# COMP 543 Assignment #4

Rung-De Chu (rc118)

## Task 1

```
from pyspark import SparkContext
sc = SparkContext()

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)

sortedWords = allCounts.sortBy(lambda x: (-x[1], x[0]))

# and get the top 20,000 words in a local array
# each entry is a ("word1", count) pair
topWords = sortedWords.take(20000)

# 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
# HINT: make use of topWords in the lambda that you supply
dictionary = twentyK.map (lambda x:(topWords[x][0],x))

def doc_to_vec(word_list):
    vec = np.zeros(20000)
    for i in word_list:
        vec[i] += 1
    return vec

pairs_word_doc = keyAndListOfWords.flatMap(lambda doc: ((word, doc[0]) for
word in doc[1]))
doc_word_pairs = dictionary.join(pairs_word_doc).map(lambda x: (x[1][1],
x[1][0]))
final_counts = (doc_word_pairs.groupByKey().map(lambda doc: (doc[0],
doc_to_vec(list(doc[1])))))

result_1 = final_counts.lookup('20_newsgroups/comp.graphics/37261')
result_2 =
final_counts.lookup('20_newsgroups/talk.politics.mideast/75944')
result_3 = final_counts.lookup('20_newsgroups/sci.med/58763')
```

```
print(result_1[0][result_1[0].nonzero()])
print(result_2[0][result_2[0].nonzero()])
print(result_3[0][result_3[0].nonzero()])
```

```
>>> result_1 = final_counts.lookup('20_newsgroups/comp.graphics/37261')
>>> print(result_1[0][result_1[0].nonzero()])
[ 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.  2.  1.  2.  1.  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.]
```

```
>>> result_2 = final_counts.lookup('20_newsgroups/talk.politics.mideast/75944')
>>> print(result_2[0][result_2[0].nonzero()])
[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.   2.   1.   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.   4.   1.   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.   2.   1.   1.   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.   1.   2.   1.   1.   3.   8.   1.   1.
```

```
 1.  1.  1.  8.  1.  1.  1.  2.  1.  1.  1.  1.  1.  1.
 1.  1.  1.  1.  1. 10.  1.  1.  1.  2.  1.  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.  2.  1.  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.  2.  1.  1.  1.  3.  1.  1.  1.  1.  1.  1.
 1.  1.  1.  1.  2.  1.  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.  1.  2.  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.  2.  1.  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.  2.  1.  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.  2.  3.  2.  1.  1.  1.]
```

```
>>> result_3 = final_counts.lookup('20_newsgroups/sci.med/58763')
>>> print(result_3[0][result_3[0].nonzero()])
[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. 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.]
```

## Task 2

```
from math import log
docFreq = keyAndListOfWords.flatMap(lambda x: set((word, 1) for word in
x[1])).reduceByKey(lambda x, y: x + y)

corpusSize = 19997
broadcastDF = sc.broadcast(docFreq.collectAsMap())
idf = docFreq.map(lambda x: (x[0], log(corpusSize / x[1]))).collectAsMap()

broadcastIDF = sc.broadcast(idf)
broadcastDict = sc.broadcast(dictionary.collectAsMap())

def TFIDF(doc, broadcastDict, broadcastIDF):
    vec = np.zeros(20000)
    idfMap = broadcastIDF.value
    for word in doc:
        if word in broadcastDict.value:
            tf = doc.count(word) / len(doc)
```

```
            idf = idfMap.get(word, 0)
            vec[broadcastDict.value[word]] = tf * idf
    return vec

doc_to_vec_TFIDF = keyAndListOfWords.map(lambda x: (x[0], TFIDF(x[1],
broadcastDict, broadcastIDF)))

result_1 = tf_idf.lookup('20_newsgroups/comp.graphics/37261')
result_2 = tf_idf.lookup('20_newsgroups/talk.politics.mideast/75944')
result_3 = tf_idf.lookup('20_newsgroups/sci.med/58763')

print(result_1[0][result_1[0].nonzero()])
print(result_2[0][result_2[0].nonzero()])
print(result_3[0][result_3[0].nonzero()])
```

```
>>> result_1 = doc_to_vec_TFIDF.lookup('20_newsgroups/comp.graphics/37261')
>>> print(result_1[0][result_1[0].nonzero()])
[1.84608343e-03 8.15379489e-04 3.50844930e-03 1.22252772e-03
 6.96883179e-03 2.10897377e-03 2.55229434e-03 6.09629053e-03
 3.28858426e-03 1.58855652e-03 9.05999264e-03 3.78906864e-03
 4.02463216e-03 4.97371181e-03 6.27878381e-03 9.22110945e-06
 9.03858461e-03 2.76068261e-03 5.15830388e-04 5.89341779e-03
 9.10153789e-03 1.44057424e-02 3.23493446e-03 4.73917171e-03
 5.73788574e-03 4.36070816e-03 4.78542361e-03 1.59057147e-02
 4.70964030e-03 6.98804191e-03 6.10467231e-03 1.28390815e-02
 6.02424401e-03 6.84499230e-03 6.84362128e-03 1.53158727e-02
 6.15392589e-03 7.33161808e-03 1.56058412e-02 8.22259414e-03
 7.33161808e-03 8.61162383e-03 7.80106680e-03 7.57057246e-03
 8.53364306e-03 8.16803914e-03 7.75141952e-03 1.65519001e-02
 8.50344170e-03 8.83971839e-03 1.92787572e-02 2.05081878e-02
 1.01166437e-02 1.91026508e-02 9.90839837e-03 9.41594749e-03
 1.00519701e-02 3.09446192e-02 4.01028723e-02 1.08096997e-02
 1.04571057e-02 1.11051082e-02 1.09464729e-02 2.20444855e-02
 1.07858841e-02 1.15857420e-02 1.06194912e-02 1.12445558e-02
 1.08773498e-02 1.11367490e-02 1.13846757e-02 1.12887249e-02
 2.33573277e-02 1.25071689e-02 1.17229761e-02 1.18004093e-02
 1.30101669e-02 6.40772934e-02 2.53151459e-02 2.53322828e-02
 1.19202198e-02 1.22112810e-02 6.32451379e-02 1.19270104e-02
 5.27485023e-02 1.22562054e-02 1.30486474e-02 1.28698818e-02
 2.56669925e-02 1.30778170e-02 2.92065564e-02 1.32794469e-02
 1.60636060e-01 1.40802457e-02 1.43310026e-02 1.40668789e-02
 1.45099008e-02 2.88982820e-02 2.87495818e-02 2.84622384e-02
 7.83099086e-02 1.47652829e-02 3.06550078e-02 1.53077244e-02
 1.54488411e-02 1.54488411e-02 3.26968086e-02 1.54695226e-02
```

```
 3.25882725e-02 1.56400121e-02 3.41685649e-02 6.89015965e-02
 1.71540567e-02 1.67257414e-02 1.64315864e-02 9.98796766e-02
 1.89571807e-02 1.74495819e-02 1.94258664e-02 1.76083451e-02
 4.03046951e-02 7.19902354e-02 1.81386756e-02 1.80909363e-02
 1.83371529e-02 5.39926765e-02 1.84412476e-02 1.80439042e-02
 1.85488715e-02 1.84412476e-02 5.72525452e-02 3.76701310e-02
 3.82992796e-02 1.92847497e-02 1.94258664e-02 1.26576019e-01
 2.04368306e-02 1.34669032e-01 2.02443715e-02 2.24448387e-02
 6.53220486e-02 2.02443715e-02 2.08607398e-02 4.17214796e-02
 2.12205134e-02 2.17740162e-02 2.19289056e-02 2.17740162e-02
 4.45267669e-02 1.95660561e-01 2.20917415e-02 2.32964704e-02
 2.38268009e-02 2.30612070e-02 2.32964704e-02]

>>> result_2 = doc_to_vec_TFIDF.lookup('20_newsgroups/talk.politics.mideast/75944')
>>> print(result_2[0][result_2[0].nonzero()])
[5.33319957e-03 2.58240433e-03 7.10746999e-03 1.95338669e-03
 4.87155429e-03 1.87038567e-03 4.15204211e-03 2.33035570e-03
 2.36962204e-03 3.98173327e-03 1.56548765e-03 6.36984676e-03
 3.09645230e-03 2.34255909e-03 1.63172381e-03 1.11233938e-03
 1.86123762e-03 3.89203247e-03 3.78949740e-03 2.83140492e-03
 9.49922710e-04 4.25738918e-03 1.43320065e-03 5.08278395e-04
 1.00193055e-02 1.78913729e-03 1.43993690e-03 1.00070747e-03
 1.57861385e-06 2.14031949e-03 4.55300129e-05 1.18631423e-02
 9.62420010e-04 9.45233721e-04 9.27963125e-04 2.91884789e-03
 3.82892577e-03 2.04971691e-03 3.43852807e-03 1.51339125e-03
 2.09545701e-03 1.03298265e-03 5.19381238e-04 5.37927859e-04
 3.69930070e-03 1.39053477e-03 2.12670025e-03 5.99969526e-03
 5.41375732e-04 4.67323223e-03 4.71529880e-03 7.54449407e-03
 1.70771989e-03 2.21522686e-03 6.83082616e-03 1.89254886e-03
 2.08341555e-03 1.25979215e-03 1.31650342e-03 1.86672579e-03
 5.59472530e-03 2.75924043e-03 1.31009633e-03 1.39697134e-03
 3.34376241e-03 6.49060473e-03 2.85008017e-03 4.93855125e-03
 1.49306855e-03 1.33996871e-03 7.15756245e-04 1.43212126e-03
 7.88240719e-04 1.66607601e-03 1.53710042e-03 7.73152656e-04
 4.93145580e-03 2.41127894e-03 2.45773115e-03 2.34193299e-03
 8.34929244e-04 8.55785303e-04 1.45654449e-03 1.80183765e-03
 1.41496749e-03 3.58946110e-03 2.79924964e-03 8.60928045e-04
 2.64311002e-03 2.63037379e-03 3.79682716e-03 6.18461938e-03
 9.04926585e-04 2.06846527e-03 9.52646538e-04 2.87734696e-03
 1.93777047e-03 9.26167490e-04 3.88071846e-03 5.94792179e-03
 2.97497170e-03 1.09997592e-02 1.00686639e-03 6.21854449e-03
 2.83365311e-03 8.06269943e-04 2.91199741e-03 1.19755170e-03
 1.99551617e-03 1.00444555e-03 1.02735078e-03 9.18012569e-04
 3.52468036e-03 3.34132248e-03 2.10591827e-03 5.25444010e-03
 2.09018671e-03 2.19799493e-03 2.09577902e-03 2.06264876e-03
 1.16902303e-03 1.30557360e-03 1.10415482e-03 1.15815542e-03
```

1.18803602e-03 3.51479462e-03 1.42572244e-03 1.14526063e-03
1.65476767e-03 1.14862445e-03 3.94214011e-03 3.54248393e-03
4.86024228e-03 1.18586355e-03 1.18369973e-03 6.17893376e-03
1.21303077e-03 1.18226196e-03 2.67165216e-03 8.18101446e-03
2.56932469e-03 2.47315915e-03 1.25623686e-03 1.27492643e-03
1.31374181e-03 1.27977259e-03 1.33805280e-03 1.29017994e-03
3.91042740e-03 6.40316148e-03 1.28063230e-03 1.30287791e-03
1.42050382e-03 1.34640428e-03 2.63851258e-03 1.70497056e-03
1.35357257e-03 1.43976070e-03 8.82115423e-03 1.72279111e-03
1.49175193e-03 2.69930176e-03 1.36017226e-03 4.38276777e-03
2.73437859e-03 1.41276865e-03 1.46412896e-03 4.19499836e-03
2.83361334e-03 2.97005333e-03 4.38997677e-03 1.42289311e-02
1.42572244e-03 1.45575225e-03 1.59126283e-03 1.42872716e-03
2.91547005e-03 4.33783633e-03 1.47838361e-03 3.18049587e-03
9.44262042e-03 1.58923493e-03 1.57970438e-03 3.03903486e-03
1.39478295e-03 2.19245528e-02 6.04274947e-03 1.52264216e-03
1.49898989e-03 1.59482998e-03 1.92034596e-03 1.59996686e-03
5.06496819e-03 1.54270571e-03 1.57278729e-03 3.22922788e-03
3.51091258e-03 1.58369689e-03 1.73192542e-03 3.35025008e-03
1.60048323e-03 1.86635175e-03 1.63679981e-03 1.55969079e-03
1.61408398e-03 5.27878341e-03 1.56692763e-03 3.38273487e-03
6.91196761e-03 6.56477698e-02 1.76586142e-03 1.77007991e-03
7.17066197e-03 3.33028381e-03 3.42507301e-03 1.85057358e-03
5.10928625e-03 1.69565888e-03 1.72799190e-03 1.73258378e-03
1.69197843e-03 1.78365864e-03 1.64739643e-03 1.74121587e-03
1.78728910e-03 3.45207154e-03 1.53418712e-02 1.74795051e-03
1.70060525e-03 1.67394099e-03 1.16414932e-02 1.92127737e-03
1.74524663e-03 1.84164394e-03 6.92507140e-03 3.77742638e-03
1.95195801e-03 1.97719366e-03 2.25241074e-03 1.82032834e-03
7.41872610e-03 1.84568469e-03 1.95591779e-03 1.77078618e-03
1.85799038e-03 1.86973265e-03 1.86719501e-03 1.75890258e-03
1.83923383e-03 5.27256432e-03 7.88820765e-03 7.70007623e-03
7.81970055e-03 5.72787534e-03 2.02806356e-03 5.81208173e-03
1.83206698e-03 1.84730944e-03 1.93544354e-03 1.96291761e-03
1.84812364e-03 2.11658932e-03 7.92953804e-03 1.93832164e-03
1.91020442e-03 2.06564073e-03 2.01906164e-03 1.98657319e-03
1.92878669e-03 1.93353325e-03 1.93832164e-03 1.93640122e-03
1.93258062e-03 1.88347059e-03 2.03491421e-03 3.82590316e-03
1.91848791e-03 2.07421892e-03 4.90681066e-03 2.05480692e-03
2.00899555e-02 2.15389472e-03 2.02919935e-03 5.04982844e-03
2.27516839e-03 1.97205191e-03 1.98552299e-03 1.06642711e-02
1.99611744e-03 1.93448756e-03 3.97314638e-03 1.94998888e-03
2.36433585e-03 4.73713054e-03 2.21594426e-03 2.02017876e-03
2.02354396e-03 2.05006035e-03 1.33637040e-02 1.96090845e-03
2.11658932e-03 2.02467034e-03 2.12466093e-03 2.40590978e-03
6.09443427e-03 2.28966200e-03 4.33677667e-03 2.24222023e-03

2.22239511e-03 2.17727374e-03 4.18103644e-03 2.05600007e-03
4.51862429e-03 2.30945576e-02 2.17727374e-03 2.49463672e-03
2.26983141e-03 4.72027816e-03 2.26454633e-03 1.99933647e-03
4.26570845e-03 4.19641816e-03 4.22519607e-03 4.24932186e-03
2.18177151e-03 2.25412790e-03 8.88310083e-03 2.10732150e-03
2.23221728e-03 2.25931215e-03 4.35753773e-03 2.13148007e-03
2.32375893e-03 2.14397437e-03 2.16399952e-03 2.45725887e-02
4.58669323e-03 2.22728401e-03 2.28236688e-03 2.14679033e-03
2.13561301e-03 2.31024367e-03 2.31792547e-03 2.20483629e-03
4.48444045e-03 2.21115587e-03 2.25931215e-03 2.15820237e-03
2.28055830e-03 2.66637243e-03 2.20640931e-03 5.20146057e-03
2.16124269e-02 2.31599482e-03 4.53260464e-03 1.60148032e-02
6.78315384e-03 6.95958903e-03 4.90185869e-03 2.51928227e-03
2.24390543e-03 4.54676714e-03 2.21594426e-03 2.37282787e-03
2.45839141e-03 4.83016525e-03 2.37712416e-03 4.81637703e-03
2.28782902e-03 2.36856527e-03 7.20414230e-03 2.09112493e-02
2.47621196e-03 2.27875572e-03 2.43151734e-03 2.43390626e-03
2.36433585e-03 2.08846570e-02 2.58469732e-03 2.28055830e-03
2.53925485e-03 4.80276153e-03 2.39688918e-03 2.30455275e-03
2.33362269e-03 2.36856527e-03 2.26279599e-03 2.51928227e-03
2.43871584e-03 2.32768298e-03 2.63428899e-03 2.51094229e-03
2.59426023e-03 2.53635629e-02 2.33962803e-03 2.51094229e-03
2.43390626e-03 4.92684730e-03 2.41972777e-03 4.75424832e-03
7.87200608e-03 1.11764426e-02 2.43151734e-03 2.46090170e-03
2.52775224e-03 2.55100619e-03 2.49732057e-03 2.41972777e-03
2.45092935e-03 9.17392080e-03 2.45340533e-03 2.56606195e-03
2.44601096e-03 2.58154671e-03 2.46090170e-03 2.76693108e-03
2.10467066e-02 2.65546614e-03 2.61058213e-03 2.59748563e-03
2.80822777e-03 2.50001774e-03 2.48141275e-03 2.62061586e-03
1.66282060e-02 2.70439340e-03 2.58154671e-03 2.56912381e-03
2.73268490e-03 2.67750206e-03 2.66637243e-03 2.54804445e-03
2.82272138e-03 5.84950974e-03 2.63428899e-03 2.72445022e-03
2.65187882e-03 2.64125675e-03 2.64477442e-03 2.75383458e-03
3.20437339e-02 5.50766917e-03 2.65187882e-03 2.59426023e-03
2.78489894e-03 2.60073029e-03 2.63776170e-03 5.33274486e-03
2.67005704e-03 2.70046935e-03 7.97723192e-03 8.48291931e-03
2.93069458e-03 2.84777090e-03 2.68505050e-03 3.00071245e-03
2.87943878e-03 2.68886439e-03 2.67005704e-03 2.72445022e-03
2.72037921e-03 5.92281548e-03 2.70439340e-03 2.78035098e-03
2.72445022e-03 8.27449533e-03 2.65546614e-03 2.70439340e-03
2.79877573e-03 2.72037921e-03 2.66271263e-03 2.70439340e-03
2.70439340e-03 3.05047723e-03 2.72037921e-03 2.86337421e-03
6.08616570e-03 2.93669993e-03 2.84777090e-03 1.17227783e-02
5.73735297e-03 8.90329286e-03 3.00754874e-03 2.96776429e-03
2.82272138e-03 1.17467997e-02 2.82763977e-03 2.83761213e-03
1.25923123e-02 3.24538691e-03 2.81301615e-03 3.05047723e-03

```
 6.29615617e-03 1.30685157e-02 3.04308285e-03 2.89041851e-03
 2.90731557e-03 3.02148485e-03 5.87339986e-03 8.82831713e-03
 2.96140774e-03 2.94277238e-03 2.96140774e-03 6.02894425e-03
 2.91306672e-03 2.91887937e-03 5.87339986e-03 3.09708963e-03
 2.98070486e-03 3.02858924e-03 3.08903795e-03 3.08110382e-03
 3.05797360e-03 2.98729255e-03 3.08110382e-03 3.02858924e-03
 6.07157546e-03 3.04308285e-03 3.21432255e-03 3.00071245e-03
 3.09708963e-03 3.05047723e-03 6.17807591e-03 3.02858924e-03
 6.11594720e-03 3.05047723e-03 6.14656772e-03 6.22711986e-03
 3.19455753e-03 3.05047723e-03 3.05797360e-03 3.12198612e-03
 3.08903795e-03 1.33524727e-02 3.10526239e-03 3.20435019e-03
 3.32562386e-03 3.30146530e-03 2.98209352e-02 7.14253590e-03
 3.17548616e-03 3.13054501e-03 3.36400298e-03 3.24538691e-03
 6.42864510e-03 1.11828507e-02 3.25614920e-03 3.18493819e-03
 3.23483360e-03 3.19455753e-03 3.28977714e-03 9.97687159e-03
 3.19455753e-03 3.27833507e-03 3.21432255e-03 3.23483360e-03
 3.57126795e-03 1.33524727e-02 3.40529966e-03 3.33811817e-03
 3.28977714e-03 9.97687159e-03 6.93149676e-03 3.37742287e-03
 3.27833507e-03 3.31341035e-03 6.78236510e-03 3.32562386e-03
 3.39118255e-03 3.33811817e-03 3.40529966e-03 3.41979328e-03
 6.83958655e-03 3.39118255e-03 3.37742287e-03 3.61154402e-03
 3.43468402e-03 3.46574838e-03 3.57126795e-03 4.18810487e-03
 3.57126795e-03 3.67817572e-03 7.22308805e-03 3.53377183e-03
 3.59103298e-03 3.63285963e-03 3.59103298e-03 3.63285963e-03
 3.57126795e-03 3.63285963e-03 3.72761690e-03 3.67817572e-03
 3.75413330e-03 3.67817572e-03 3.75413330e-03 7.68491761e-03
 3.78201009e-03 3.98825445e-03 3.94797838e-03 3.94797838e-03
 3.87540697e-03 1.61270237e-02 8.37620974e-03 1.27563522e-02
 8.06351184e-03 3.94797838e-03 4.13084372e-03 4.62882782e-03]

>>> result_3 = doc_to_vec_TFIDF.lookup('20_newsgroups/sci.med/58763')
>>> print(result_3[0][result_3[0].nonzero()])
[2.21952778e-03 3.92129067e-03 4.18927865e-03 2.76537536e-03
 1.26779912e-03 2.04573211e-03 1.02409972e-02 7.62635934e-03
 2.44316987e-03 8.13360031e-03 3.95383223e-03 3.90592453e-03
 9.11111926e-03 3.97693516e-03 5.97984435e-03 5.03261298e-03
 1.40557386e-02 2.21728968e-05 6.39505524e-04 2.70359209e-03
 1.31857538e-02 7.55562794e-03 4.88279385e-03 7.46780242e-03
 8.15142049e-03 5.99657367e-03 8.90723809e-03 9.90941582e-03
 1.04856723e-02 1.00576455e-02 1.09647753e-02 1.39238882e-02
 1.32323274e-02 1.39286207e-02 1.45574069e-02 1.28942223e-02
 1.49665649e-02 3.28397310e-02 3.23936119e-02 1.61650126e-02
 1.72837891e-02 1.73576155e-02 2.01420409e-02 1.89022398e-02
 2.04472071e-02 2.05988170e-02 2.62738119e-02 2.18534439e-02
 2.19071073e-02 2.39213402e-02 2.36798054e-02 5.45862711e-02
 2.44286216e-02 2.58110286e-02 2.86794525e-02 2.67792057e-02
```

6.29403668e-02 2.87286488e-02 2.95990196e-02 3.12839891e-02
6.28933185e-02 3.89889084e-02 2.10262916e-01 3.24760557e-02
3.29472434e-02 3.67144072e-02 3.59571045e-02 3.84412450e-02
3.57893266e-02 3.65293414e-02 7.99984496e-02 4.19589183e-02
8.46813544e-02 3.95789211e-02 4.12483044e-02 4.14198529e-02
4.02183859e-02 4.17749675e-02 4.51477366e-02 4.26400720e-02
4.80337376e-02 4.72501181e-02 1.54014131e-01 5.31213631e-02
5.39704137e-02]

## Task 3

```
from collections import Counter
def predictLabel(k, text):
    preprocess_text = re.sub('[^a-zA-Z]', ' ', text).lower().split()
    input_vec = TFIDF(preprocess_text , broadcastDict, broadcastIDF)
    broadcastVec = sc.broadcast(input_vec)
    distances = doc_to_vec_TFIDF.map(lambda doc: (doc[0],
np.linalg.norm(doc[1] - broadcastVec.value)))
    nearest_docs = distances.takeOrdered(k, key=lambda x: x[1])

    nearest_labels = [doc_id.split('/')[1] for doc_id, _ in nearest_docs]
    most_common_label, _ = Counter(nearest_labels).most_common(1)[0]
    return most_common_label
```

comp.graphics
talk.religion.misc
alt.atheism
alt.atheism
sci.space
talk.politics.misc
talk.politics.guns
talk.politics.mideast