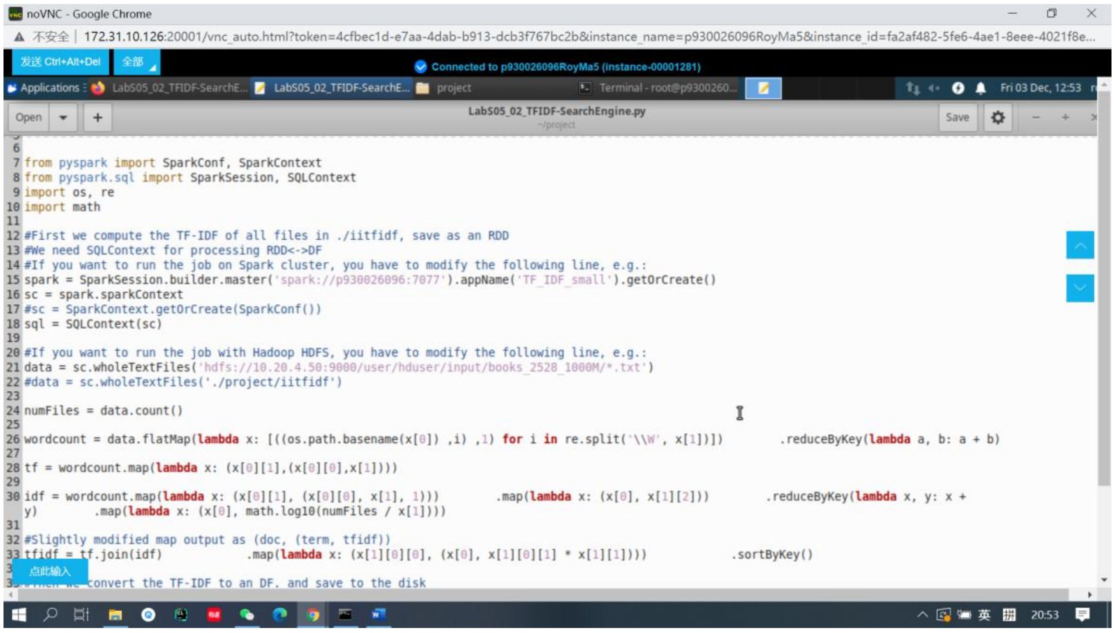
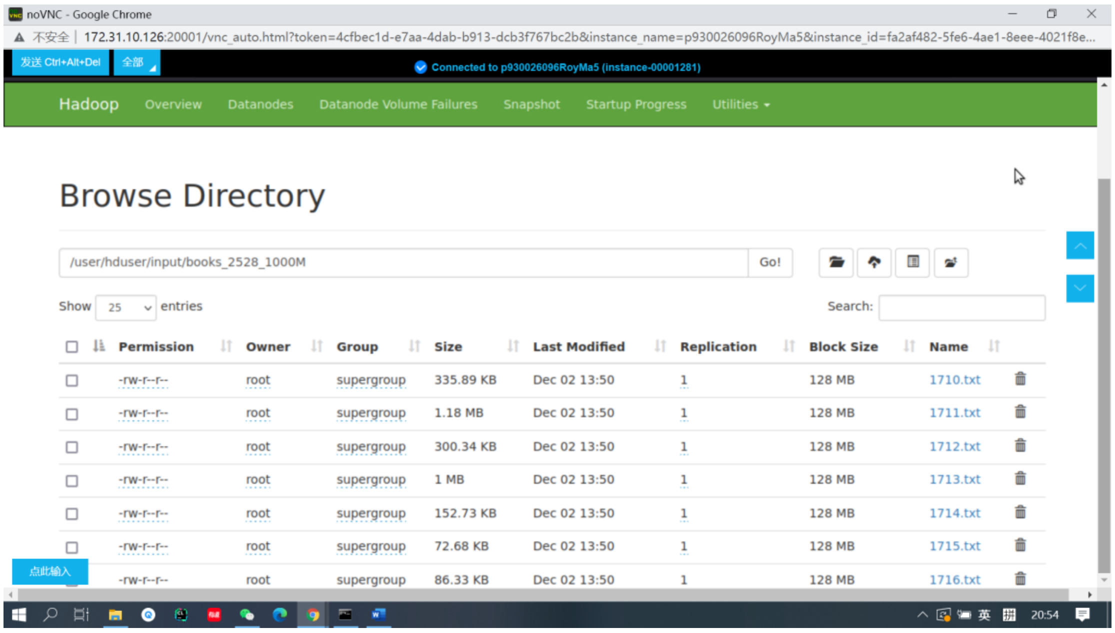
Bonus

Bonus(1)



```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, SQLContext
import os, re
import math

#First we compute the TF-IDF of all files in ./iitfidf, save as an RDD
#We need SQLContext for processing RDD<->DF
#If you want to run the job on Spark cluster, you have to modify the following line, e.g.:
spark = SparkSession.builder.master('spark://p930026096:7077').appName('TF_IDF_small').getOrCreate()
sc = spark.sparkContext
#sc = SparkContext.getOrCreate(SparkConf())
sql = SQLContext(sc)

#If you want to run the job with Hadoop HDFS, you have to modify the following line, e.g.:
data = sc.wholeTextFiles('hdfs://10.20.4.50:9000/user/hduser/input/books_2528_1000M/*.txt')
#data = sc.wholeTextFiles('./project/iitfidf')

numFiles = data.count()

wordcount = data.flatMap(lambda x: [((os.path.basename(x[0]) ,i) ,1) for i in re.split('\\W', x[1])])        .reduceByKey(lambda a, b: a + b)

tf = wordcount.map(lambda x: (x[0][1],(x[0][0],x[1])))

idf = wordcount.map(lambda x: (x[0][1], (x[0][0], x[1], 1)))        .map(lambda x: (x[0], x[1][2]))        .reduceByKey(lambda x, y: x +
y)        .map(lambda x: (x[0], math.log10(numFiles / x[1])))

#Slightly modified map output as (doc, (term, tfidf))
tfidf = tf.join(idf)        .map(lambda x: (x[1][0][0], (x[0], x[1][0][1] * x[1][1])))        .sortByKey()

convert the TF-IDF to an DF. and save to the disk
```

Run a large file (1G) from our HDFS, since there was an error when I ran the code in Jupyter notebook, so, I submit the task from the console.

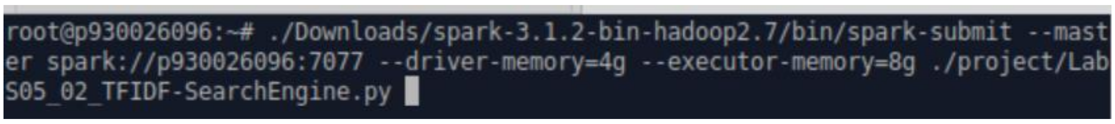


Browse Directory

/user/hduser/input/books_2528_1000M

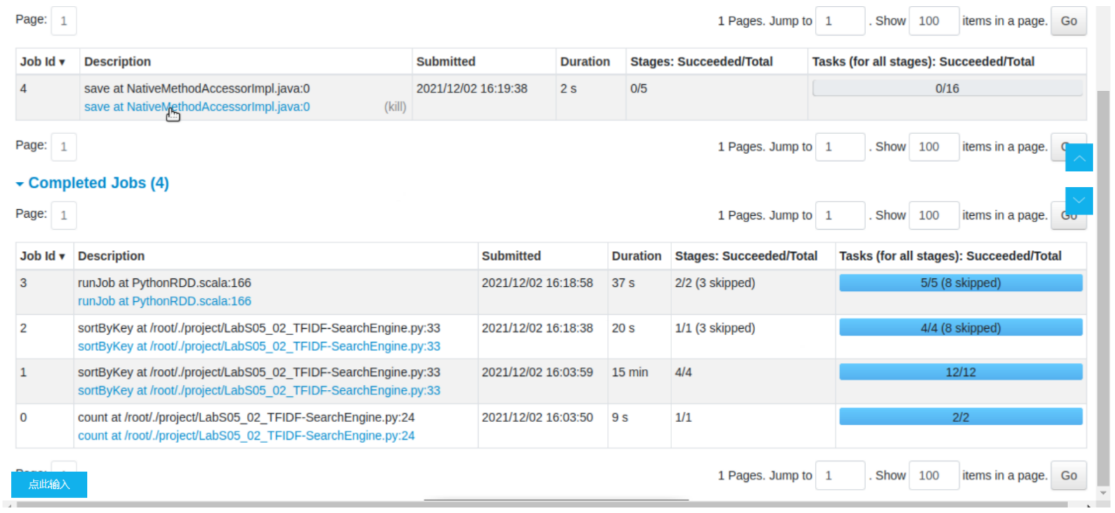
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	335.89 KB	Dec 02 13:50	1	128 MB	1710.txt
-rw-r--r--	root	supergroup	1.18 MB	Dec 02 13:50	1	128 MB	1711.txt
-rw-r--r--	root	supergroup	300.34 KB	Dec 02 13:50	1	128 MB	1712.txt
-rw-r--r--	root	supergroup	1 MB	Dec 02 13:50	1	128 MB	1713.txt
-rw-r--r--	root	supergroup	152.73 KB	Dec 02 13:50	1	128 MB	1714.txt
-rw-r--r--	root	supergroup	72.68 KB	Dec 02 13:50	1	128 MB	1715.txt
-rw-r--r--	root	supergroup	86.33 KB	Dec 02 13:50	1	128 MB	1716.txt

Here is a screen shout of my file directory in HDFS which contains more than 1G.



```
root@p930026096:~# ./Downloads/spark-3.1.2-bin-hadoop2.7/bin/spark-submit --mast
er spark://p930026096:7077 --driver-memory=4g --executor-memory=8g ./project/Lab
S05_02_TFIDF-SearchEngine.py
```

Submit the file in the console by setting the driver memory and executor memory.



Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
4	save at NativeMethodAccessorImpl.java:0 save at NativeMethodAccessorImpl.java:0 (kill)	2021/12/02 16:19:38	2 s	0/5	0/16

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

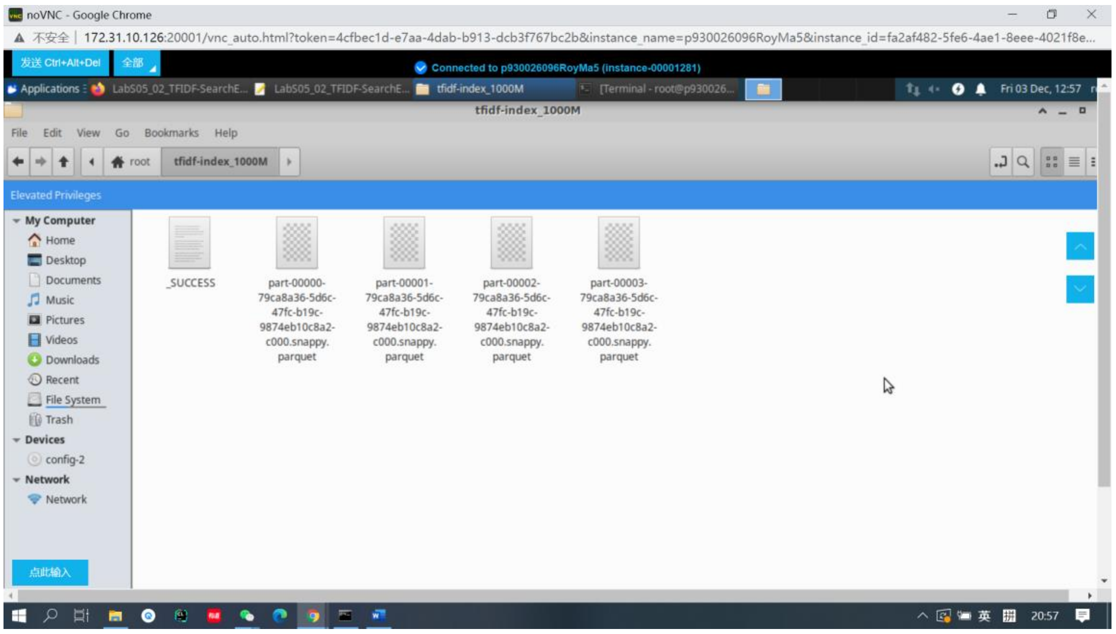
▾ Completed Jobs (4)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	runJob at PythonRDD.scala:166 runJob at PythonRDD.scala:166	2021/12/02 16:18:58	37 s	2/2 (3 skipped)	5/5 (8 skipped)
2	sortByKey at /root/./project/LabS05_02_TFIDF-SearchEngine.py:33 sortByKey at /root/./project/LabS05_02_TFIDF-SearchEngine.py:33	2021/12/02 16:18:38	20 s	1/1 (3 skipped)	4/4 (8 skipped)
1	sortByKey at /root/./project/LabS05_02_TFIDF-SearchEngine.py:33 sortByKey at /root/./project/LabS05_02_TFIDF-SearchEngine.py:33	2021/12/02 16:03:59	15 min	4/4	12/12
0	count at /root/./project/LabS05_02_TFIDF-SearchEngine.py:24 count at /root/./project/LabS05_02_TFIDF-SearchEngine.py:24	2021/12/02 16:03:50	9 s	1/1	2/2

1 Pages. Jump to 1 . Show 100 items in a page. Go

The result of task. It takes approximately 16 mins.



noVNC - Google Chrome

172.31.10.126:20001/vnc_auto.html?token=4cfbec1d-e7aa-4dab-b913-dcb3f767bc2b&instance_name=p930026096RoyMa5&instance_id=fa2af482-5fe6-4ae1-8eee-4021f8e...

Connected to p930026096RoyMa5 (instance-00001281)

tfidf-index_1000M

_SUCCESS

part-00000-79ca8a36-5d6c-47fc-b19c-9874eb10c8a2-c000.snappy.parquet

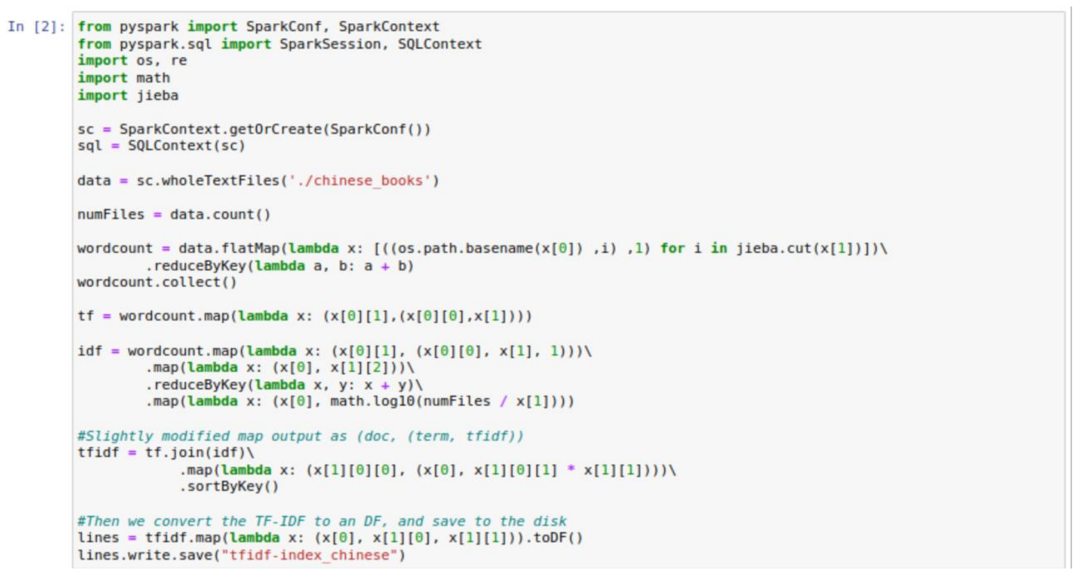
part-00001-79ca8a36-5d6c-47fc-b19c-9874eb10c8a2-c000.snappy.parquet

part-00002-79ca8a36-5d6c-47fc-b19c-9874eb10c8a2-c000.snappy.parquet

part-00003-79ca8a36-5d6c-47fc-b19c-9874eb10c8a2-c000.snappy.parquet

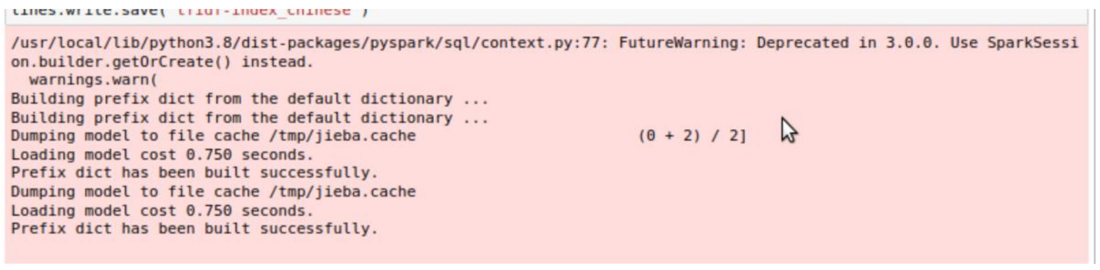
Here is the output files of the codes.

Bonus（2）



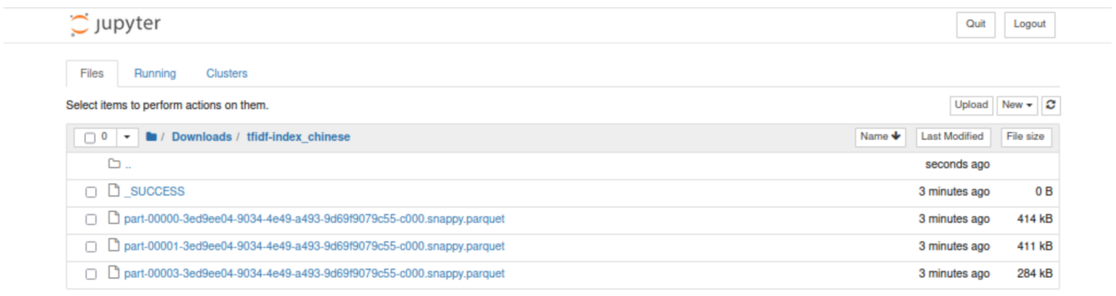
```
In [2]: from pyspark import SparkConf, SparkContext
        from pyspark.sql import SparkSession, SQLContext
        import os, re
        import math
        import jieba

        sc = SparkContext.getOrCreate(SparkConf())
        sql = SQLContext(sc)

        data = sc.wholeTextFiles('./chinese_books')

        numFiles = data.count()

        wordcount = data.flatMap(lambda x: [((os.path.basename(x[0]) ,i) ,1) for i in jieba.cut(x[1])])\
                .reduceByKey(lambda a, b: a + b)
        wordcount.collect()

        tf = wordcount.map(lambda x: (x[0][1],(x[0][0],x[1])))

        idf = wordcount.map(lambda x: (x[0][1], (x[0][0], x[1], 1)))\
                .map(lambda x: (x[0], x[1][2]))\
                .reduceByKey(lambda x, y: x + y)\
                .map(lambda x: (x[0], math.log10(numFiles / x[1])))

        #Slightly modified map output as (doc, (term, tfidf))
        tfidf = tf.join(idf)\
                    .map(lambda x: (x[1][0][0], (x[0], x[1][0][1] * x[1][1])))\
                    .sortByKey()

        #Then we convert the TF-IDF to an DF, and save to the disk
        lines = tfidf.map(lambda x: (x[0], x[1][0], x[1][1])).toDF()
        lines.write.save("tfidf-index_chinese")
```

Run the code based on some Chinese books by using the package jieba.



```
lines.write.save("tfidf-index_chinese")
/usr/local/lib/python3.8/dist-packages/pyspark/sql/context.py:77: FutureWarning: Deprecated in 3.0.0. Use SparkSessi
on.builder.getOrCreate() instead.
  warnings.warn(
Building prefix dict from the default dictionary ...
Building prefix dict from the default dictionary ...
Dumping model to file cache /tmp/jieba.cache                          (0 + 2) / 2]
Loading model cost 0.750 seconds.
Prefix dict has been built successfully.
Dumping model to file cache /tmp/jieba.cache
Loading model cost 0.750 seconds.
Prefix dict has been built successfully.
```

The program is running.



```
jupyter                                                               Quit   Logout
Files   Running   Clusters
Select items to perform actions on them.                              Upload  New ▾
0 ▾  / Downloads / tfidf-index_chinese                  Name ↓   Last Modified   File size
  ..                                                             seconds ago
_SUCCESS                                                         3 minutes ago       0 B
part-00000-3ed9ee04-9034-4e49-a493-9d69f9079c55-c000.snappy.parquet   3 minutes ago   414 kB
part-00001-3ed9ee04-9034-4e49-a493-9d69f9079c55-c000.snappy.parquet   3 minutes ago   411 kB
part-00003-3ed9ee04-9034-4e49-a493-9d69f9079c55-c000.snappy.parquet   3 minutes ago   284 kB
```

The files that generate by the codes.