

TF-IDF “Term Frequency — Inverse Document Frequency”

1. (Building Index) Compute TFIDF scores for all words in all documents and build an inverted index

Code and Explanation:

Mapper File for Inverted index and TFIDF:

```
#!/usr/bin/python
```

```
import os
import sys
```

```
# Read pairs as lines of input from STDIN
```

```
for line in sys.stdin:
```

```
    line = line.strip()
```

```
    words = line.split()
```

```
    for word in words:
```

```
        try:
```

```
            filename = os.environ['mapreduce_map_input_file'] ##Get the input filename
```

```
        except KeyError:
```

```
            filename = os.environ['map_input_file']
```

```
        word_file = word + "_" + filename[-7:-4] #remove last 4 char .txt and take only 100 from
```

```
/path/100.txt
```

```
        #print '%s\t%s\t%s' % (word,1,filename)
```

```
        print ("%s\t%s" % ( word_file, 1))
```

Reducer File for Inverted index and TFIDF:

```
#!/usr/bin/env python

from __future__ import division
import sys
import math

word_hits={} ##store all word as key and list all docs containing the word as value

##Inverse_index dictionary:For all word in all document, key is doc and value(another dictionary) is each word in
the doc along with their frequency, tf, idf, tf_idf
##e.g., inverse_index={"doc1":{"word":[hits,freq,tf,tf_idf]}}
```

```
inverse_index={}

for line in sys.stdin:
    word_file, count = line.strip().split("\t", 1)
    word, doc = word_file.split("_")

    try:
        count = int(count)
    except ValueError:
        continue

    if word not in word_hits:
        doc_list=[]
        doc_list.append(doc) #which doc has the word, add it to the list
        word_hits[word] = [count,doc_list]
    else:
        current_count = word_hits[word][0]
        current_doc_list=[]
        current_doc_list = word_hits[word][1]
        if doc not in current_doc_list:
            current_doc_list.append(doc)
        word_hits[word] = [current_count+1, current_doc_list]

    if doc not in inverse_index:
        word_dict={}
        word_list=[0,1,0,0]

        word_dict[word]=word_list

        inverse_index[doc]=word_dict
```

```

else:
    word_dict={}
    word_dict=inverse_index[doc]

    word_list=[0,1,0,0]

    if word not in word_dict:
        word_dict[word]=word_list
    else:
        current_doc_count=word_dict[word][1]
        word_dict[word]=[0,current_doc_count+1,0,0]
    inverse_index[doc]=word_dict ##Add inner_doc_dict to doc key as value

print '\n\n----Inverse Index-----\n\n'

for word,count_doc_list in word_hits.items():
    print '%s\t%s\t%s\n' %(word, count_doc_list[0], str(count_doc_list[1]))

print '\n\n----TF-IDF RELEVENCE SCORE---\n\n'

for doc,doc_dict in inverse_index.items():
    print '\n%s\t%s\t%s\t%s\t%s\t%s\t%s' %(doc,'HITS','FREQ','TF','IDF','TF-IDF')

    N=len(inverse_index) ##Number of documents in the directory

    ##GET Total Word in each document
    total_word_per_doc=0
    for word_count in doc_dict.values():
        total_word_per_doc+=word_count[1]

    ##for each word in a doc
    for word_key,word_value_list in doc_dict.items():

        #hits: How many doc contain the word
        hits=len(word_hits[word_key][1])

        #frequency: how many times a word appear in a doc
        frequency=word_value_list[1]

        #tf: frequency: how many times a word appear in a doc/total_word_per_doc
        tf=frequency/total_word_per_doc

        #idf: log(total number of docs/hits: how many doc contain the word))
        idf=math.log10(N/hits)

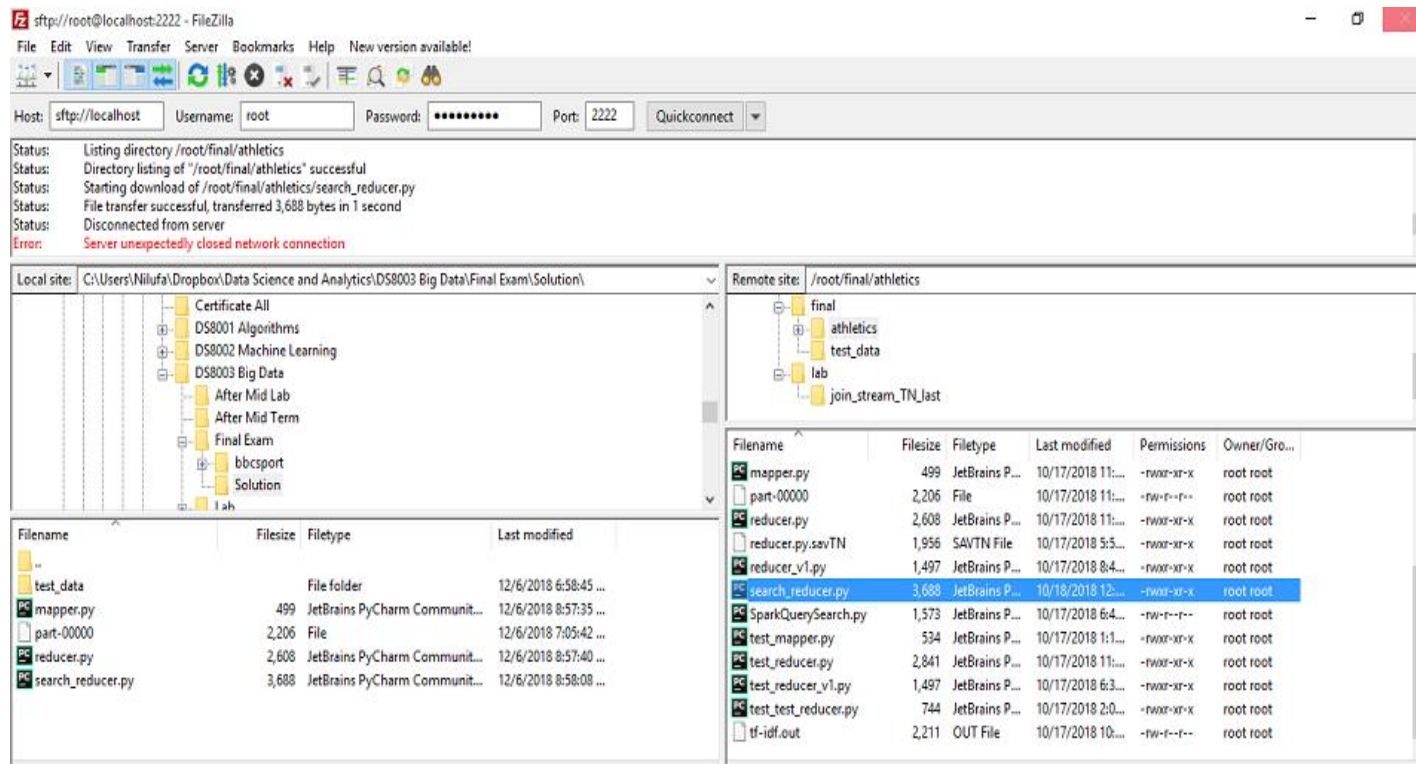
```

#tf_idf: Term Frequency (TF) * idf
 tf_idf=tf*idf

print '%s\t%s\t%s\t%s\t%s\t%s' % (word_key,hits,frequency,tf,idf,tf_idf)

ScreenShot:

FileZilla file and folder



Hadoop Streaming for Question 1:

```

localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help
[root@sandbox-hdp athletics]# clear

[root@sandbox-hdp athletics]# hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar -file /root/final/athletics/mapper.py -mapper mapper.py -file /root/final/athletics/reducer.py -reducer reducer.py -input /user/root/athletics/data/*.txt -output /user/root/final_exam_output_final
18/10/18 05:12:11 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/root/final/athletics/mapper.py, /root/final/athletics/reducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob2850681100227674869.jar tmpDir=null
18/10/18 05:12:12 INFO client.RMPProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
18/10/18 05:12:13 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
18/10/18 05:12:13 INFO client.RMPProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
18/10/18 05:12:13 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
18/10/18 05:12:14 INFO mapred.FileInputFormat: Total input paths to process : 101
18/10/18 05:12:15 INFO mapreduce.JobSubmitter: number of splits:101
18/10/18 05:12:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1539213858901_0238
18/10/18 05:12:16 INFO impl.YarnClientImpl: Submitted application application_1539213858901_0238
18/10/18 05:12:16 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8

```

OutputFile for first question answer:

```

localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help
Reduce input records=31317
Reduce output records=30814
Spilled Records=62634
Shuffled Maps =101
Failed Shuffles=0
Merged Map outputs=101
GC time elapsed (ms)=127604
CPU time spent (ms)=144510
Physical memory (bytes) snapshot=20408516608
Virtual memory (bytes) snapshot=197477912576
Total committed heap usage (bytes)=11383865344
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=182744
File Output Format Counters
Bytes Written=1337775
18/10/18 05:19:27 INFO streaming.StreamJob: Output directory: /user/root/final_exam_output_final
[root@sandbox-hdp athletics]#

```

Result for first question (Inverted Index):

```
localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

----Inverse Index-----

"Knowing      1      ['041']
four    21      ['015', '016', '022', '023', '026', '033', '034', '036', '041', '045', '061', '067',
'069', '073', '079', '081', '082', '083', '093']
Olympics    22      ['015', '017', '022', '034', '036', '043', '047', '048', '050', '051', '066',
'068', '072', '081', '082', '092', '096']
hanging 2      ['034', '066']
candidates. 1      ['043']
Ronald 2      ['058', '070']
innocence. 1      ['045']
Euro 3      ['012', '020', '073']
regional 2      ['021', '043']
```

Result for first question (TFIDF):

```
localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

----TF-IDF RELEVANCE SCORE---

098  HITS  FREQ  TF      IDF      TF-IDF
gold  37    1      0.00537634408602  0.436119649716  0.00234472929955
Union 19    1      0.00537634408602  0.72556777283   0.00390090200446
Trials 1    1      0.00537634408602  2.00432137378   0.0107759213644
indoor 30   1      0.00537634408602  0.527200119063  0.00283440924227
Maurice," 1    1      0.00537634408602  2.00432137378   0.0107759213644
Gardener 16   1      0.00537634408602  0.800201391127  0.00430215801681
European 41   2      0.010752688172  0.391537517063  0.00421008082863
4x100m 13   1      0.00537634408602  0.890378021476  0.00478697861008
fit 10    1      0.00537634408602  1.00432137378   0.0053995772784
personal 19   1      0.00537634408602  0.72556777283   0.00390090200446
Greene, 3    1      0.00537634408602  1.52720011906   0.0082107533283
to 101    5      0.0268817204301  0.0 0.0
Championships 27 1      0.00537634408602  0.572957609624  0.00308041725604
treatment 3    1      0.00537634408602  1.52720011906   0.0082107533283
has 77    1      0.00537634408602  0.11783064861   0.000633498110807
hope 7     1      0.00537634408602  1.15922333377   0.00623238351488
his 49    3      0.0161290322581  0.314125293754  0.00506653699603
```

Result for Result for first question Test data Streamin:

```
localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

[root@sandbox-hdp athletics]# hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar -file /root/final/athletics/mapper.py -mapper mapper.py -file /root/final/athletics/reducer.py -reducer reducer.py -input /user/root/test_data/*.txt -output /user/root/final_exam_output_test_Final
18/10/18 05:31:09 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/root/final/athletics/mapper.py, /root/final/athletics/reducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob5991646472179977972.jar
tmpDir=null
18/10/18 05:31:12 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
18/10/18 05:31:13 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
18/10/18 05:31:13 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
18/10/18 05:31:13 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
18/10/18 05:31:14 INFO mapred.FileInputFormat: Total input paths to process : 5
18/10/18 05:31:14 INFO mapreduce.JobSubmitter: number of splits:5
18/10/18 05:31:15 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1539213858901_0239
18/10/18 05:31:16 INFO impl.YarnClientImpl: Submitted application application_1539213858901_0239
18/10/18 05:31:16 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1539213858901_0239/
```

Result for first question Test data Streaming Output file:

```
localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

Reduce output records=96
Spilled Records=68
Shuffled Maps =5
Failed Shuffles=0
Merged Map outputs=5
GC time elapsed (ms)=3547
CPU time spent (ms)=7760
Physical memory (bytes) snapshot=1147351040
Virtual memory (bytes) snapshot=11640664064
Total committed heap usage (bytes)=610271232
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=169
File Output Format Counters
Bytes Written=2206
18/10/18 05:31:55 INFO streaming.StreamJob: Output directory: /user/root/final_exam_output_test_Final
```


Result for first question Test Data (Inverted Index):

```
localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

[root@sandbox-hdp athletics]# hadoop fs -cat /user/root/final_exam_output_test_Final/part-00000

----Inverse Index-----

is      3      ['001', '002', '003']
weather 1      ['004']
data    2      ['001', '003']
there   1      ['002']
hadoop  2      ['001']
how      1      ['004']
does     1      ['004']
storm   2      ['001', '002']
new      1      ['003']
```

Result for first question Test Data (TFIDF):

```
localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

----TF-IDF RELEVENCE SCORE---

003      HITS      FREQ      TF      IDF      TF-IDF
new       1         1        0.2    0.698970004336  0.139794000867
is        3         1        0.2    0.221848749616  0.0443697499233
oil       1         1        0.2    0.698970004336  0.139794000867
data      2         1        0.2    0.397940008672  0.0795880017344
the       3         1        0.2    0.221848749616  0.0443697499233

002      HITS      FREQ      TF      IDF      TF-IDF
a         1         1        0.125  0.698970004336  0.087371250542
this      2         1        0.125  0.397940008672  0.049742501084
big       2         1        0.125  0.397940008672  0.049742501084
is        3         1        0.125  0.221848749616  0.027731093702
there     1         1        0.125  0.698970004336  0.087371250542
storm     2         1        0.125  0.397940008672  0.049742501084
coming    1         1        0.125  0.698970004336  0.087371250542
weekend   2         1        0.125  0.397940008672  0.049742501084

001      HITS      FREQ      TF      IDF      TF-IDF
```


Summary section: I've have created a test data same as our class lecture example. My test data given below:

001.txt file- hadoop is taking the big data world by storm big hadoop

002.txt file- there is a big storm coming this weekend

003.txt file- data is the new oil

004.txt file- how does the weather look like this weekend

005.txt file- hello world

I've checked my implementation (code) first from my test data. After that, I've worked on BBC articles (Athletics). I've set my first question answer based on lecture slide 16 example [week 11 lecture slide 16].

- My system will provide a Inverted index and TFIDF for BBC articles (Athletics
- I've worked with Hadoop MapReduce. Hadoop Mapreduce can access structured and unstructured data as well which is good.
- I think it's good. Hadoop Mapreduce support for numerous languages (java straming, python straming) that can be used for data processing and storage.

Why I chose Mapreduce, Compare with other tools: Hadoop Mapreduce support for numerous languages (java straming, python straming) for large data processing which is really good I think. However, I've also plan to work on Python Spark with same dataset.

2. (Search) Given a new query (one or more words) and a value N, retrieve the top N matching documents with a score (use TFIDF scores to retrieve the matching documents)

Code and Explanation:

Mapper File for Search and TopN matching:

```
#!/usr/bin/python
```

```
import os
```

```
import sys
```

```
# Read pairs as lines of input from STDIN
```

```
for line in sys.stdin:
```

```
    line = line.strip()
```

```
    words = line.split()
```

```
    for word in words:
```

```
        try:
```

```
            filename = os.environ['mapreduce_map_input_file'] ##Get the input filename
```

```
        except KeyError:
```

```
            filename = os.environ['map_input_file']
```

```
        word_file = word + "_" + filename[-7:-4] #remove last 4 char .txt and take only 100 from
```

```

/path/100.txt
# print '%s\t%s\t%s' % (word,1,filename)
print ("%s\t%s" %( word_file, 1))

```

Reducer File for Search and TopN matching:

```

#!/usr/bin/env python

from __future__ import division
import sys
import math

word_hits={} ##store all word as key and list all docs containing the word as value

##Inerse_index dictionary:For all word in all document, key is doc and value(another dictionary) is each word in
the doc along with their frequency, tf, idf, tf_idf
##e.g., inverse_index={"doc1":{"word":[hits,freq,tf,tf_idf]}}

inverse_index={}

query="the big data"
topN=3

for line in sys.stdin:
    word_file, count = line.strip().split("\t", 1)
    word, doc = word_file.split("_")

    try:
        count = int(count)
    except ValueError:
        continue

    if word not in word_hits:
        doc_list=[]
        doc_list.append(doc) #which doc has the word, add it to the list
        word_hits[word] = [count,doc_list]
    else:
        current_count = word_hits[word][0]
        current_doc_list=[]
        current_doc_list = word_hits[word][1]
        if doc not in current_doc_list:
            current_doc_list.append(doc)
        word_hits[word] = [current_count+1, current_doc_list]

    if doc not in inverse_index:

```



```
tf=frequency/total_word_per_doc
```

```
    #idf: log(total number of docs/hits: how many doc contain the word))  
idf=math.log10(N/hits)
```

```
    #tf_idf: Term Frequecy (TF) * idf  
tf_idf=tf*idf
```

```
    #tf_idf_list=[]  
    #tf_idf_list.append(word_key) #load the word in a doc  
    #tf_idf_list.append(tf_idf)  #load tf_idf for the word in a doc
```

```
    each_doc_dict[word_key]=tf_idf
```

```
    #print '%s\t%s\n' % (word_key,each_doc_dict[word_key])  
all_tf_idf_dict[doc] = each_doc_dict
```

```
###Print all word and their tf-idf  
#for key,value in all_tf_idf_dict.items():  
    #print '%s\t%s\n' % (key,'TF-IDF')  
#    for w,tfidf in value.items():  
        #print '%s\t%s\n' % (w,tfidf)
```

```
##Search
```

```
query_word_list=query.split()
```

```
score_dict={ }
```

```
for dd,vv in all_tf_idf_dict.items():  
    score_in_doc=0  
    score_in_doc_sum=0  
    count=0  
    for query_word in query_word_list:  
        #get_docs_list=word_hits[query_word][1]  
        #print '%s\t%s\n' % (query_word,str(get_docs_list))  
  
        #for d in get_docs_list:  
        if query_word in vv:  
            score_in_doc_sum+=float(vv[query_word])  
            count+=1  
    score_in_doc=score_in_doc_sum*(count/len(query_word_list))  
    score_dict[dd]=score_in_doc
```

```
#print "\n\nScore\n\n"
```

```

#for dc,sc in score_dict.items():
#    print '%s\t%s' %(dc,str(sc))

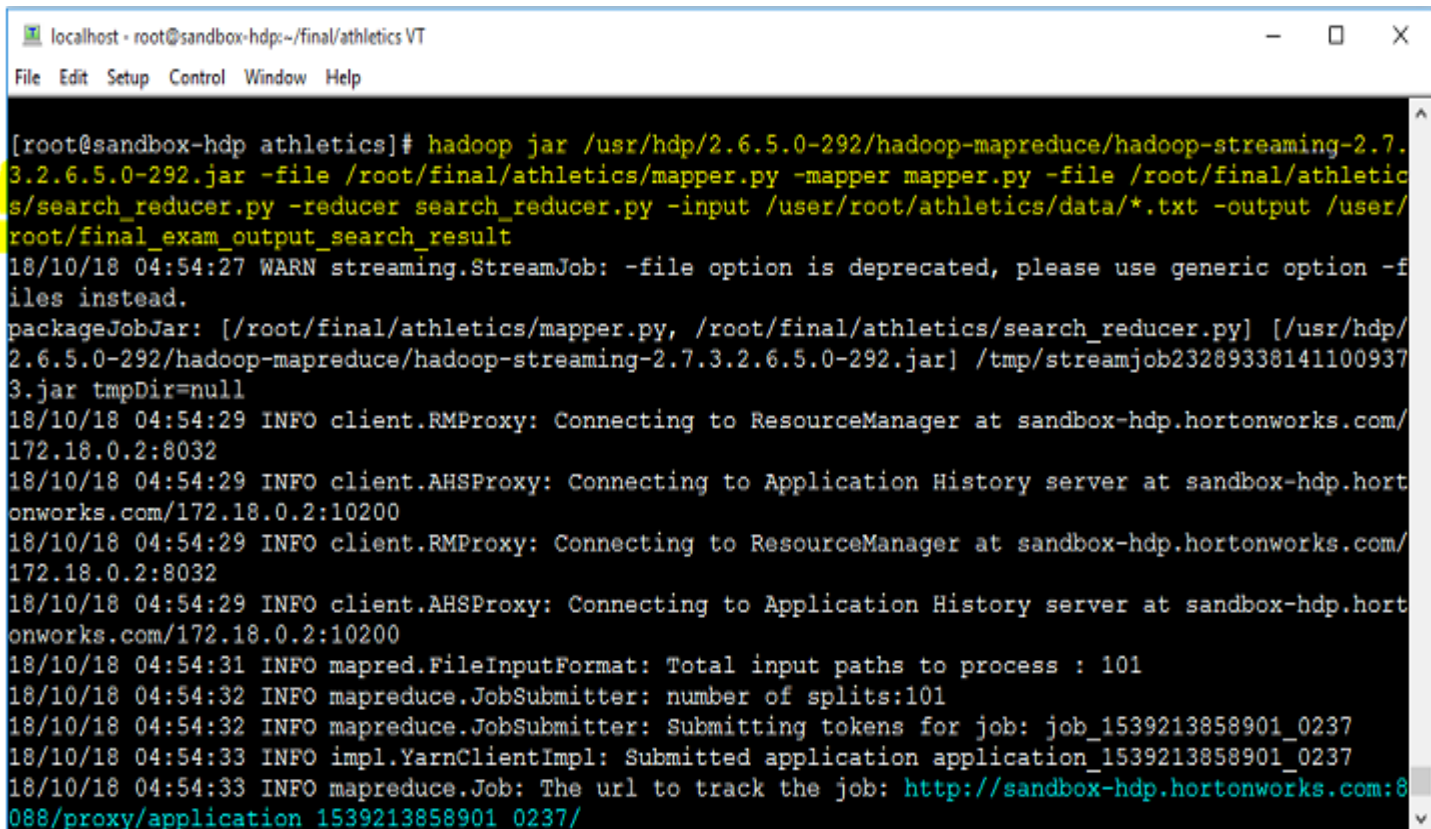
print 'Top %s Matching Document\n\n' %(topN)
print '%s\t\t%s\n' %("DOCNAME","SCORE")

c=0
for key, v2 in sorted(score_dict.iteritems(), key=lambda (k,v): (v,k), reverse=True):
    if c < topN:
        print '%s\t\t%s\n' % (key, v2)

    c+=1

```

Hadoop Streaming for Question 2:



```

localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

[root@sandbox-hdp athletics]# hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar -file /root/final/athletics/mapper.py -mapper mapper.py -file /root/final/athletics/search_reducer.py -reducer search_reducer.py -input /user/root/athletics/data/*.txt -output /user/root/final_exam_output_search_result
18/10/18 04:54:27 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/root/final/athletics/mapper.py, /root/final/athletics/search_reducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob232893381411009373.jar tmpDir=null
18/10/18 04:54:29 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
18/10/18 04:54:29 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
18/10/18 04:54:29 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
18/10/18 04:54:29 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
18/10/18 04:54:31 INFO mapred.FileInputFormat: Total input paths to process : 101
18/10/18 04:54:32 INFO mapreduce.JobSubmitter: number of splits:101
18/10/18 04:54:32 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1539213858901_0237
18/10/18 04:54:33 INFO impl.YarnClientImpl: Submitted application application_1539213858901_0237
18/10/18 04:54:33 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1539213858901_0237/

```

Streaming Output file :

```
localhost - root@sandbox-hdp:~/final/athletics VT
File Edit Setup Control Window Help

Reduce output records=11
Spilled Records=62634
Shuffled Maps =101
Failed Shuffles=0
Merged Map outputs=101
GC time elapsed (ms)=119970
CPU time spent (ms)=145650
Physical memory (bytes) snapshot=19908669440
Virtual memory (bytes) snapshot=197461716992
Total committed heap usage (bytes)=11382292480

Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

File Input Format Counters
Bytes Read=182744
File Output Format Counters
Bytes Written=115
18/10/18 05:02:20 INFO streaming.StreamJob: Output directory: /user/root/final_exam_output_search_re
sult
```

Search result from Documents and Top N:

```
[root@sandbox-hdp athletics]# hadoop fs -cat /user/root/final_exam_output_search_result/part-00000
Top 3 Matching Document

DOCNAME      SCORE
031          0.00482794880158
039          0.00409387471479
059          0.00308393205387
```

Summary section: For the second question answer, I've used TFIDF to find Relevant Search results. I've already find out TFIDF scorers in first question. So, I can use here TFIDF scores from the previous file. However, for the first question answer, I've saved my output file (Inverted index and TFIDF) in one file. So, here I calculate again TFIDF scores. Here is my searching query and TopN,

query="the big data"
topN=3

- I've Searched **"the big data"** in all text file
- I've used TFIDF score to get top N matching documents from text file