,   2.,   2.,   3.,   7.,   4.,
         1.,   2.,   4.,   8.,   4.,   7.,   2.,   1.,   1.,   2.,   1.,
         2.,   2.,   1.,   5.,   3.,   3.,   3.,   1.,   1.,   1.,   2.,
         1.,   4.,   3.,   1.,   3.,   3.,   4.,   7.,   1.,   2.,   1.,
         3.,   2.,   1.,   4.,   6.,   3.,  11.,   1.,   6.,   3.,   1.,
         3.,   1.,   2.,   1.,   1.,   1.,   3.,   3.,   2.,   5.,   2.,
         2.,   2.,   2.,   1.,   1.,   1.,   1.,   1.,   3.,   1.,   1.,
         1.,   1.,   3.,   3.,   4.,   1.,   1.,   5.,   1.,   1.,   2.,
         6.,   2.,   2.,   1.,   1.,   1.,   1.,   1.,   1.,   3.,   5.,
         1.,   1.,   1.,   1.,   2.,   1.,   1.,   1.,   6.,   1.,   1.,
         2.,   1.,   3.,   2.,   1.,   1.,   3.,   2.,   2.,   3.,   8.,
         1.,   1.,   1.,   1.,   2.,   3.,   1.,   2.,   6.,   1.,   1.,
         2.,   1.,  13.,   4.,   1.,   1.,   1.,   1.,   1.,   3.,   1.,
         1.,   2.,   2.,   1.,   1.,   1.,   2.,   1.,   1.,   1.,   1.,
         3.,   1.,   2.,   4.,  26.,   1.,   1.,   3.,   2.,   2.,   1.,
         3.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   2.,   6.,
         1.,   1.,   1.,   6.,   1.,   1.,   1.,   4.,   2.,   1.,   1.,
         1.,   1.,   1.,   4.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,
         3.,   4.,   4.,   4.,   3.,   1.,   3.,   1.,   1.,   1.,   1.,
         1.,   1.,   4.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,
         1.,   1.,   1.,   1.,   2.,   1.,   1.,   2.,   1.,   9.,   1.,
         1.,   2.,   1.,   1.,   1.,   5.,   1.,   1.,   2.,   1.,   1.,
         2.,   1.,   1.,   1.,   1.,   6.,   1.,   1.,   1.,   1.,   1.,
         3.,   1.,   2.,   1.,   1.,   1.,   2.,   1.,   2.,   9.,   1.,
         1.,   1.,   2.,   1.,   1.,   2.,   2.,   2.,   1.,   1.,   2.,
         4.,   1.,   1.,   1.,   2.,   1.,   1.,   1.,   1.,  11.,   2.,
         1.,   1.,   1.,   1.,   1.,   1.,   1.,   2.,   1.,   1.,   1.,
         1.,   1.,   1.,   2.,   9.,   1.,   2.,   7.,   3.,   3.,   2.,
         1.,   1.,   2.,   1.,   1.,   1.,   2.,   2.,   1.,   1.,   3.,
         1.,   8.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   8.,   1.,
         2.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,
         1.,  10.,   1.,   1.,   1.,   1.,   2.,   2.,   3.,   4.,   1.,
         1.,   1.,   1.,   1.,   1.,   1.,   3.,   1.,   1.,   1.,   1.,
         1.,   1.,   8.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   6.,
         1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   2.,   1.,
         1.,   1.,   1.,   1.,  11.,   2.,   1.,   1.,   1.,   1.,   1.,
         2.,   1.,   1.,   3.,   3.,   1.,   1.,   1.,   1.,   1.,   1.,
         1.,   1.,   1.,   1.,   1.,   2.,   1.,   3.,   1.,   1.,   1.,
         1.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,   2.,   1.,   4.,
         2.,   3.,   1.,   1.,   1.,   4.,   1.,   1.,   4.,   1.,   1.,
         1.,   2.,   4.,   1.,   1.,   1.,   1.,   2.,   3.,   1.,   1.,
         1.,   2.,   1.,   2.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,
         1.,   1.,   1.,   2.,   1.,   1.,   1.,   1.,   1.,   2.,   1.,
         2.,   1.,   2.,   2.,   1.,   1.,   1.,   1.,   1.,   4.,   1.,
         1.,   1.,   1.,   8.,   1.,   2.,   1.,   1.,   1.,   2.,   3.,
         1.,   1.,   1.,   1.,   1.,   3.,   1.,   1.,   1.,   1.,   1.,
         4.,   1.,   1.,   1.,   3.,   2.,   1.,   1.,   1.,   1.,   2.,
         1.,   1.,   1.,   1.,   2.,   1.,   1.,   1.,   1.,   1.,   1.,
         1.,   1.,   1.,   2.,   1.,   1.,   1.,   1.,   1.,   1.,   1.,
         1.,   1.,   1.,   1.,   1.,   2.,   1.,   1.,   1.,   1.,   1.,
         4.,   3.,   2.,   1.,   2.,   1.,   1.,   2.])
```

In [6]:
```python
arr58763 = np.array(result.lookup("20_newsgroups/sci.med/58763"))
arr58763[arr58763.nonzero()]
```

```
array([4., 4., 3., 2., 1., 1., 4., 3., 1., 2., 1., 5., 1., 2., 1., 1., 1.,
       2., 1., 1., 1., 1., 1., 1., 2., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 2., 2., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 2., 1., 1., 1., 1., 2., 1., 1., 1., 2., 1., 5., 1.,
       1., 1., 1., 1., 1., 1., 2., 1., 2., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 3., 1., 1.])
```

In [7]:
```python
# Task 2
# i => word_i, d => documents_d
import math


def findWord(wordList):
    arr = np.zeros(20000)
    for idx, i in enumerate(wordList):
        if i > 0:
            arr[idx] += 1
    return arr

def IDF(result):
    cnt = 0
    val = result.map(lambda x: ("key", findWord(x[1])))
    finalList = val.reduceByKey(lambda a, b: a + b)
    return finalList


# Calculate IDF values
size = result.count()
IDFList = IDF(result)
IDFArr = np.array(IDFList.lookup("key"))
IDFArr = np.log(size / IDFArr)

# 20_newsgroups/comp.graphics/37261
arrCnt = np.array(result.lookup("20_newsgroups/comp.graphics/37261"))
TFArr = arrCnt / arrCnt.sum()
finalResult = np.multiply(TFArr, IDFArr)
print(finalResult[finalResult.nonzero()])
```

```
[1.92555059e-03 8.50478606e-04 3.65947527e-03 1.27515309e-03
 7.26881462e-03 2.19975741e-03 2.66216132e-03 6.35871363e-03
 3.43014584e-03 1.65693809e-03 9.44999233e-03 3.95217425e-03
 4.19787791e-03 5.18781198e-03 6.54906258e-03 9.61804463e-06
 9.42766276e-03 2.87951994e-03 5.38035007e-04 6.14710796e-03
 9.49332595e-03 1.50258572e-02 3.37418661e-03 4.94317579e-03
 5.98488082e-03 4.54842076e-03 4.99141867e-03 1.65903978e-02
 4.91237317e-03 7.28885166e-03 6.36745621e-03 1.33917572e-02
 6.28356577e-03 7.13964429e-03 7.13821425e-03 1.59751652e-02
 6.41882998e-03 7.64721754e-03 1.62776158e-02 8.57654687e-03
 7.64721754e-03 8.98232287e-03 8.13687431e-03 7.89645803e-03
 8.90098531e-03 8.51964347e-03 8.08508990e-03 1.72643992e-02
 8.86948389e-03 9.22023607e-03 2.01086375e-02 2.13909906e-02
 1.05521284e-02 1.99249503e-02 1.03349188e-02 9.82126974e-03
 1.04846708e-02 3.22766724e-02 4.18291549e-02 1.12750179e-02
 1.09072460e-02 1.15831427e-02 1.14176787e-02 2.29934203e-02
 1.12501771e-02 1.20844660e-02 1.10766217e-02 1.17285929e-02
 1.13455801e-02 1.16161455e-02 1.18747446e-02 1.17746633e-02
 2.43627756e-02 1.30455569e-02 1.22276075e-02 1.23083739e-02
 1.35702072e-02 6.68355875e-02 2.64048707e-02 2.64227453e-02
 1.24333419e-02 1.27369322e-02 6.59676107e-02 1.24404248e-02
 5.50191332e-02 1.27837904e-02 1.36103441e-02 1.34238833e-02
 2.67718630e-02 1.36407694e-02 3.04637923e-02 1.38510787e-02
 1.67550858e-01 1.46863490e-02 1.49479000e-02 1.46724068e-02
 1.51344992e-02 3.01422477e-02 2.99871466e-02 2.96874341e-02
 8.16808649e-02 1.54008745e-02 3.19745942e-02 1.59666661e-02
 1.61138574e-02 1.61138574e-02 1.61354292e-02 3.41042871e-02
 1.63132576e-02 3.39910789e-02 3.56393971e-02 7.18675593e-02
 1.78924764e-02 1.74457237e-02 1.71389064e-02 1.04179133e-01
 1.97732182e-02 1.82007228e-02 2.02620792e-02 1.83663202e-02
 4.20396654e-02 7.50891528e-02 1.88696852e-02 1.89194795e-02
 1.91265005e-02 5.63168646e-02 1.92350761e-02 1.93473329e-02
 1.88206286e-02 1.92350761e-02 5.97170587e-02 3.92916929e-02
 3.99479240e-02 2.01148879e-02 2.02620792e-02 1.32024656e-01
 2.13165617e-02 1.40466044e-01 2.11158179e-02 2.34110073e-02
 6.81339249e-02 2.11158179e-02 2.17587186e-02 4.35174373e-02
 2.21339792e-02 2.27113083e-02 2.27113083e-02 2.28728651e-02
 4.64434820e-02 2.04083036e-01 2.30427105e-02 2.42992986e-02
 2.40539080e-02 2.48524578e-02 2.42992986e-02]
```

In [8]:
```python
# 20_newsgroups/talk.politics.mideast/75944
arrCnt = np.array(result.lookup("20_newsgroups/talk.politics.mideast/75944"))
TFArr = arrCnt / arrCnt.sum()
finalResult = np.multiply(TFArr, IDFArr)
print(finalResult[finalResult.nonzero()])
```

```
[5.54725111e-03 2.68605086e-03 7.39273306e-03 2.03178717e-03
 5.06707738e-03 1.94545485e-03 4.31868710e-03 2.42388609e-03
 2.46472840e-03 4.14154280e-03 1.62831955e-03 6.62550483e-03
 3.22073049e-03 2.43657927e-03 1.69721414e-03 1.15698386e-03
 1.93593964e-03 4.04824180e-03 3.94159141e-03 2.94504525e-03
 9.88048494e-04 4.42826235e-03 1.49072312e-03 5.28678488e-04
 1.04214371e-02 1.86094552e-03 1.49772974e-03 1.04087154e-03
 1.64197257e-06 2.22622265e-03 4.73573904e-05 1.23392775e-02
 1.00104738e-03 9.83171309e-04 9.65207547e-04 3.03599781e-03
 3.98260227e-03 2.13198368e-03 3.57653570e-03 1.57413222e-03
 2.17955958e-03 1.07444211e-03 5.40226952e-04 5.59517954e-04
 3.84777461e-03 1.44634481e-03 2.21205679e-03 6.24049705e-03
 5.63104209e-04 4.86079553e-03 4.90455048e-03 7.84729739e-03
 1.77626038e-03 2.30413648e-03 7.10498595e-03 1.96850758e-03
 2.16703483e-03 1.31035475e-03 1.36934217e-03 1.94164808e-03
 5.81927334e-03 2.86998439e-03 1.36267792e-03 1.45303972e-03
 3.47796655e-03 6.75110950e-03 2.96447004e-03 5.13676332e-03
 1.55299386e-03 1.39374925e-03 7.44483602e-04 1.48960041e-03
 8.19877288e-04 1.73294509e-03 1.59879298e-03 8.04183656e-04
 5.12938309e-03 2.50805724e-03 2.55637383e-03 2.43592804e-03
 8.68439688e-04 8.90132820e-04 1.51500388e-03 1.87415561e-03
 1.47175815e-03 3.73352652e-03 2.91159940e-03 8.95481969e-04
 2.74919301e-03 2.73594560e-03 3.94921536e-03 6.43284322e-03
 9.41246420e-04 2.15148451e-03 9.90881645e-04 2.99283121e-03
 2.01554418e-03 9.63339843e-04 4.03647370e-03 6.18664562e-03
 3.09437419e-03 1.14412419e-02 1.04727765e-03 6.46812994e-03
 2.94738368e-03 8.38630128e-04 3.02887238e-03 1.24561624e-03
 2.07560755e-03 1.04475964e-03 1.06858419e-03 9.54857618e-04
 3.66614576e-03 3.47542870e-03 2.19044071e-03 5.46533058e-03
 2.17407776e-03 2.28621293e-03 2.17989452e-03 2.14543455e-03
 1.21594255e-03 1.35797367e-03 1.14847082e-03 1.20463876e-03
 1.23571864e-03 3.65586326e-03 1.48294476e-03 1.19122644e-03
 1.72118288e-03 1.19472526e-03 4.10036054e-03 3.68466389e-03
 5.05531136e-03 1.23345898e-03 1.23120831e-03 6.42692941e-03
 1.26171657e-03 1.22971283e-03 2.77888071e-03 8.50936495e-03
 2.67244625e-03 2.57242105e-03 1.30665677e-03 1.32609646e-03
 1.36646971e-03 1.33113712e-03 1.39175645e-03 1.34196217e-03
 4.06737502e-03 6.66015665e-03 1.33203133e-03 1.35516979e-03
 1.47751669e-03 1.40044312e-03 2.74441105e-03 1.77340069e-03
 1.40789911e-03 1.49754646e-03 9.17519717e-03 1.79193649e-03
 1.55162439e-03 2.80764004e-03 1.41476369e-03 4.55867309e-03
 2.84412471e-03 1.46947107e-03 1.52289276e-03 4.36336743e-03
 2.94734231e-03 3.08925841e-03 4.56617143e-03 1.48000188e-02
 1.48294476e-03 1.51417984e-03 1.65512923e-03 1.48607009e-03
 3.03248439e-03 4.51193830e-03 1.53771952e-03 3.30814720e-03
 9.82160631e-03 1.65301994e-03 1.64310687e-03 3.16100856e-03
 1.45076350e-03 2.28045094e-02 6.28527927e-03 1.58375442e-03
 1.55915286e-03 1.65883955e-03 1.99742033e-03 1.66418261e-03
 5.26825408e-03 1.60462324e-03 1.63591216e-03 3.35883510e-03
 3.65182541e-03 1.64725962e-03 1.80143740e-03 1.66471970e-03
 3.48471461e-03 1.94125903e-03 1.70249386e-03 1.62229003e-03
 1.67886632e-03 5.49065092e-03 1.62981732e-03 3.51850320e-03
 7.18938406e-03 6.82825870e-02 1.83673546e-03 1.84112325e-03
 7.45846129e-03 3.46394698e-03 3.56254061e-03 1.92484759e-03
 5.31435087e-03 1.76371529e-03 1.79734601e-03 1.80212219e-03
 1.75988711e-03 1.85524697e-03 1.71351579e-03 1.81110074e-03
 1.85902314e-03 3.59062274e-03 1.59576275e-02 1.81810567e-03
 1.76886018e-03 1.74112573e-03 1.21087323e-02 1.99838912e-03
 1.81529327e-03 1.91555955e-03 7.20301378e-03 3.92903592e-03
 2.03030115e-03 2.05654965e-03 2.34281276e-03 1.89338843e-03
 1.91976248e-03 7.71648164e-03 2.03441986e-03 1.84185787e-03
 1.93256207e-03 1.94477563e-03 1.94213613e-03 1.82949731e-03
 1.91305272e-03 5.48418222e-03 8.20480615e-03 8.00912394e-03
 8.13354947e-03 5.95776745e-03 2.10946125e-03 6.04535353e-03
 1.90559822e-03 1.92145244e-03 2.04170062e-03 2.01312387e-03
 1.92229932e-03 2.20154005e-03 8.24779536e-03 2.01611747e-03
 1.98687176e-03 2.14854661e-03 2.10009803e-03 2.06630563e-03
 2.00619983e-03 2.01611747e-03 2.01113690e-03 2.01411998e-03
 2.01014603e-03 1.95906495e-03 2.11658686e-03 3.97945835e-03
 1.99548771e-03 2.15746909e-03 5.10374880e-03 2.13727797e-03
 2.08962793e-02 2.24034273e-03 2.11064263e-03 5.25250669e-03
 2.36648380e-03 2.05120154e-03 2.06521328e-03 1.10922888e-02
 2.07623295e-03 2.01212951e-03 4.13261127e-03 2.02825299e-03
 2.45923005e-03 4.92725845e-03 2.30488266e-03 2.10125999e-03
 2.10476025e-03 2.13234090e-03 1.39000653e-02 2.03961082e-03
 2.20154005e-03 2.10593184e-03 2.20993562e-03 2.50247258e-03
 6.33903847e-03 2.38155912e-03 4.51083611e-03 2.33221324e-03
 2.31159243e-03 2.26466008e-03 4.34884514e-03 2.13851901e-03
 4.69998231e-03 2.40214731e-02 2.26466008e-03 2.59476063e-03
 2.36093261e-03 4.90972969e-03 2.35543541e-03 2.07958118e-03
 4.43691552e-03 4.36484421e-03 4.39477714e-03 2.26933837e-03
 2.34459883e-03 4.41987124e-03 9.23963003e-03 2.19190026e-03
 2.32180881e-03 2.34999115e-03 4.53243042e-03 2.21702845e-03
 2.41702455e-03 2.23002422e-03 2.25085309e-03 2.55588260e-02
 4.77078324e-03 2.31667754e-03 2.37397121e-03 2.23295320e-03
 2.22132726e-03 2.40296685e-03 2.41095697e-03 2.29332887e-03
 4.66442648e-03 2.29990209e-03 2.34999115e-03 2.24482326e-03
 2.77338907e-03 2.37209004e-03 2.29496502e-03 5.41022468e-03
 2.24798561e-02 2.40894882e-03 4.71452376e-03 1.66575680e-02
 7.05540026e-03 7.23891680e-03 5.09859807e-03 2.33396608e-03
 2.62039535e-03 4.72925468e-03 2.30488266e-03 2.46806290e-03
 2.55706059e-03 5.02402716e-03 5.00968555e-03 2.47253163e-03
 2.37965257e-03 7.49328537e-03 2.46362922e-03 2.17505363e-02
 2.57559639e-03 2.37021511e-03 2.52910792e-03 2.53159272e-03
 2.45923005e-03 2.37209004e-03 2.68843588e-03 2.17228767e-02
 2.64116954e-03 4.99552358e-03 2.49308994e-03 2.39704752e-03
 2.42728420e-03 2.46362922e-03 2.35361483e-03 2.53659533e-03
 2.62039535e-03 2.42110609e-03 2.74001794e-03 2.61172064e-03
 2.69838260e-03 2.63815465e-02 2.43353057e-03 2.61172064e-03
 2.53159272e-03 2.51684516e-03 5.12458961e-03 4.94506326e-03
 8.18795431e-03 1.16250166e-02 2.55967164e-03 2.52910792e-03
 2.62920526e-03 2.65339253e-03 2.59755220e-03 2.51684516e-03
 2.54929904e-03 9.54212225e-03 2.55187440e-03 2.66905257e-03
 2.68515882e-03 2.54418324e-03 2.55967164e-03 2.87798371e-03
 2.18914302e-02 2.71535959e-03 2.76204506e-03 2.70173746e-03
 2.92093787e-03 2.60035762e-03 2.58100591e-03 2.72579603e-03
 1.72955902e-02 2.81293604e-03 2.68515882e-03 2.67223731e-03
 2.84236304e-03 2.78496540e-03 2.77338907e-03 2.65031192e-03
 2.93601319e-03 2.74001794e-03 6.08428374e-03 2.83379786e-03
 2.75831375e-03 2.74726536e-03 2.75092422e-03 2.86436158e-03
 3.33298306e-02 5.72872316e-03 2.75831375e-03 2.69838260e-03
 2.89667272e-03 2.70511234e-03 2.74363004e-03 5.54677814e-03
 2.77722157e-03 2.80885450e-03 8.29740347e-03 8.82338696e-03
 2.96206809e-03 3.04831997e-03 2.79281680e-03 3.12114805e-03
 2.99500698e-03 2.79678377e-03 2.77722157e-03 2.83379786e-03
 2.82956345e-03 2.89194223e-03 2.83379786e-03 6.16053165e-03
 2.81293604e-03 8.60659775e-03 2.76204506e-03 2.81293604e-03
 2.91110647e-03 2.82956345e-03 2.76958238e-03 2.81293604e-03
 2.81293604e-03 3.17291018e-03 2.82956345e-03 2.97829766e-03
 3.05456635e-03 6.33043804e-03 2.96206809e-03 1.21932799e-02
 5.96762547e-03 9.26063248e-03 3.12825873e-03 3.08687749e-03
 2.93601319e-03 1.22182654e-02 2.94112899e-03 2.95150159e-03
 1.30977132e-02 2.92591844e-03 3.17291018e-03 3.37564268e-03
 6.54885662e-03 1.35930294e-02 3.16521902e-03 3.00642740e-03
 3.02400263e-03 3.14275417e-03 6.10913270e-03 9.18264755e-03
 3.08026583e-03 3.06088252e-03 3.08026583e-03 6.27091997e-03
 3.02998460e-03 6.10913270e-03 3.03603055e-03 3.22139340e-03
 3.15014370e-03 3.10033744e-03 3.21301856e-03 3.20476599e-03
 3.10718954e-03 3.18070742e-03 3.20476599e-03 3.15014370e-03
 6.31526221e-03 3.16521902e-03 3.22139340e-03 3.34333154e-03
 3.12114805e-03 3.17291018e-03 6.42603712e-03 3.15014370e-03
 6.36141484e-03 3.17291018e-03 6.39326433e-03 6.47704949e-03
 3.17291018e-03 3.32277323e-03 3.18070742e-03 3.24728912e-03
 3.21301856e-03 1.38883831e-02 3.22989418e-03 3.33295894e-03
 3.45910000e-03 3.43397182e-03 3.10178184e-02 3.25619153e-03
 7.42920637e-03 3.30293642e-03 3.49901949e-03 3.37564268e-03
 6.68666308e-03 1.16316819e-02 3.38683693e-03 3.31276782e-03
 3.36466581e-03 3.42181455e-03 3.32277323e-03 1.03773000e-02
 3.32277323e-03 3.34333154e-03 3.40991324e-03 3.36466581e-03
 3.71460318e-03 1.38883831e-02 3.47209577e-03 3.54197365e-03
 3.42181455e-03 1.03773000e-02 7.20969702e-03 3.51297800e-03
 3.40991324e-03 3.45910000e-03 3.52728994e-03 7.05457987e-03
 3.44639630e-03 3.55704897e-03 3.54197365e-03 3.47209577e-03
 7.11409794e-03 3.52728994e-03 3.51297800e-03 3.75649576e-03
 3.57253737e-03 3.60484851e-03 3.71460318e-03 4.35619727e-03
 3.82580177e-03 3.71460318e-03 7.51299152e-03 3.67560213e-03
 3.73516149e-03 3.77866688e-03 3.73516149e-03 3.77866688e-03
 3.71460318e-03 3.77866688e-03 3.87722731e-03 3.90480795e-03
 3.82580177e-03 3.82580177e-03 3.90480795e-03 7.99335692e-03
 3.93380360e-03 4.14832571e-03 4.10643313e-03 4.10643313e-03
 4.03094902e-03 1.67742926e-02 1.32683369e-02 8.71239453e-03
 4.29663790e-03 8.38714629e-03 4.10643313e-03 4.14832571e-03
 8.59327580e-03]
```

In [9]:
```python
# 20_newsgroups/sci.med/58763
arrCnt = np.array(result.lookup("20_newsgroups/sci.med/58763"))
TFArr = arrCnt / arrCnt.sum()
finalResult = np.multiply(TFArr, IDFArr)
print(finalResult[finalResult.nonzero()])
```

```
[2.34482371e-03 4.14265386e-03 4.42577019e-03 2.92148526e-03
 1.33936842e-03 2.16121698e-03 1.08191180e-02 8.05687963e-03
 2.58109075e-03 8.59275516e-03 4.17703243e-03 4.12642026e-03
 9.62545664e-03 4.20143956e-03 6.31741620e-03 5.31671210e-03
 1.48492077e-02 2.34245926e-05 6.75606643e-04 2.85621422e-03
 1.39301109e-02 7.98215533e-03 5.15843543e-03 7.88937191e-03
 8.61158132e-03 6.33508993e-03 9.41006605e-03 1.04688183e-02
 1.10776054e-02 1.06254158e-02 1.15837546e-02 1.47099141e-02
 1.39793137e-02 1.47149138e-02 1.53791960e-02 1.36221220e-02
 1.58114516e-02 3.46935867e-02 3.42222836e-02 1.70775536e-02
 1.82594869e-02 1.83374808e-02 2.12790916e-02 1.99693017e-02
 2.16014850e-02 2.17616535e-02 2.77570110e-02 2.30871061e-02
 2.31437988e-02 2.52717384e-02 2.50165686e-02 5.76677542e-02
 2.58076567e-02 2.72681028e-02 3.02984539e-02 2.82909350e-02
 6.64934520e-02 3.03504273e-02 3.12699320e-02 3.30500207e-02
 6.64437477e-02 4.11898951e-02 2.22132596e-01 3.43093815e-02
 3.48071684e-02 3.87869947e-02 3.79869411e-02 4.06113152e-02
 3.78096918e-02 3.85914817e-02 8.45144911e-02 4.43275669e-02
 8.94617535e-02 4.18132151e-02 4.35768377e-02 4.37580704e-02
 4.24887787e-02 4.41332318e-02 4.76963992e-02 4.50471728e-02
 5.07453196e-02 4.99174635e-02 1.62708477e-01 5.61201497e-02
 5.70171306e-02]
```

In [27]:
```python
# Task 3
import re
import numpy as np


dictTuple = dict(dictionary.collect())

def TFIDF(arrCnt):
    TFArr = arrCnt / arrCnt.sum()
    finalResult = np.multiply(TFArr, IDFArr)
    return finalResult

def predictLabel(cnt, inputStr):
    regex = re.compile('[^a-zA-Z]')
    # Input string to a bunch of words
    words = regex.sub(' ', inputStr).lower().split()
    rdd = sc.parallelize(list(words))

    wordArr = np.zeros(20000)
    for word in words:
        if word in dictTuple.keys():
            wordArr[dictTuple[word]] += 1
    inputTFIDF = TFIDF(wordArr)

    # np.linalg.norm() is to sqare each element of the matrix, sum them up, and apply sqrt() on it
    knnDisRdd = result.map(lambda x: (x[0], np.linalg.norm(inputTFIDF - TFIDF(x[1]))))
    knnDis = knnDisRdd.collect()
    knnDis.sort(key = lambda x: x[1])

    # It then finds the k documents in the corpus that are closest to the query vector
    topCntList = list()
    for i in range(cnt):
        topCntList.append(knnDis[i][0])

    # Returns the newsgroup label that is most frequent in those top k
    categoryDict = dict()
    for newsGroup in topCntList:
        category = newsGroup.split('/')[1]
        if category not in categoryDict:
            categoryDict[category] = 1
        else:
            categoryDict[category] += 1
    mostCategory = max(categoryDict, key = categoryDict.get)

    return "20_newsgroups/" + mostCategory

predictLabel(10, 'Graphics are pictures and movies created using computers - usually referring to image
 data created by a computer specifically with help from specialized graphical hardware and software. It
 is a vast and recent area in computer science. The phrase was coined by computer graphics researchers
 Verne Hudson and William Fetter of Boeing in 1960. It is often abbreviated as CG, though sometimes err
oneously referred to as CGI. Important topics in computer graphics include user interface design, sprit
e graphics, vector graphics, 3D modeling, shaders, GPU design, implicit surface visualization with ray
 tracing, and computer vision, among others. The overall methodology depends heavily on the underlying
 sciences of geometry, optics, and physics. Computer graphics is responsible for displaying art and ima
ge data effectively and meaningfully to the user, and processing image data received from the physical
 world. The interaction and understanding of computers and interpretation of data has been made easier
 because of computer graphics. Computer graphic development has had a significant impact on many types
 of media and has revolutionized animation, movies, advertising, video games, and graphic design genera
lly.')
```

```
'20_newsgroups/comp.graphics'
```

In [28]:
```python
predictLabel(10, 'A deity is a concept conceived in diverse ways in various cultures, typically as a na
tural or supernatural being considered divine or sacred. Monotheistic religions accept only one Deity
 (predominantly referred to as God), polytheistic religions accept and worship multiple deities, henoth
eistic religions accept one supreme deity without denying other deities considering them as equivalent
 aspects of the same divine principle, while several non-theistic religions deny any supreme eternal cr
eator deity but accept a pantheon of deities which live, die and are reborn just like any other being.
 A male deity is a god, while a female deity is a goddess. The Oxford reference defines deity as a god
 or goddess (in a polytheistic religion), or anything revered as divine. C. Scott Littleton defines a d
eity as a being with powers greater than those of ordinary humans, but who interacts with humans, posit
ively or negatively, in ways that carry humans to new levels of consciousness beyond the grounded preoc
cupations of ordinary life.')
```

```
'20_newsgroups/talk.religion.misc'
```

In [29]:
```python
predictLabel(10, 'Egypt, officially the Arab Republic of Egypt, is a transcontinental country spanning
 the northeast corner of Africa and southwest corner of Asia by a land bridge formed by the Sinai Penin
sula. Egypt is a Mediterranean country bordered by the Gaza Strip and Israel to the northeast, the Gulf
 of Aqaba to the east, the Red Sea to the east and south, Sudan to the south, and Libya to the west. Ac
ross the Gulf of Aqaba lies Jordan, and across from the Sinai Peninsula lies Saudi Arabia, although Jor
dan and Saudi Arabia do not share a land border with Egypt. It is the worlds only contiguous Eurafrasia
n nation. Egypt has among the longest histories of any modern country, emerging as one of the worlds fi
rst nation states in the tenth millennium BC. Considered a cradle of civilisation, Ancient Egypt experi
enced some of the earliest developments of writing, agriculture, urbanisation, organised religion and c
entral government. Iconic monuments such as the Giza Necropolis and its Great Sphinx, as well the ruins
 of Memphis, Thebes, Karnak, and the Valley of the Kings, reflect this legacy and remain a significant
 focus of archaeological study and popular interest worldwide. Egypts rich cultural heritage is an inte
gral part of its national identity, which has endured, and at times assimilated, various foreign influe
nces, including Greek, Persian, Roman, Arab, Ottoman, and European. One of the earliest centers of Chri
stianity, Egypt was Islamised in the seventh century and remains a predominantly Muslim country, albeit
 with a significant Christian minority.')
```

```
'20_newsgroups/alt.atheism'
```

In [30]:
```python
predictLabel(10, 'The term atheism originated from the Greek atheos, meaning without god(s), used as a
 pejorative term applied to those thought to reject the gods worshiped by the larger society. With the
 spread of freethought, skeptical inquiry, and subsequent increase in criticism of religion, applicatio
n of the term narrowed in scope. The first individuals to identify themselves using the word atheist li
ved in the 18th century during the Age of Enlightenment. The French Revolution, noted for its unprecede
nted atheism, witnessed the first major political movement in history to advocate for the supremacy of
 human reason. Arguments for atheism range from the philosophical to social and historical approaches.
 Rationales for not believing in deities include arguments that there is a lack of empirical evidence;
 the problem of evil; the argument from inconsistent revelations; the rejection of concepts that cannot
 be falsified; and the argument from nonbelief. Although some atheists have adopted secular philosophie
s (eg. humanism and skepticism), there is no one ideology or set of behaviors to which all atheists adh
ere.')
```

```
'20_newsgroups/alt.atheism'
```

In [31]:
```python
predictLabel(10, 'President Dwight D. Eisenhower established NASA in 1958 with a distinctly civilian (r
ather than military) orientation encouraging peaceful applications in space science. The National Aeron
autics and Space Act was passed on July 29, 1958, disestablishing NASAs predecessor, the National Advis
ory Committee for Aeronautics (NACA). The new agency became operational on October 1, 1958. Since that
 time, most US space exploration efforts have been led by NASA, including the Apollo moon-landing missi
ons, the Skylab space station, and later the Space Shuttle. Currently, NASA is supporting the Internati
onal Space Station and is overseeing the development of the Orion Multi-Purpose Crew Vehicle, the Space
 Launch System and Commercial Crew vehicles. The agency is also responsible for the Launch Services Pro
gram (LSP) which provides oversight of launch operations and countdown management for unmanned NASA lau
nches.')
```

```
'20_newsgroups/sci.space'
```

In [32]:
```python
predictLabel(10, 'The transistor is the fundamental building block of modern electronic devices, and is
 ubiquitous in modern electronic systems. First conceived by Julius Lilienfeld in 1926 and practically
 implemented in 1947 by American physicists John Bardeen, Walter Brattain, and William Shockley, the tr
ansistor revolutionized the field of electronics, and paved the way for smaller and cheaper radios, cal
culators, and computers, among other things. The transistor is on the list of IEEE milestones in electr
onics, and Bardeen, Brattain, and Shockley shared the 1956 Nobel Prize in Physics for their achievemen
t.')
```

```
'20_newsgroups/talk.politics.misc'
```

In [33]:
```python
predictLabel(10, 'The Colt Single Action Army which is also known as the Single Action Army, SAA, Model
 P, Peacemaker, M1873, and Colt .45 is a single-action revolver with a revolving cylinder holding six m
etallic cartridges. It was designed for the U.S. government service revolver trials of 1872 by Colts Pa
tent Firearms Manufacturing Company - todays Colts Manufacturing Company - and was adopted as the stand
ard military service revolver until 1892. The Colt SAA has been offered in over 30 different calibers a
nd various barrel lengths. Its overall appearance has remained consistent since 1873. Colt has disconti
nued its production twice, but brought it back due to popular demand. The revolver was popular with ran
chers, lawmen, and outlaws alike, but as of the early 21st century, models are mostly bought by collect
ors and re-enactors. Its design has influenced the production of numerous other models from other compa
nies.')
```

```
'20_newsgroups/talk.politics.guns'
```

In [34]:
```python
predictLabel(10, 'Howe was recruited by the Red Wings and made his NHL debut in 1946. He led the league
 in scoring each year from 1950 to 1954, then again in 1957 and 1963. He ranked among the top ten in le
ague scoring for 21 consecutive years and set a league record for points in a season (95) in 1953. He w
on the Stanley Cup with the Red Wings four times, won six Hart Trophies as the leagues most valuable pl
ayer, and won six Art Ross Trophies as the leading scorer. Howe retired in 1971 and was inducted into t
he Hockey Hall of Fame the next year. However, he came back two years later to join his sons Mark and M
arty on the Houston Aeros of the WHA. Although in his mid-40s, he scored over 100 points twice in six y
ears. He made a brief return to the NHL in 1979-80, playing one season with the Hartford Whalers, then
 retired at the age of 52. His involvement with the WHA was central to their brief pre-NHL merger succe
ss and forced the NHL to expand their recruitment to European talent and to expand to new markets.')
```

```
'20_newsgroups/talk.politics.mideast'
```