Sentiment Analysis on Yelp

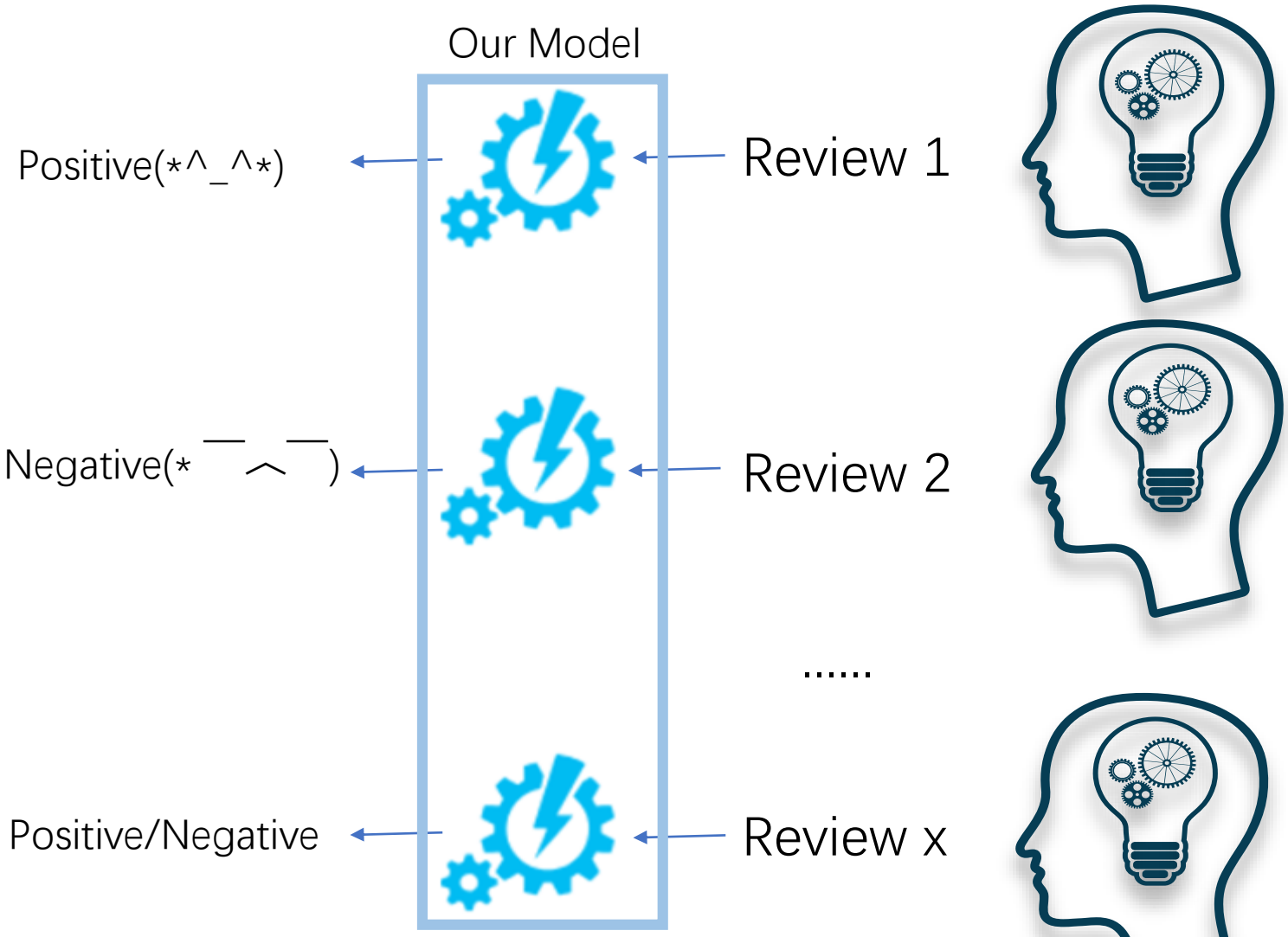
Group23

Zheng Binfan 3035588324 u3558832@connect.hku.hk
Xie Diangu 3035585803 u3558580@ connect.hku.hk
Gao Peixin 3035585114 u3558511@ connect.hku.hk

Job Distribution

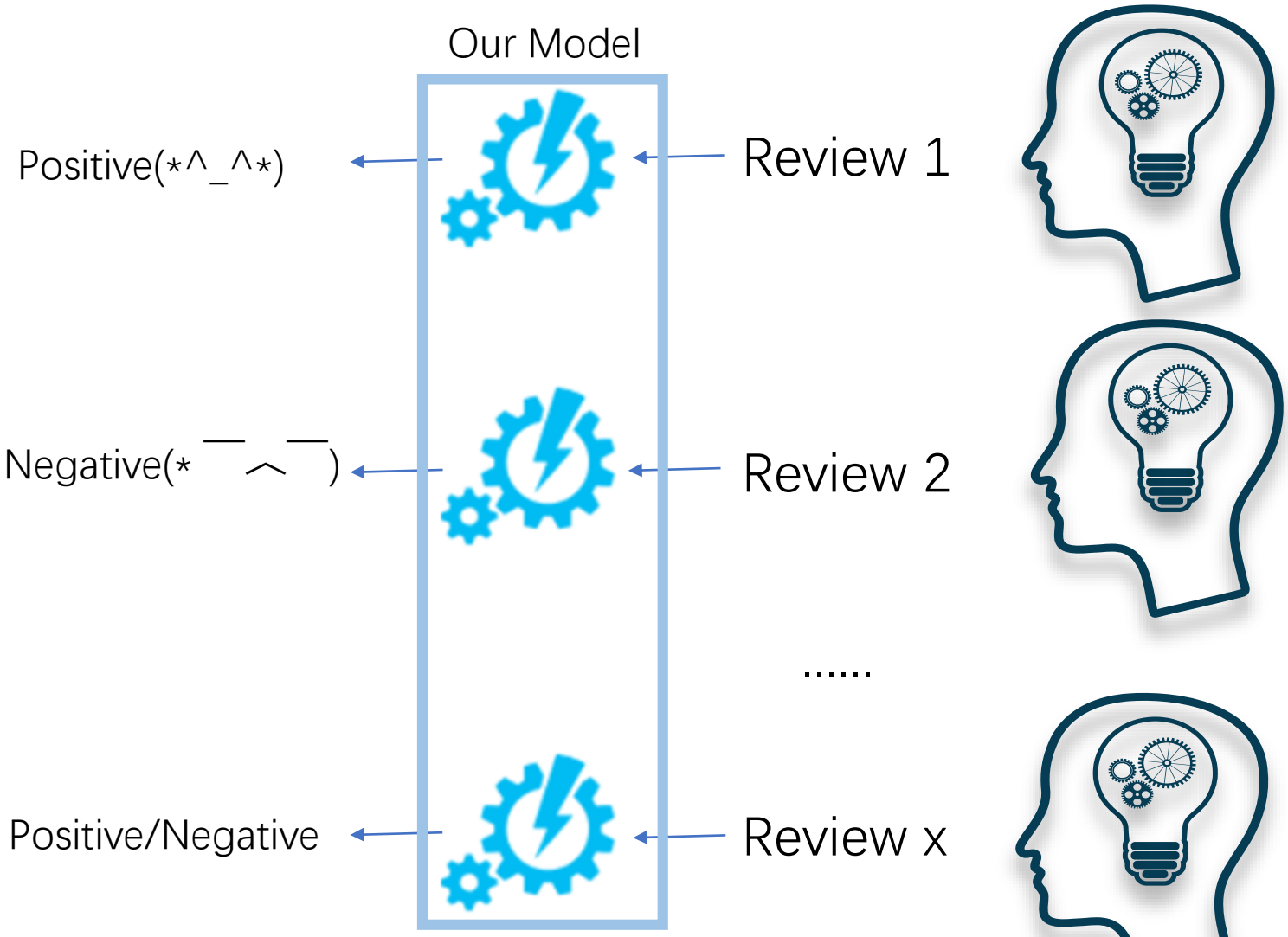
Tasks		Group leader Zheng	Member Gao	Member Xie
Data	Data collection	√	√	√
	Data pre-processing	√	√	√
	Database organization	√	√	√
Spark Clustera	Deploy the project to Spark cluster and accomplish the training process	√	√	√
	Cluster maintenance	√	√	√
	Performance Evaluation	√	√	√
Algorithm	Deep learning algorithm			
	Revise the algorithm to suit cluster computation	√	√	√
Others	Web UI Design		√	
	Prepare presentation PPT	√	√	√
	Live demo	√	√	√
	Demo video preparation	√	√	√
	Write the final report	√	√	√

Target of application



Client can use our application to know the sentiment behind a review, i.e., positive or negative.

What Clients can do



1. Train the model by their own data(it may more suitable for their own region) and make sentiment analysis on text.
2. Use our data(Yelp's review) to make sentiment analysis on text.

CONTENTS

PART

01

System Design & Data Processing

The layered view of the overall system configuration and the data processing.

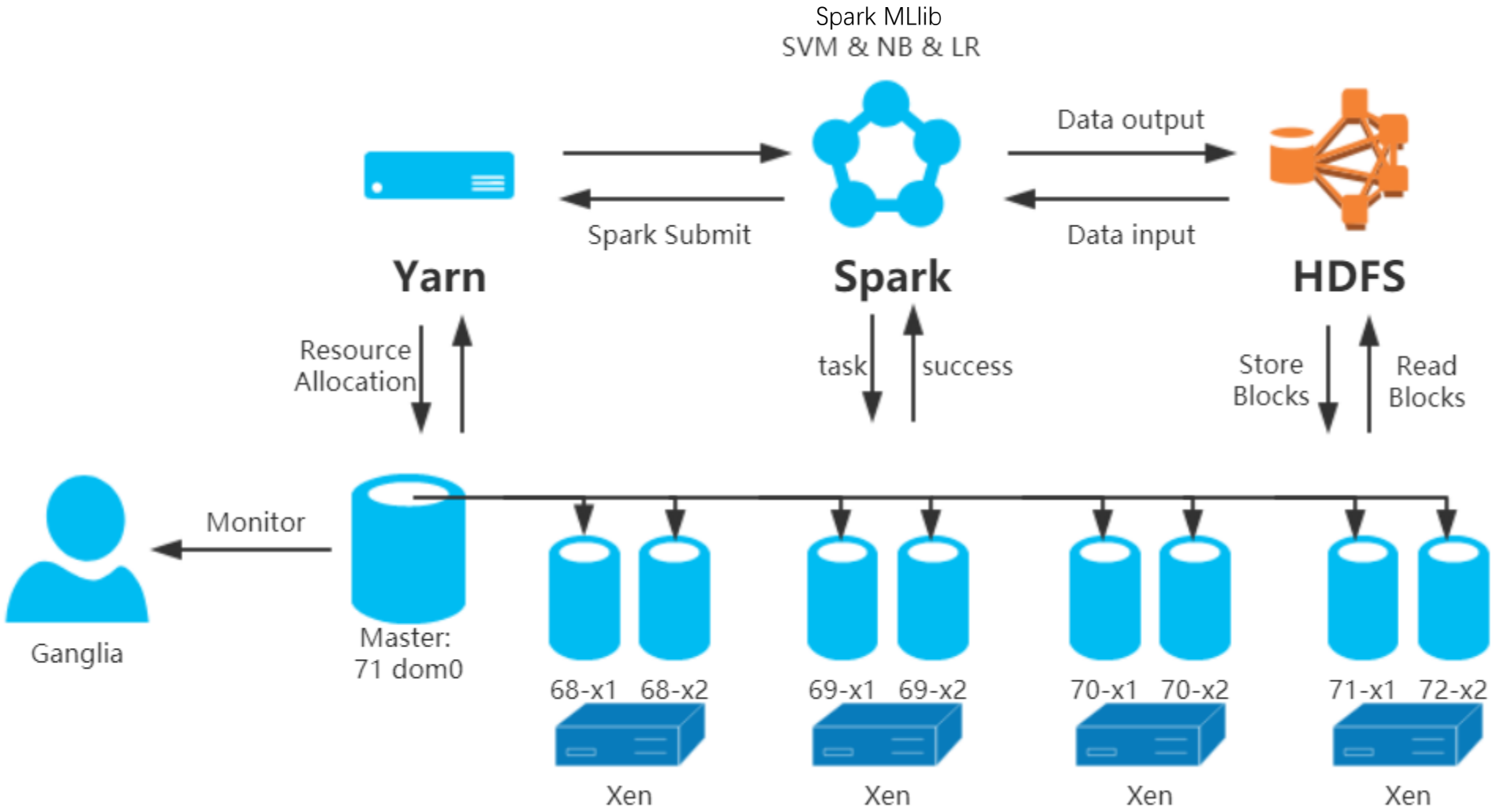
1.1. System Design

1.1.1 System Architecture[Some description from Wiki or Lecture of 7305]

Packages	Description
Xen 4.9	Providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently.
Ubuntu	Free and open-source Linux distribution based on Debian.
Hadoop-2.7.5	A collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation.
Spark-2.4.0	A unified analytics engine for large-scale data processing.
Spark-Mlib	A apache Spark's scalable machine learning library.

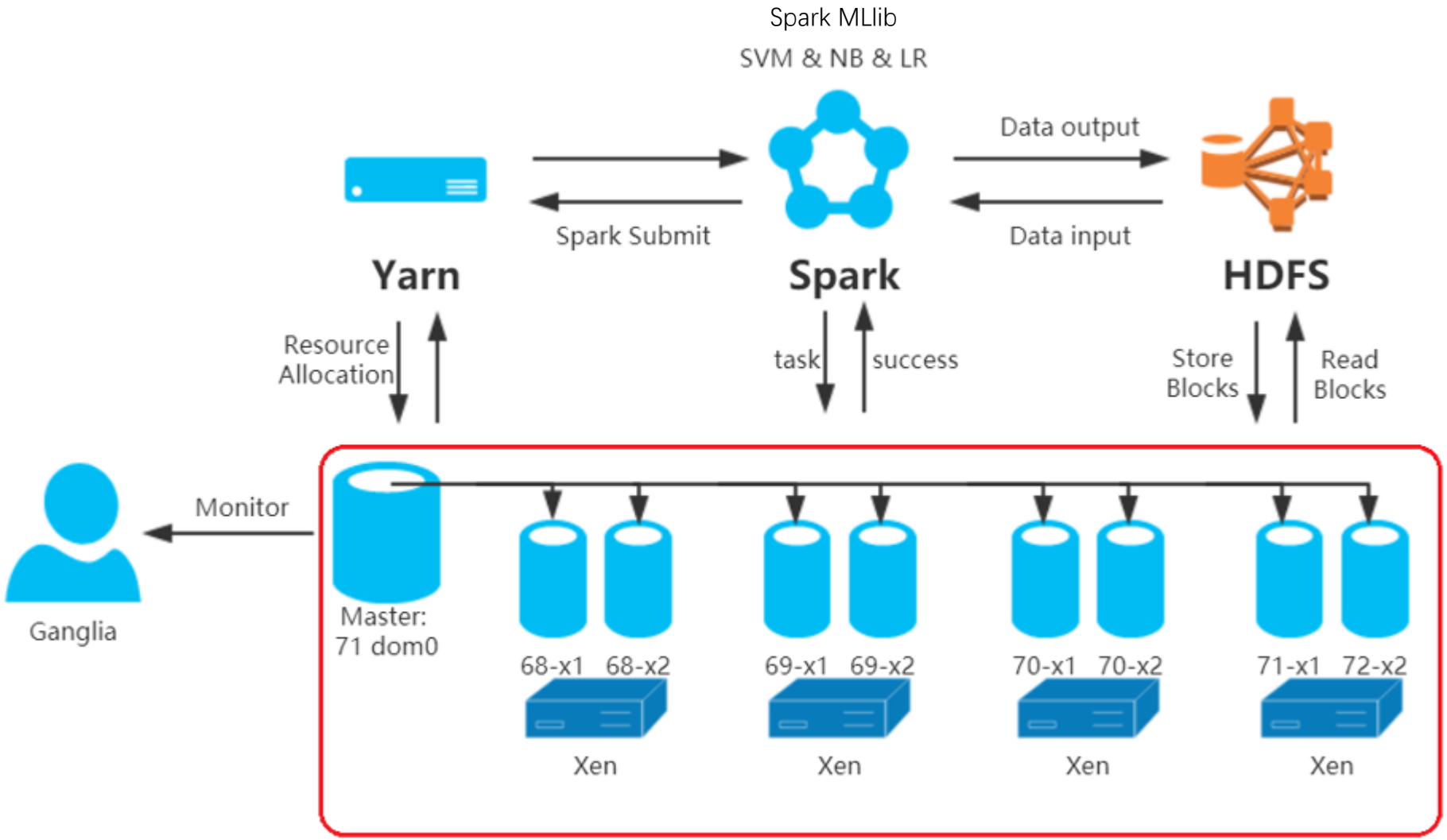
1.1. System Design

1.1.2 System Design Graph



1.1. System Design

1.1.2 System Design Graph



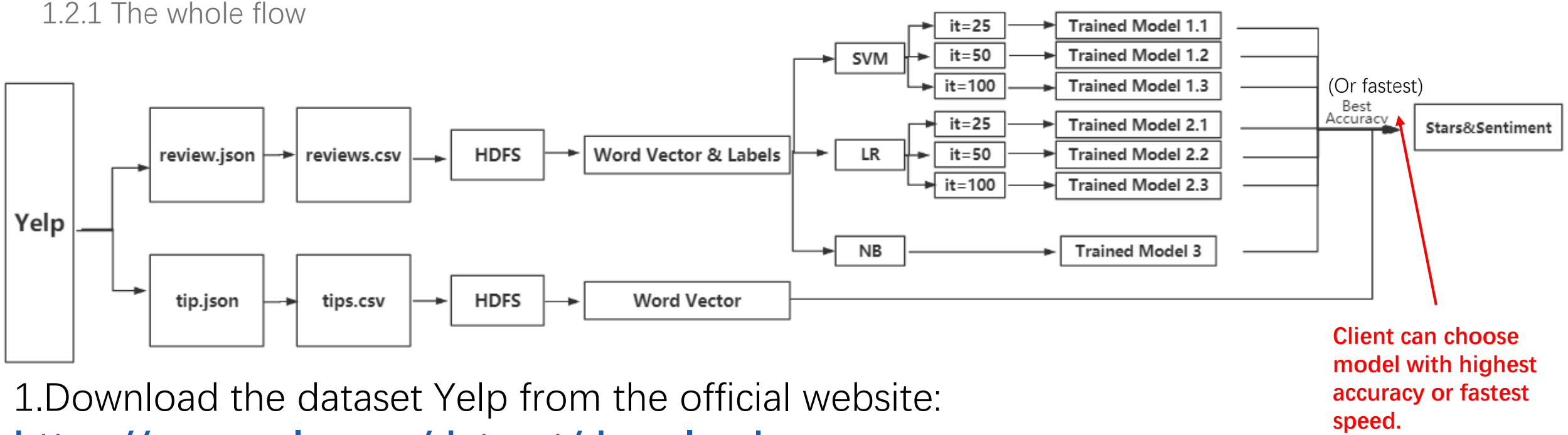
1.1. System Design

1.1.3 Cluster Configuration

Physical Machines		student68			student69			student70			student71		
CPU model		Intel® Core™ i5-8500 CPU@3.00Ghz			Intel® Core™ i5-8500 CPU@3.00Ghz			Intel® Core™ i5-8500 CPU@3.00Ghz			Intel® Core™ i5-8500 CPU@3.00Ghz		
# of cores		6			6			6			6		
RAM(GB)		16			16			16			16		
Disk sotrage(GB)		260			260			260			260		
Virtual Machines	Name	Dom0	X1	X2	Dom0	X1	X2	Dom0	X1	X2	Dom0	X1	X2
	vCPU	6	1	1	6	1	1	6	1	1	6	1	1
	Memory(GB)	7.5	4	4	7.5	4	4	7.5	4	4	7.5	4	4
	Swap(GB)	2	8	8	2	8	8	2	8	8	2	8	8
Role	Ganglia	none	slave	slave	none	slave	slave	none	slave	slave	none	slave	slave
	HDFS	none	DN	DN	none	DN	DN	none	DN	DN	none	DN	DN
	Yarn	none	NM	NM	none	NM	NM	none	NM	NM	none	NM	NM

1.2. Processing Flow

1.2.1 The whole flow



1.Download the dataset Yelp from the official website: **https://www.yelp.com/dataset/download.**

2.Extract needed attributes in tip.json and review.json and change their form to csv, and save them into HDFS.

3.Change their data form to RDD in spark and use tf-idf to change text to word vectors.

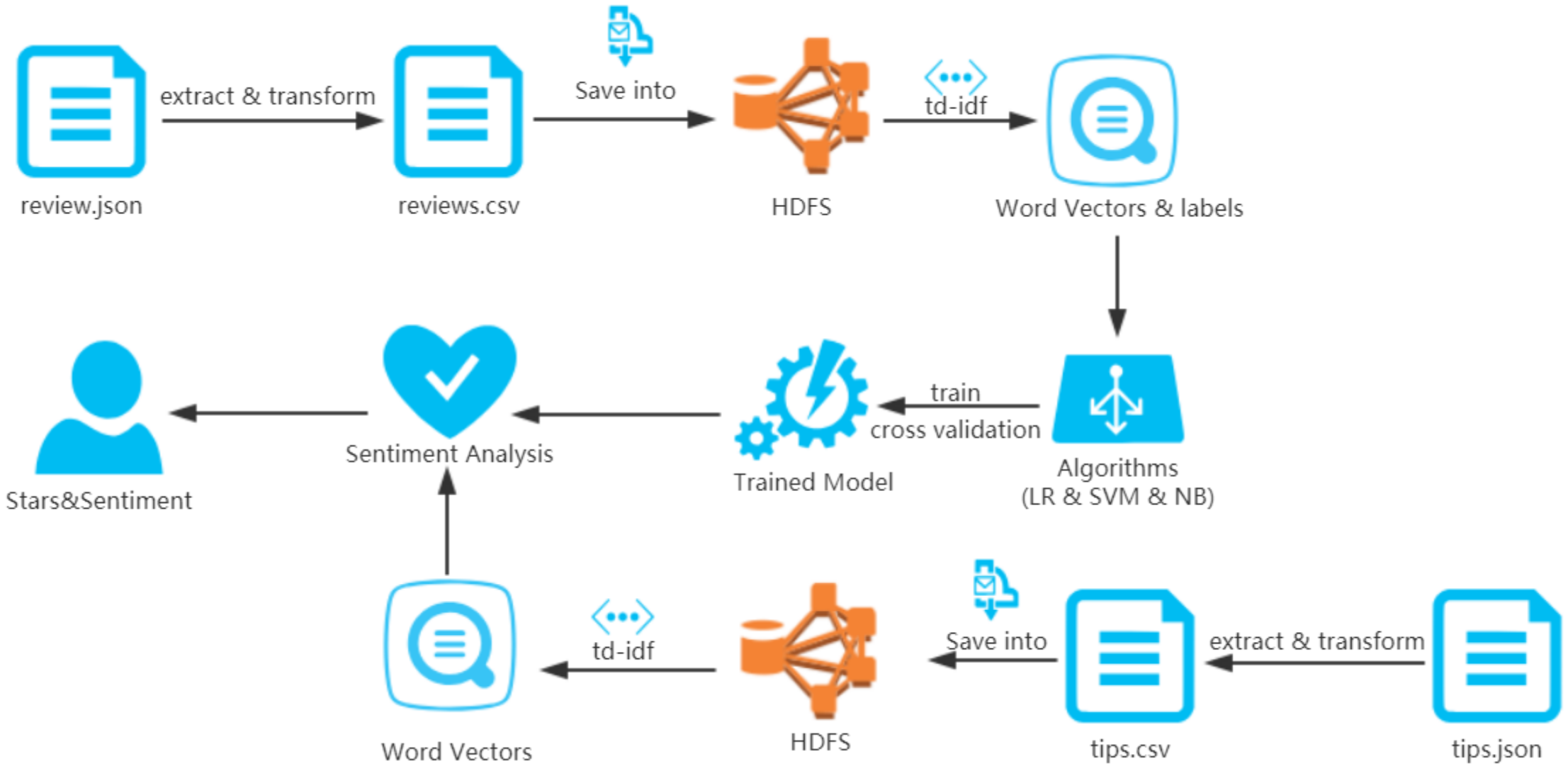
4.Use three ML algorithms we are interested to train models based on reviews, and get the best model.

5.Predict tip's texts' star and sentiment.

[PS:A more specific version is displayed in 3.3]

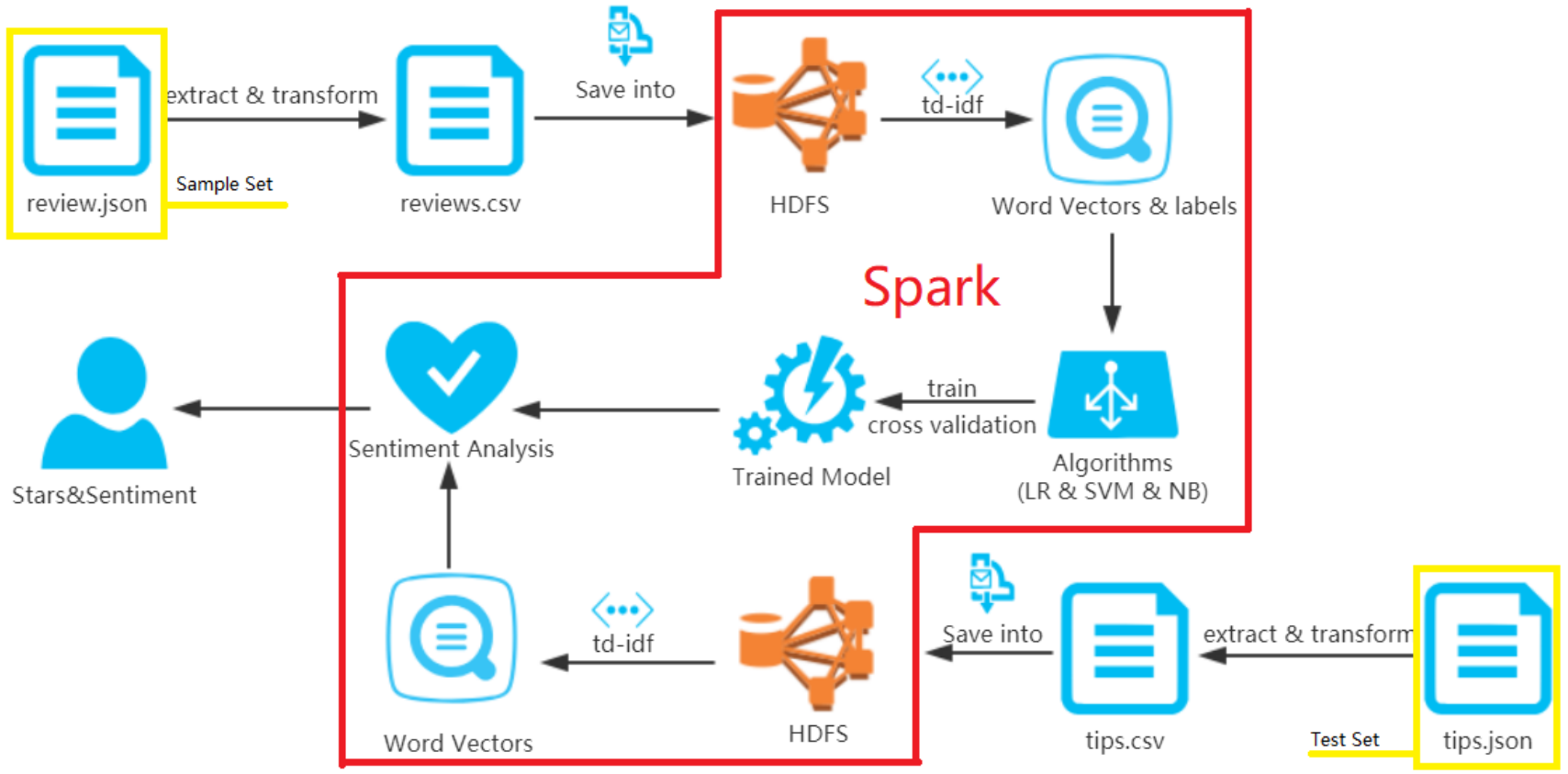
1.2. Processing Flow

1.2.2 Training and Predicting Data Processing Architecture



1.2. Processing Flow

1.2.2 Training and Predicting Data Processing Architecture



1. review.json is the sample set of our project, and client can use their own dataset to train the model. This dataset should contains text attributes and a binary attribute to state this text is positive or negative.
2. tips.json is the test set of our project, and client can use their own dataset using the model trained by their or our sample set. This dataset should have a text attribute.

PART

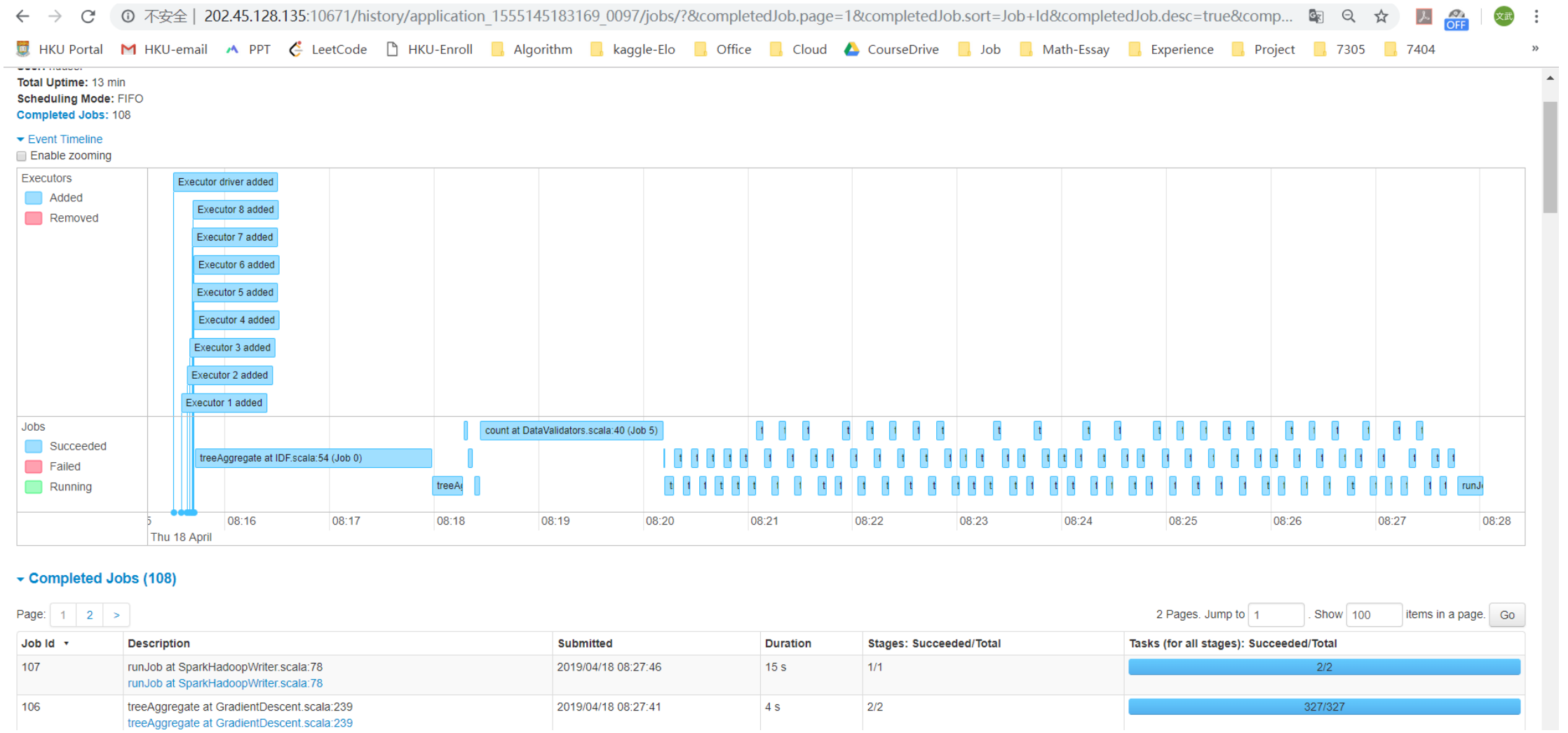
02

Installation Test & Ganglia Monitor

Some shots to prove the success of software installation and the resource usage using ganglia after running the most computation intensive task.

2.1. Installation Test

A timeline in spark of a test program of our own code.

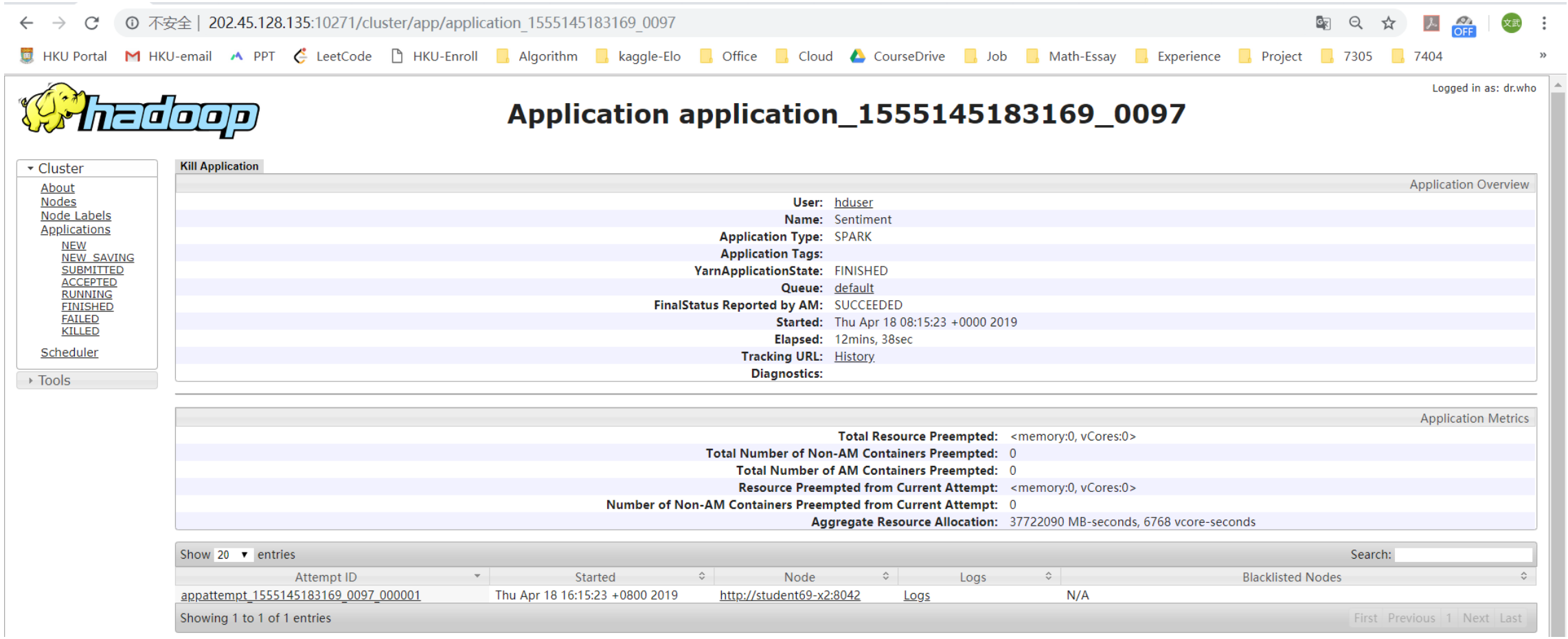


We can find some separated tasks in this graph, and it cost 13min to get the result.

2.1. Installation Test

The information in Hadoop of a test program of our own code.

application_1555145183169_0097	hduser	Sentiment	SPARK	default	Thu Apr 18 16:15:23 +0800 2019	Thu Apr 18 16:28:01 +0800 2019	FINISHED	SUCCEEDED		History	N/A



We can find some information in this Hadoop web interface.

2.2. Ganglia Monitor

The ganglia monitor after we finished the most computation intensive task.

Begin

End

At the beginning when we run the code, all screens about slaves are red.

After finished the whole process, machines become green gradually.

PART

03

Configuration Design & Key Features

1)Summary of the configurations of Hadoop and Spark; 2)Key features/functions/algorithm

3.1. Configuration Design

3.1.1 The configuration files of Hadoop

core-site.xml

```
  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://student71:9000</value>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/var/hadoop/hadoop-${user.name}</value>
        </property>
</configuration>
~
```

mapred-site.xml

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
        <property>
                <name>mapreduce.map.memory.mb</name>
                <value>200</value>
        </property>
        <property>
                <name>mapreduce.reduce.memory.mb</name>
                <value>300</value>
        </property>
</configuration>
~
```

hdfs-site.xml

```
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this

<configuration>
       <property>
              <name>dfs.replication</name>
              <value>2</value>
       </property>
       <property>
        <name>dfs.blocksize</name>
        <value>64m</value>
       </property>
</configuration>
~
~
~
```

yarn-site.xml

```
configuration>
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>student71</value>
        </property>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
        <property>
                <name>yarn.nodemanager.vmem-check-enabled</name>
                <value>false</value>
                <description>Whether virtual memory limits will be enforced for containers</description>
        </property>
        <property>
                <name>yarn.nodemanager.vmem-pmem-ratio</name>
                <value>4</value>
                <description>Ratio between virtual memory to physical memory when setting memory limits for containers</description>
        </property>
/configuration>
```

3.1. Configuration Design

3.1.2 The configuration files of spark

1. Since this algorithm need large memory, we change the driver memory and executor memory to 5G in configuration of Spark because of the memory error.
2. In order to use the resource efficiently, we change the number of executor to 10 in spark.[We do make a mistake here! Since we just assign each VM 4G memory, some memory are slopping over to disk, which slows our speed…]

```
# spark.serializer                org.apache.spark.serializer.KryoSerializer
# spark.driver.memory             5g
# spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one
spark.master spark://student71:7077
spark.serializer org.apache.spark.serializer.KryoSerializer
spark.executor.instances 8
spark.driver.memory 5g
spark.executor.cores 2
spark.executor.memory 5g
spark.eventLog.enabled true
spark.eventLog.dir hdfs://student71:9000/tmp/sparkLog
spark.history.fs.logDirectory hdfs://student71:9000/tmp/sparkLog
~
~
~
~
```

spark-defaults.xml

3.2. Key features

3.2.1 Key functions

Functions	Description
json-csv.py json2bigcsv.py	Change tip.json(review.json) file to csv file so Spark can read it directly.
Sentiment.py	1.Read csv file and change it to RDD form. 2.Transform text to word-vectors by tf-idf. 3.Train the model and make the prediction.

1. In json-csv.py and json2bigcsv.py, use library “json” and “csv” to write json and store data set as csv form.
2. In sentiment.py, use library “pyspark”’s “pyspark.mllib” & “pyspark.ml” and “jieba”.
   2.1.Use library functions SparkConf, SparkContext, textFile to read csv files and change its form to RDD.
   2.2.Use function HashingTF to get the word vectors of text.
   2.3.Use NaïveBayes, SVMWithSGD and LogisticRegression to train the model and get prediction.

3.2. Key features

3.2.1 Key functions[Showing Part1:json-csv.py & json2bigcsv.py]

Original code

```
1  # -*- coding:utf-8 -*-
2  # /bin/python
3
4  import json
5  import csv
6
7  outfile = open("review.tsv", 'wb')
8  sfile = csv.writer(outfile, delimiter ="\t", quoting=csv.QUOTE_MINIMAL)
9  sfile.writerow(['stars', 'text'])
10
11 with open('yelp_academic_dataset_review.json') as f:
12     for line in f:
13         row = json.loads(line)
14         # some special char must be encoded in 'utf-8'
15         sfile.writerow([row['stars'], (row['text']).encode('utf-8')])
16 outfile.close()
```

Our code

```
import json
import csv
outfile = open("tips.csv", 'w')
sfile = csv.writer(outfile, delimiter ="\t", quoting=csv.QUOTE_MINIMAL)
sfile.writerow(['compliment_count', 'text'])
with open('tip.json', encoding="utf8") as f:
    for line in f:
        row = json.loads(line)
        # some special char must be encoded in 'utf-8'
        sfile.writerow([row['compliment_count'], (row['text']).encode('utf-8')])
outfile.close()
~
~
```

```
import json
import csv
outfile = open("reviews.csv", 'w')
sfile = csv.writer(outfile, delimiter ="\t", quoting=csv.QUOTE_MINIMAL)
sfile.writerow(['stars', 'text'])
with open('review.json', encoding="utf8") as f:
    for line in f:
        row = json.loads(line)
        # some special char must be encoded in 'utf-8'
        sfile.writerow([row['stars'], (row['text']).encode('utf-8')])
outfile.close()
~
```

https://github.com/zhourunlai/sentiment/blob/master/classifier/sparkmllib.py

3.2. Key features

3.2.1 Key functions[Showing Part2:sentiment_test3.py]

```
import jieba
from pyspark import SparkConf, SparkContext
from pyspark.mllib.feature import HashingTF, IDF
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.classification import NaiveBayes, SVMWithSGD
from pyspark.mllib.classification import LogisticRegressionWithSGD
from pyspark.mllib.feature import Word2Vec
from pyspark.mllib.tree import RandomForest
from pyspark.mllib.tree import DecisionTree

APP_NAME = "Sentiment"

if __name__ == "__main__":
    # Configuration
    conf = SparkConf().setAppName(APP_NAME)
    conf = conf.setMaster("yarn")
    sc = SparkContext(conf=conf)

    # File import
    # originData = sc.textFile('hdfs:///tmp/output.txt')
    #originData = sc.textFile('hdfs:///tmp/review.txt')
    originData = sc.textFile(u'hdfs:///dft/reviews.csv')
#    originData = sc.textFile(u'review.csv')
    # print originData.count()
    # Processing[Delete some empty review.]
    #     originDistinctData = originData.distinct()
    # rateDocument1 = originData.map(lambda line: line.split('\t')).filter(lambda line: len(line) >= 2)
    rateDocument = originData.map(lambda line: line.split('\t')).filter(lambda line: len(line) >= 2).filter(lambda line: line[0]!='stars')
    # Remember the number
    fiveRateDocument = rateDocument.filter(lambda line: int(float(line[0])) == 5).map(lambda line: (1, line[1]))
#    print fiveRateDocument.count()
    fourRateDocument = rateDocument.filter(lambda line: int(float(line[0])) == 4).map(lambda line: (1, line[1]))
#    print fourRateDocument.count()
    threeRateDocument = rateDocument.filter(lambda line: int(float(line[0])) == 3).map(lambda line: (0, line[1]))
#    print threeRateDocument.count()
    twoRateDocument = rateDocument.filter(lambda line: int(float(line[0])) == 2).map(lambda line: (0, line[1]))
#    print twoRateDocument.count()
    oneRateDocument = rateDocument.filter(lambda line: int(float(line[0])) == 1).map(lambda line: (0, line[1]))
#    print oneRateDocument.count()
    allRateDocument = oneRateDocument.union(twoRateDocument).union(threeRateDocument).union(fourRateDocument).union(fiveRateDocument)

    rate = allRateDocument.map(lambda s: s[0])
    document = allRateDocument.map(lambda s: s[1].split(" "))
```

filter and clean data & map star 4, 5 as 1 and 1, 2, 3 as 0

What we changed

merge the data together

3.2. Key features

3.2.1 Key functions[Showing Part2:sentiment_test3.py]

```
rate = allRateDocument.map(lambda s: s[0])
document = allRateDocument.map(lambda s: s[1].split(" "))
print 'lookherer~', document.take(3)
# words = document.map(lambda w: "/".join(jieba.cut_for_search(w))).map(lambda line: line.split("/"))


hashingTF = HashingTF()
tf=hashingTF.transform(document)
tf.cache()

idfModel = IDF().fit(tf)
tfidf = idfModel.transform(tf)

zipped = rate.zip(tfidf)
data = zipped.map(lambda line: LabeledPoint(line[0], line[1]))
training, test = data.randomSplit([0.6, 0.4], seed=0)
 LR = LogisticRegression(regParam=0.01)
 LRmodel = LR.fit(training)
 LRmodel = LogisticRegressionWithSGD.train(training, iterations = 20)
 NBmodel = NaiveBayes.train(training, 1.0)
SVMmodel = SVMWithSGD.train(training, iterations=100)
 predictionAndLabel = test.map(lambda p: (SVMmodel.predict(p.features), p.label))
 RFmodel = RandomForest.trainClassifier(training, numClasses=5, categoricalFeaturesInfo={}, numTrees=5, featureSubsetSt
pth=4, maxBins=32)
 DTmodel = DecisionTree.trainClassifier(training, numClasses=2,categoricalFeaturesInfo={}, impurity="gini", maxDepth=3,
predictionAndLabel = test.map(lambda p: (SVMmodel.predict(p.features), p.label))
accuracy = 1.0 * predictionAndLabel.filter(lambda x: 1.0 if x[0] == x[1] else 0.0).count() / test.count()
print accuracy
```

using tf-idf to process text and transform text to word vector

try to use different algorithms like LR, NB and SVM to train the data

What we changed

finally we can get the accuracy

3.2. Key features

3.2.2 Key Algorithms [Description from Wiki].

Algorithm	Description
SVM	Supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis..
Logistical Regression	A widely used statistical model. In its basic form it uses a logistic function to model a binary dependent variable,
Naïve Bayes	A family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

We use three different ML algorithms and make some adjustment of their parameters to find the best trained model.
According to the accuracy of results, we can find SVM with iteration 100 times is the best model whose accuracy is 89.78%[The comparison is displayed in the 5.2.1].

3.2. Key features

3.2.3 Attributes

Attributes	Description
text	The word text of a review, and it stored both in review.json and tip.json.
stars	In review.json, we have the stars of each text, so we can know this text is positive or negative. The star's range is [1,2,3,4,5]

1. Use text and star in tip.json to train the model and get the accuracy of each model.
2. Use the best model to predict the star or sentiment of tip.json.

PART

04

Application Execution(Test & Target Finish)

1. Different execution time of our application for different sizes of input data sets.
2. Some information about the largest data test case.

4.1. Different data sets' test

4.1.1:467.19M

Input Data size: 467.19MB → Execution time: 4 min and 42 sec

Spark Jobs (?)

User: hduser
Total Uptime: 4.7 min
Scheduling Mode: FIFO
Completed Jobs: 109

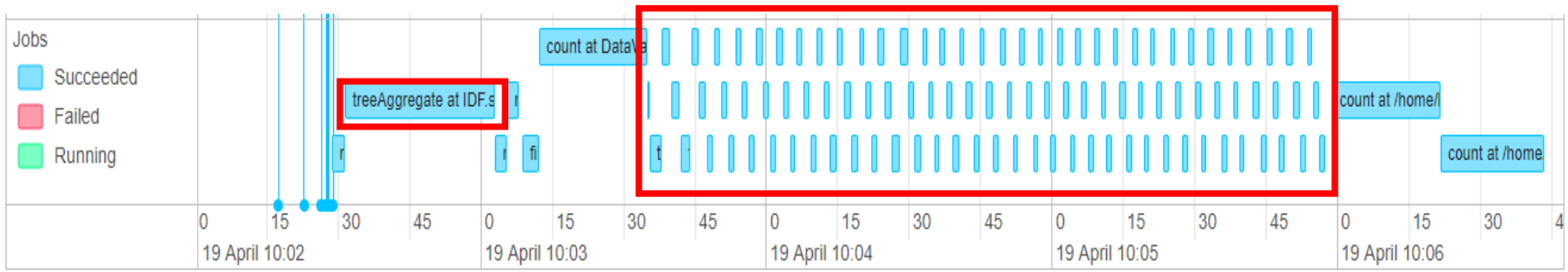
▸ Event Timeline

▾ **Completed Jobs (109)**

Page: 1 2 >

Job Id ▾	Description
108	count at /home/hdu count at /home/hdu
107	count at /home/hdu count at /home/hdu

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	1.15 GB	2019/4/19 下午6:00:32	2	64 MB	normal_reviews.csv
-rw-r--r--	hduser	supergroup	3.85 GB	2019/4/13 下午5:21:33	2	64 MB	reviews.csv
-rw-r--r--	hduser	supergroup	467.19 MB	2019/4/19 下午6:00:59	2	64 MB	small_reviews.csv



About 35 seconds

About 2 minutes and 30 seconds

4.1. Different data sets' test

4.1.2: 1.15G

Input Data size: 1.15GB → Execution time: 7 min and 18 sec

Spark Jobs (?)

User: hduser
Total Uptime: 7.3 min
Scheduling Mode: FIFO
Completed Jobs: 109

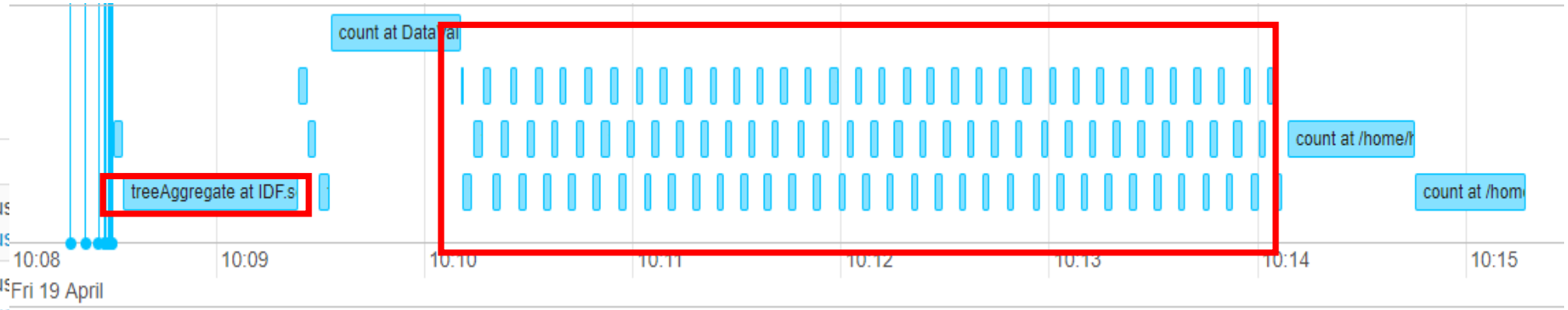
▸ Event Timeline

▾ **Completed Jobs (109)**

Page: 1 2 >

Job Id ▾	Description
108	count at /home/hdus count at /home/hdus
107	count at /home/hdus count at /home/hdus

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	1.15 GB	2019/4/19 下午6:00:32	2	64 MB	normal_reviews.csv
-rw-r--r--	hduser	supergroup	3.85 GB	2019/4/13 下午5:21:33	2	64 MB	reviews.csv
-rw-r--r--	hduser	supergroup	467.19 MB	2019/4/19 下午6:00:59	2	64 MB	small_reviews.csv



About 50 seconds

About 4 minutes

4.1. Different data sets' test

4.1.3: 3.9G[The largest data set]

Input Data size:
3.85GB(The whole dataset)

→ Execution time: 16 min

Spark Jobs (?)

User: hduser
Total Uptime: 16 min
Scheduling Mode: FIFO
Completed Jobs: 109

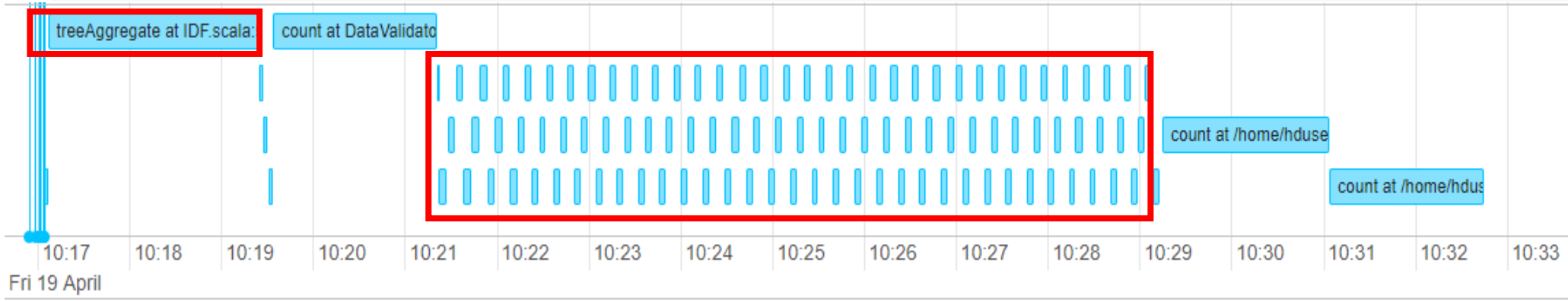
▸ Event Timeline

▾ **Completed Jobs (109)**

Page: 1 2 >

Job Id ▾	Description
108	count at /home/hd count at /home/hd
107	count at /home/hd count at /home/hd

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	supergroup	1.15 GB	2019/4/19 下午6:00:32	2	64 MB	normal_reviews.csv
-rw-r--r--	hduser	supergroup	3.85 GB	2019/4/13 下午5:21:33	2	64 MB	reviews.csv
-rw-r--r--	hduser	supergroup	467.19 MB	2019/4/19 下午6:00:59	2	64 MB	small_reviews.csv



About 2 minutes 30 seconds

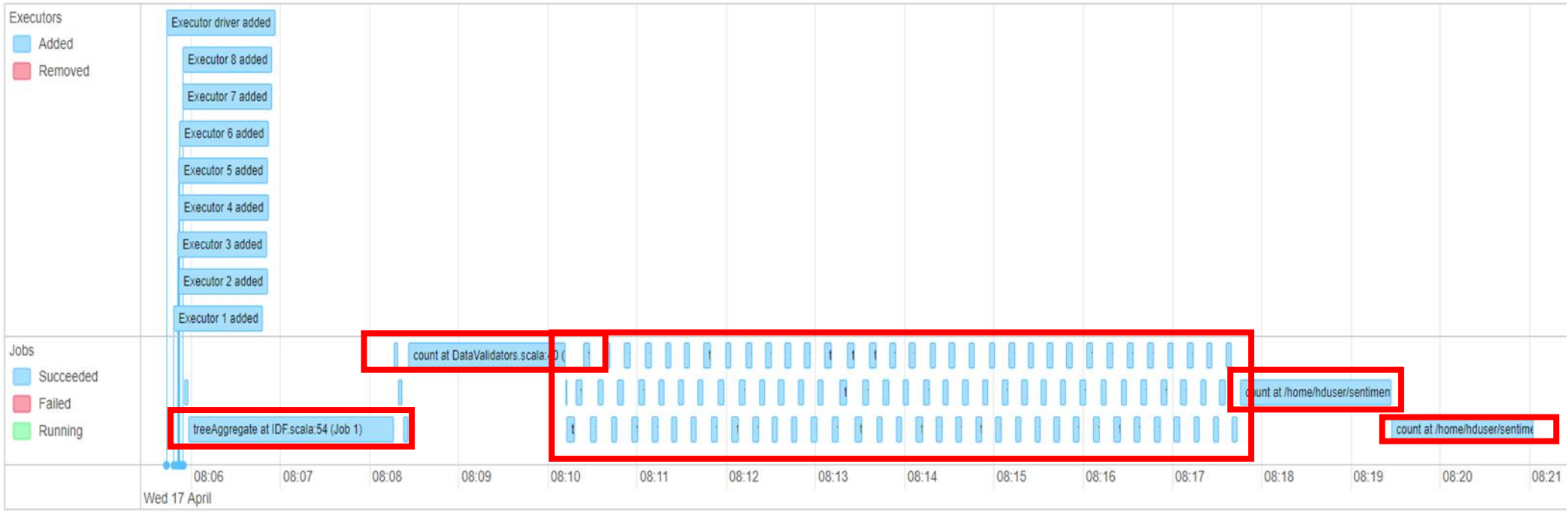
About 8 minutes

4.1. Different data sets' test

4.1.4: Summary

Data Size	TF-IDF	Train
467.19MB	35s	150s
1.15GB	50s	240s
3.85GB	150s	480s

4.2. The largest data test case



Word vector
2 min and 18 sec
Longest execution time!

Split Dataset
1 min and 48 sec

100 Iterations
5s for each iteration

Predicting
1 min and 42 sec

Computing accuracy
1 min and 36 sec

PART

05

Discussion & Analysis & Novelty

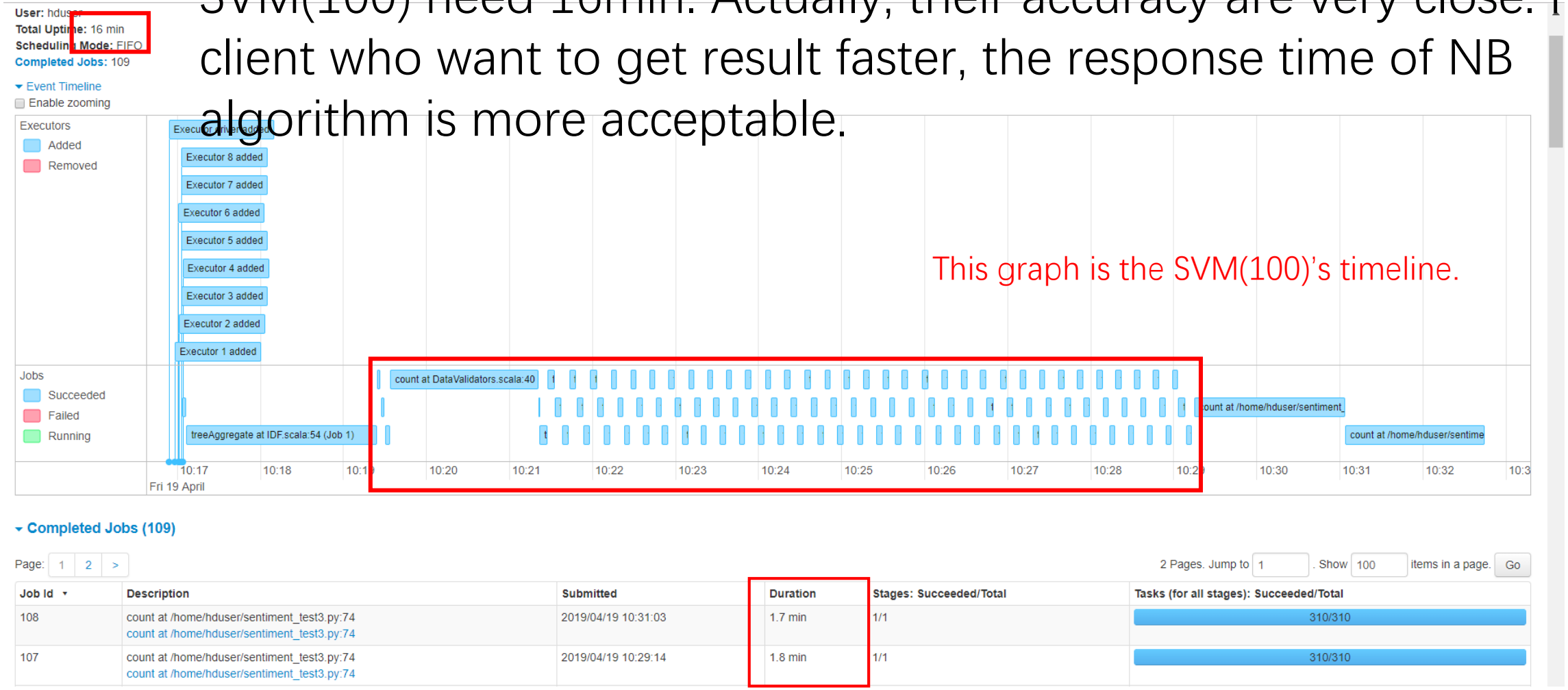
1.The key findings here.
2.Results Analysis(Comparison & Result Visualization).
3.Noverlty.

5.1. The key finding here

Q&A Part1

1.Is the response time (from query submission till result shown) acceptable?

As for the biggest data set(3.9G), NB algorithm need 8min, while SVM(100) need 16min. Actually, their accuracy are very close. For client who want to get result faster, the response time of NB algorithm is more acceptable.



5.1. The key finding here

Q&A Part2

2.Which part of your system is the bottleneck?

As for the whole processing flow, I think the train part is still a little slowly, since we can find it may cost around 8 minutes to train this model while the iteration is 100. We ignore too many iterations to avoid the overfitted problems and the accuracy we get is from the outside the sample which is divided from the original dataset(40 percent).
As for some configurations, this problem will also be discussed in next question 3&4.

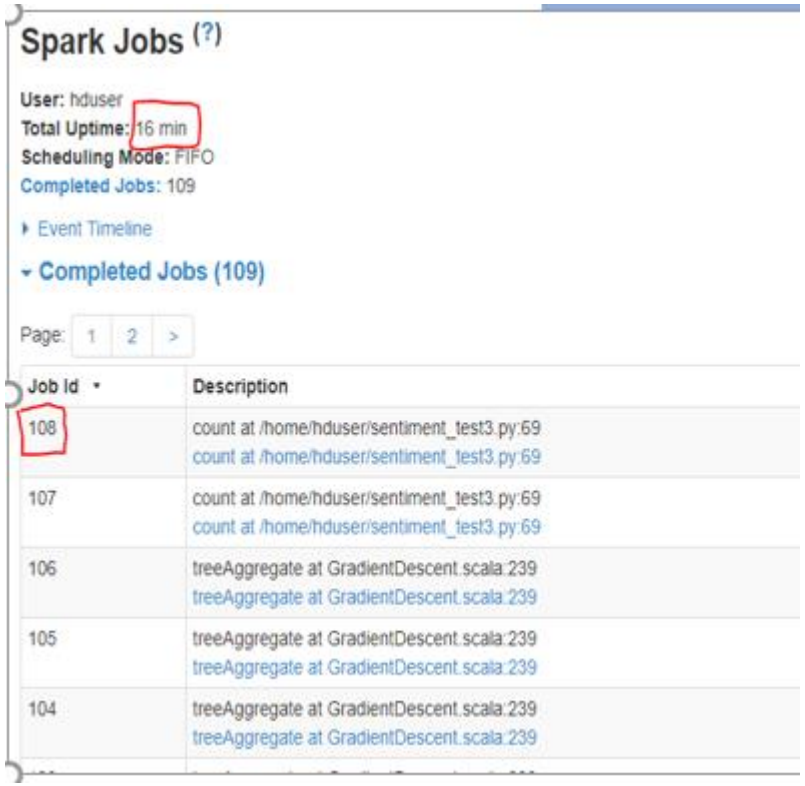
3. How did you improve the execution speed?

Increase the number of execution from 2 to 8 to 10(10 looks no speeding).[See the next slider]

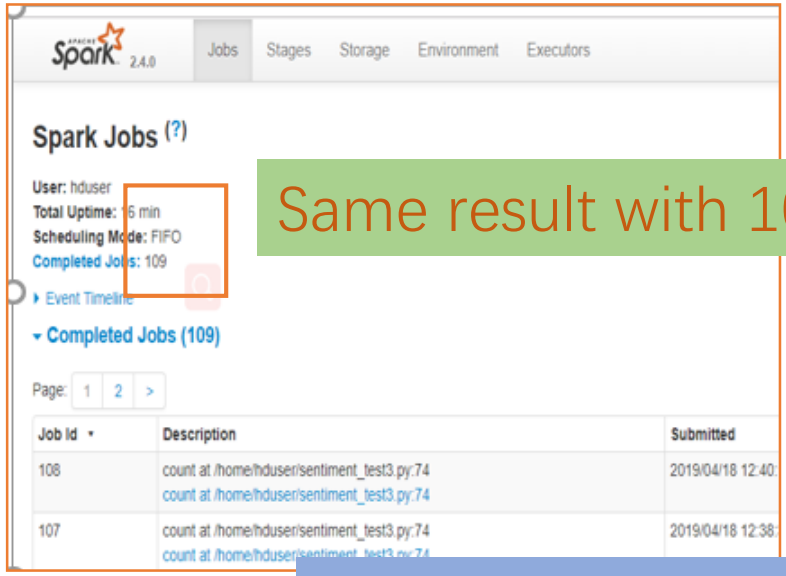
5.1. The key finding here

The attachment for answer 3.1

10 executors → Execution time: 16 min
Stage: 109 jobs

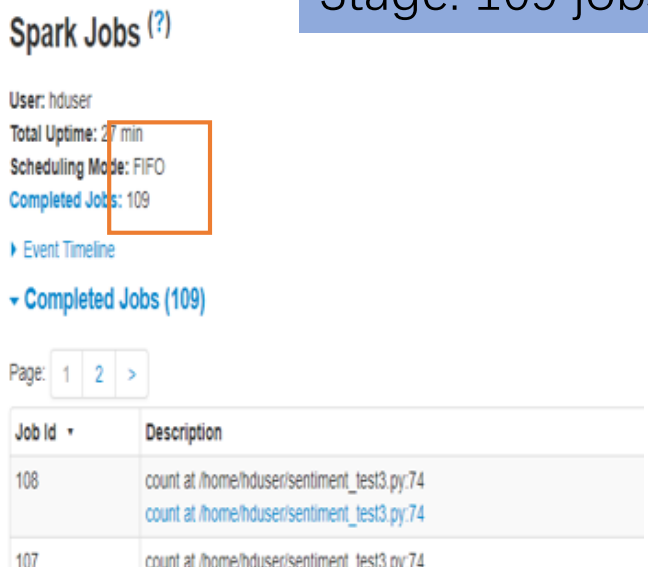


8 executors → Execution time: 16 min
Stage: 109 jobs



Same result with 10 executors!

5 executors → Execution time: 27 min
Stage: 109 jobs



5.1. The key finding here

Q&A Part3

4. Did you reconfigure the software system (Xen/Hadoop/Spark), rewrite some of the Spark code, or redesign the algorithms

4.1 We change some spark code, and add a LR algorithm.

4.2 Change the executor memory and driver memory to 5G.[As showed in 3.1 Configuration Design.]

5.2. Result Analysis

5.2.1 Comparison

Algorithm	Accuracy	Execution Time
SVM (100 iterations)	0.897803	16min
SVM (50 iterations)	0.891528	12min
SVM (20 iterations)	0.878525	9min and 48sec
Logistical Regression (100 iterations)	0.889385	17min
Logistical Regression (50 iterations)	0.883875	12min
Logistical Regression (20 iterations)	0.872767	10min
Naïve Bayes	0.859039	8min and 18sec

Best Accuracy: SVM (100 iterations)
Fastest: Naïve Bayes

```
2019-04-17 08:21:00 INFO  TaskSetManager:54 - Finished task 303.0 in stage 209.0 (TID 34266) in 3719 ms on student71-x1 (executor 6) (307/310)
2019-04-17 08:21:01 INFO  TaskSetManager:54 - Finished task 307.0 in stage 209.0 (TID 34268) in 3822 ms on student69-x2 (executor 7) (308/310)
2019-04-17 08:21:02 INFO  TaskSetManager:54 - Finished task 308.0 in stage 209.0 (TID 34269) in 3860 ms on student69-x1 (executor 8) (309/310)
2019-04-17 08:21:03 INFO  TaskSetManager:54 - Finished task 309.0 in stage 209.0 (TID 34270) in 2425 ms on student71-x1 (executor 6) (310/310)
2019-04-17 08:21:03 INFO  YarnScheduler:54 - Removed TaskSet 209.0, whose tasks have all completed, from pool
2019-04-17 08:21:03 INFO  DAGScheduler:54 - ResultStage 209 (count at /home/hduser/sentiment_test3.py:69) finished in 95.942 s
2019-04-17 08:21:03 INFO  DAGScheduler:54 - Job 108 finished: count at /home/hduser/sentiment_test3.py:69, took 95.945505 s
0.897803380559
2019-04-17 08:21:03 INFO  SparkContext:54 - Invoking stop() from shutdown hook
```

5.2. Result Analysis

5.2.2 Result Visualization

This result is get by using SVM(100)

Now it's time to input a new data file[tips.json as showed in our processing] which has text content and we can predict it to get some result![This code is also finished in spark in the sentiment.py]
【It is stored in the tips.csv. 】

Some Results

0	I hate it I hate it I hate it WOOOOORST place to rent your car!
1	Awesome Juice selections.Down to Earth peeps...my new close to the house Vape s
1	Free Comic Book Day!
1	Up there at the top of my list for great food and atmosphere in all of Arizona
1	The place is nice and close!
1	Its all good! I always get the Brisket Sandwich, Creamed Corn, Peach Cobbler, and d
0	Get drinks at the bar vs from the waitstaff, much more efficient!
1	20 years in vegas best view, also top of cromwell and mandalay bay hotel dance clul
1	Bagels, spreads and cookies all good . I would rather go to paradise but the bf loved
0	Call ahead on weekends to put your name on waiting list.
1	Very nice place . Its very artistic and beautiful
0	My dirty 30 just got dirtier
0	Gettin the groove on

This result is according with our own views~ O(∩_∩)O

Word Cloud For Positive Response

5.2. Novelty

5.2.2 Result Visualization

1.who are the target users?

Firstly, the target users may be merchants in Yelp(of course) or the owner of Yelp who can get more scores for a store from reviews even with no scores. Secondly, any other regions caring about sentiment analysis can use this model to get more information based on text.

2.Why they like to use your applications?

Our goal is to build a sentiment analysis model that predicts whether a user liked a local business or not, based on their reviews on Yelp.Restaurant owners can use this model to interpret thousands of tweets, Facebook posts, blog posts or articles posts online, so they can understand the attitude of customers to them.

PART

06

Problem Solving

Some problems we encountered and how we solve it

6.1.Schrodinger's 68-x2

Occur and disappear, it's a problem

Time: 6 April

Problem Introduction: At that day, I entered my own machine 68-x2 and wanted to continue my work, while I find I cannot enter 68-x2. Then I do some tests and restarted 68, stop and restart Hadoop under the help from our TA Miss.Wu(So thanks!), then I find some strange thing by "sudo xentop":

```
xentop - 07:08:03   Xen 4.9.2
3 domains: 1 running, 1 blocked, 1 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 16568628k total, 12339696k used, 4228932k free    CPUs: 6 @ 3000MHz
      NAME  STATE    CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS NETS NETTX(k) NETRX(k) VBDS   VBD_OO   VBD_RD   VBD_WR  VBD_RSECT  VBD_WSECT SSID
  Domain-0 -----r       2350    8.0    7921332   47.8    no limit       n/a     6    0        0        0    0        0        0        0          0          0    0
student68- --b---        126    2.6    4194304   25.3     4195328      25.3     1    1     1040     2857    2        0      987    23982      98624     258736    0
student68- ----p-          0    0.0          0    0.0           0       0.0     0    0        0        0    0        0        0        0          0          0    0
```

```
xentop - 07:09:00   Xen 4.9.2
2 domains: 1 running, 1 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
Mem: 16568628k total, 12339508k used, 4229120k free    CPUs: 6 @ 3000MHz
      NAME  STATE    CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS NETS NETTX(k) NETRX(k) VBDS   VBD_OO   VBD_RD   VBD_WR  VBD_RSECT  VBD_WSECT SSID
  Domain-0 -----r       2370   36.7    7921332   47.8    no limit       n/a     6    0        0        0    0        0        0        0          0          0    0
student68- --b---        127    0.6    4194304   25.3     4195328      25.3     1    1     1054     2883    2        0      987    24226      98624     261200    0
```

The result of "xentop" changed around every ten seconds. Sometimes 68-x2 occurred while the state is paused or blocked, while sometimes its disappeared and I can just saw 68-x1. Additionally, I cannot connect 68-x2.

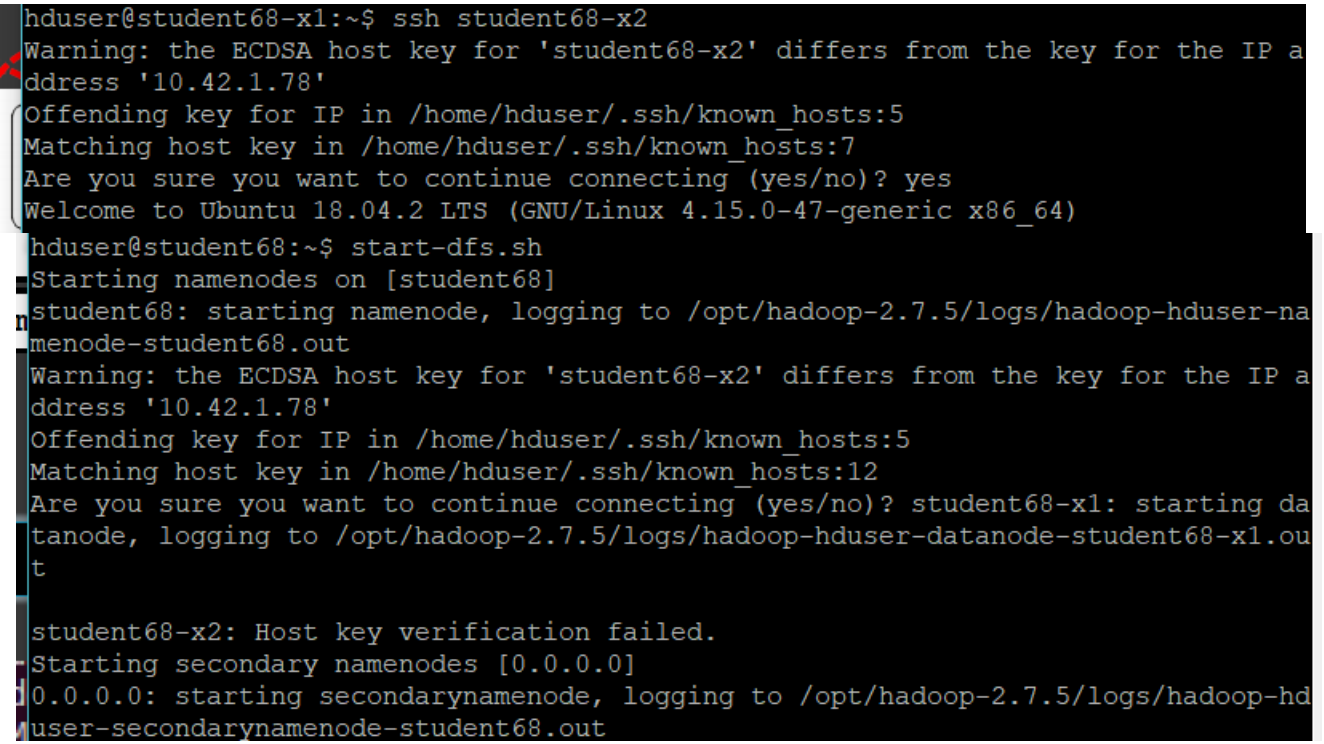
Around five hours' struggle in some advice from google, I accepted the suggestion from TA that I should destroy and rebuilt x2. Then I solved some subproblems when rebuilt VM 68-x2.

6.1.1 Subproblem1

The know_hosts conflicts in 68-x2.

Time: 10 April

Problem Introduction: Cannot enter 68-x2 without password after finish steps of 'ssh-keygen –t rsa –P ""' and copy it to VMs. So Hadoop cannot restart normally:



Process: 1. Check the IP address(it's right)

2. Check the nameserver(it's right)

3. Delete all know_hosts in all VMs and master, then repeat the 'ssh-keygen –t rsa -P's steps. Then format and restart Hadoop. The problem is solved finally.

Analysis: The know_hosts generated before make a conflict with the newly know_hosts(although there are no errors occurred when you rebuilt it). So you cannot enter the target without password until you delete all know_hosts and redo it.

6.1.2 Unhealthy Nodes

Time: 11 April

Problem Introduction: We can run our code successfully, while we find the number of active nodes in the monitor is just 7, and there is an unhealthy node: the 68-x2!!! While in other parts(HDFS and monitored index), 68-x2 is right.

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
2	0	1	1	1	2 GB	56 GB	0 B	1	56	0	7	0	0	1	0

cheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memor

Show 20 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	M			
	/default-rack	RUNNING	student70-x2:42497	student70-x2:8042	Fri Apr 12 12:39:19 +0000 2019		0	0 B	8			
	/default-rack	RUNNING	student69-x2:36403	student69-x2:8042	Fri Apr 12 12:39:19 +0000 2019		0	0 B	8			
	/default-rack	RUNNING	student71-x2:45931	student71-x2:8042	Fri Apr 12 12:39:19 +0000 2019		1	2 GB	6 GB	1	7	2.7.5
	/default-rack	RUNNING	student70-x1:39073	student70-x1:8042	Fri Apr 12 12:39:19 +0000 2019		0	0 B	8 GB	0	8	2.7.5
	/default-rack	RUNNING	student68-x1:36469	student68-x1:8042	Fri Apr 12 12:39:19 +0000 2019		0	0 B	8 GB	0	8	2.7.5
	/default-rack	RUNNING	student69-x1:36519	student69-x1:8042	Fri Apr 12 12:39:19 +0000 2019		0	0 B	8 GB	0	8	2.7.5
	/default-rack	RUNNING	student71-x1:43933	student71-x1:8042	Fri Apr 12 12:39:19 +0000 2019		0	0 B	8 GB	0	8	2.7.5

Showing 1 to 7 of 7 entries First Previous 1 Next Last

Node HTTP Address	Last health-update	Health-report
student68-x2:8042	Sat Apr 13 06:55:19 +0000 2019	1/1 log-dirs are bad: /opt/hadoop-2.7.5/logs/userlogs

Process: 1. repeat the chmod step for 68-x2(Still weak).

2. Delete all VMs Hadoop/..by 'rm -rf /var/hadoop/*' , format and restarted(still weak).

3. Redeploy Hadoop's configuration files. Then repeat the step2.(Works)

Analysis: Something wrong in the configuration of Hadoop. Although it can run successfully but 68-x2 wasn't active and unhealthy.

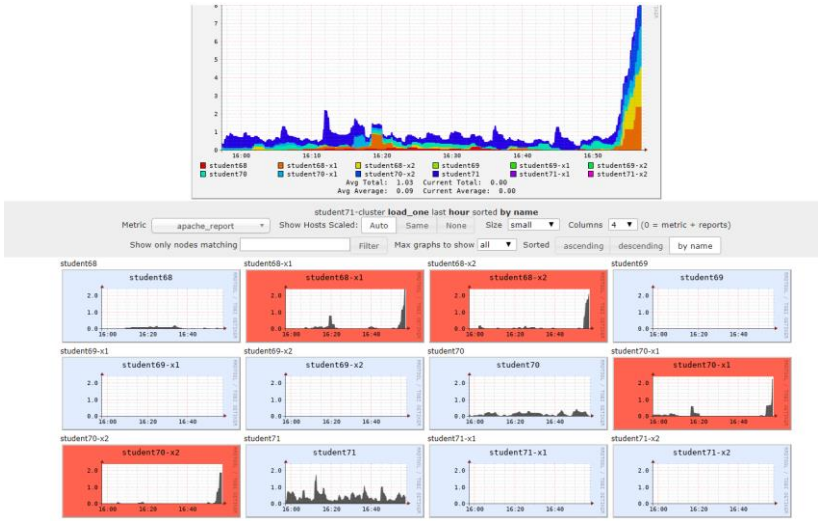
/Cores eserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
	8	0	0	0

6.2. No monitored information in 4 vms.

69-x1, 69-x2, 71-x1, 71-x2's information cannot be monitored in our ganglia.

Time: 15 April

Problem Introduction: We have finished our code and run it successfully and get our result, while we find that 69-x1, 69-x2, 71-x1, 71-x2's information cannot be monitored in our ganglia website and 'gstat -a':

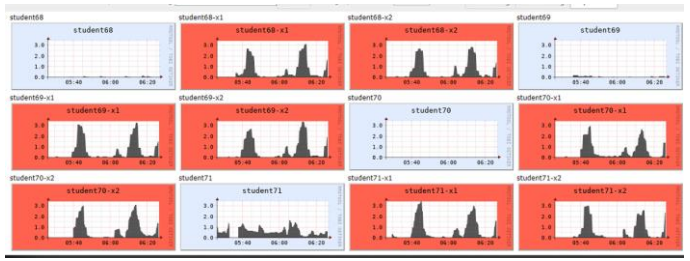


```
student70-x1
    1 (    0/  223) [  0.02,   0.06,   0.05] [   0.3,    0.0,    0.6,   99.2,    0.0]
OFF
student69
    0 (    0/    0) [  0.00,   0.00,   0.00] [   0.0,    0.0,    0.0,    0.0,    0.0]
OFF
student69-x2
    0 (    0/    0) [  0.00,   0.00,   0.00] [   0.0,    0.0,    0.0,    0.0,    0.0]
OFF
student69-x1
    0 (    0/    0) [  0.00,   0.00,   0.00] [   0.0,    0.0,    0.0,    0.0,    0.0]
OFF
student71-x2
    0 (    0/    0) [  0.00,   0.00,   0.00] [   0.0,    0.0,    0.0,    0.0,    0.0]
OFF
student71-x1
    0 (    0/    0) [  0.00,   0.00,   0.00] [   0.0,    0.0,    0.0,    0.0,    0.0]
OFF
```

Process: 1. check configuration core-site.xml and other Hadoop in all VMs.(All right so it looks no problems in Hadoop).

2. restart ganglia and restart the monitor in this nodes by code:" sudo -S su - -c 'service ganglia-monitor restart'"(Workful!)

Analysis: Some nodes' monitor are not started or lose efficacy.



6.3. SVM's invalid problems

SVM algorithm stopped in some places.

Time: 17 April

Problem Introduction: We want to use SVM to do another experience for sentiment analysis while the process showed some place are wrong here.

```
2019-04-18 13:11:50 INFO  TaskSetManager:54 - Finished task 301.0 in stage 208.0
 (TID 33950) in 7900 ms on student68-x2 (executor 1) (300/310)
2019-04-18 13:11:50 INFO  TaskSetManager:54 - Finished task 306.0 in stage 208.0
 (TID 33952) in 7762 ms on student68-x2 (executor 1) (301/310)
2019-04-18 13:11:52 INFO  TaskSetManager:54 - Finished task 303.0 in stage 208.0
 (TID 33954) in 8086 ms on student71-x1 (executor 7) (302/310)
2019-04-18 13:11:52 INFO  TaskSetManager:54 - Finished task 307.0 in stage 208.0
 (TID 33956) in 6028 ms on student71-x1 (executor 7) (303/310)
2019-04-18 13:11:53 INFO  TaskSetManager:54 - Finished task 309.0 in stage 208.0
 (TID 33960) in 2529 ms on student68-x2 (executor 1) (304/310)
2019-04-18 13:11:53 INFO  TaskSetManager:54 - Finished task 304.0 in stage 208.0
 (TID 33955) in 8082 ms on student69-x1 (executor 6) (305/310)
2019-04-18 13:11:53 INFO  TaskSetManager:54 - Finished task 308.0 in stage 208.0
 (TID 33959) in 6492 ms on student69-x1 (executor 6) (306/310)
2019-04-18 13:11:54 INFO  TaskSetManager:54 - Finished task 300.0 in stage 208.0
 (TID 33957) in 8155 ms on student69-x2 (executor 5) (307/310)
2019-04-18 13:11:55 INFO  TaskSetManager:54 - Finished task 305.0 in stage 208.0
 (TID 33958) in 7743 ms on student69-x2 (executor 5) (308/310)
2019-04-18 13:11:57 INFO  TaskSetManager:54 - Finished task 291.0 in stage 208.0
 (TID 33949) in 15219 ms on student68-x1 (executor 8) (309/310)
2019-04-18 13:11:57 INFO  TaskSetManager:54 - Finished task 292.0 in stage 208.0
 (TID 33953) in 14982 ms on student68-x1 (executor 8) (310/310)
```

Process: Change the data to start from zero.

Analysis: SVM in Spark.Mlib can just recognize status begin with 0, and it will stop here without any errors or warnings.

PART

07

Reference

Tiles and URLS of all the important references.

7 Reference

- Spark Configuration:
http://spark.apache.org/docs/latest/configuration.html

- Spark Mllib:
https://spark.apache.org/mllib/

- Yelp Website:
https://www.yelp.com/dataset/documentation/main

- Yelp Dataset:
https://www.yelp.com/dataset/download

- Open-source code:
https://github.com/zhourunlai/sentiment/blob/master/classifier/sparkmllib.py
https://github.com/haoopeng/CNN-yelp-challenge-2016-sentiment-classification/blob/master/json-csv.py