

All the codes are in https://github.com/TerryYeo/9GAG_test

Question 1 (Coding)

Assumption:

The log must follow the same structure. I treat different structure data to be a wrong data and it would be filtered by using regular expression.

Every log should have its own user_name. "Unknown" wouldn't be counted.

How to upgrade to streaming processing?

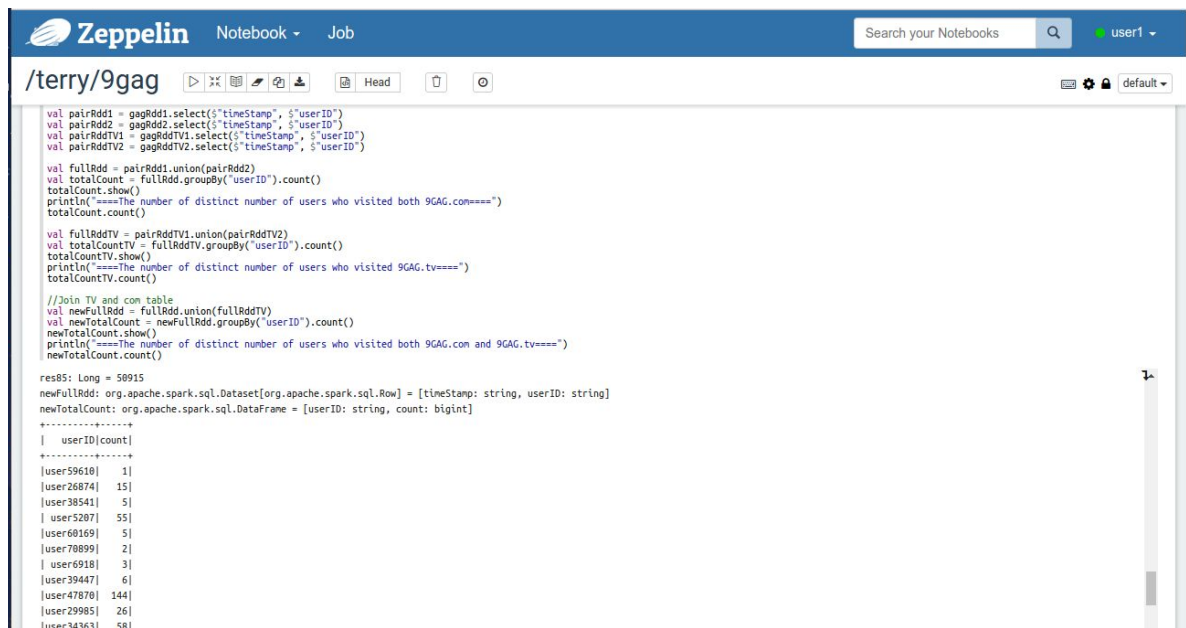
<https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html>

Program Design:

I used Apache Spark and Apache Zeppelin to write my code with functional language, Scala. All the explanations are in the comments of codes in Github repository.

Modifying to real-time streaming:

Apache Spark Structured Streaming is a good choice since I use Spark for analysis also. Thus, we don't need to change the logic or main code and add some streaming function to get data from sources, control windowing and event time.



The screenshot shows the Apache Zeppelin Notebook interface. The top bar includes the Zeppelin logo, 'Notebook' and 'Job' tabs, a search bar, and a user profile 'user1'. The main area displays a Scala script for data analysis. The script defines RDDs for different sources, unions them, and calculates counts grouped by user ID. Comments explain the steps. The output shows the total count of users who visited both 9GAG.com and 9GAG.tv, followed by a table of user counts.

```
val pairRdd1 = gagRdd1.select($"timestamp", $"userID")
val pairRdd2 = gagRdd2.select($"timestamp", $"userID")
val pairRddTV1 = gagRddTV1.select($"timestamp", $"userID")
val pairRddTV2 = gagRddTV2.select($"timestamp", $"userID")

val fullRdd = pairRdd1.union(pairRdd2)
val totalCount = fullRdd.groupBy("userID").count()
totalCount.show()
println("====The number of distinct number of users who visited both 9GAG.com====")
totalCount.count()

val fullRddTV = pairRddTV1.union(pairRddTV2)
val totalCountTV = fullRddTV.groupBy("userID").count()
totalCountTV.show()
println("====The number of distinct number of users who visited 9GAG.tv====")
totalCountTV.count()

//Join TV and com table
val newFullRdd = fullRdd.union(fullRddTV)
val newTotalCount = newFullRdd.groupBy("userID").count()
newTotalCount.show()
println("====The number of distinct number of users who visited both 9GAG.com and 9GAG.tv====")
newTotalCount.count()

res05: Long = 50915
newFullRdd: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [timestamp: string, userID: string]
newTotalCount: org.apache.spark.sql.DataFrame = [userID: string, count: bigint]
-----
+----+-----+
| userID|count|
+----+-----+
|user59610| 1|
|user26874| 15|
|user38541| 5|
| user5207| 55|
|user68169| 5|
|user78899| 2|
| user6918| 3|
|user39447| 6|
|user47870| 144|
|user29985| 26|
|user34363| 58|
```

Question 2 (Coding)

To setup Redis and HAproxy:

```
>> wget http://download.redis.io/releases/redis-3.2.4.tar.gz
>> tar xzf redis-3.2.4.tar.gz
>> cd redis-3.2.4
>> make
```

```
>> sudo apt-get install software-properties-common
>> sudo apt-get repository ppa:vbernat/haproxy-1.5
>> sudo apt-get update
>> sudo apt-get install haproxy
```

In the task of **high-availability** and **single endpoint for client communication**, I used one computer and three different ports to pretend that there are three entrances for HAproxy to manage the master and two slaves.

By using the code in my repository, the result will be following:

- A is master, B is slave
- A crashes, sentinel convert B to master
- A is recovered (still master)
- haproxy balancing between A and B, until sentinel convert A to slave
- data written to A are lost

```
>> redis-server --port 6666                #run redis master
>> redis-server --port 6667                #run redis slave 1
>> redis-server --port 6668                #run redis slave 2

>> redis-cli -p 6667 SLAVEOF 127.0.0.1 6666 #set slave 1 in same computer
>> redis-cli -p 6668 SLAVEOF 127.0.0.1 6666 #set slave 2 in same computer

>> redis-server sentinel.conf --sentinel    #run sentinel
>> haproxy -f haproxy.cfg -db              #run haproxy
```

In the task of **horizontal scalability** because it is hard for me to find three servers to test it, I could not have enough time to finish it because I found that if I used HAproxy, the maximum of size this cluster can be scaled is that the maximum of size HAproxy can handle. After the limitation of HAproxy, even increasing the number of server, it still cannot be scaled. However, redis cluster may be able to build scalability. <http://redis.io/topics/cluster-spec>

I did not have enough time to create test data and testing the redis.

Question 3 (Coding)

I use $\frac{1}{4}$ of dataset to be the test data. If you want to test by using new data, you should tokenize the sentence and test and the result is the majority of sentiment of words.

Program Design:

I used NLTK, Scikit-learn and Pandas to write my code with Python. All the explanations are in the comments of codes in Github repository.

When tokenizing the sentence, this is the error.

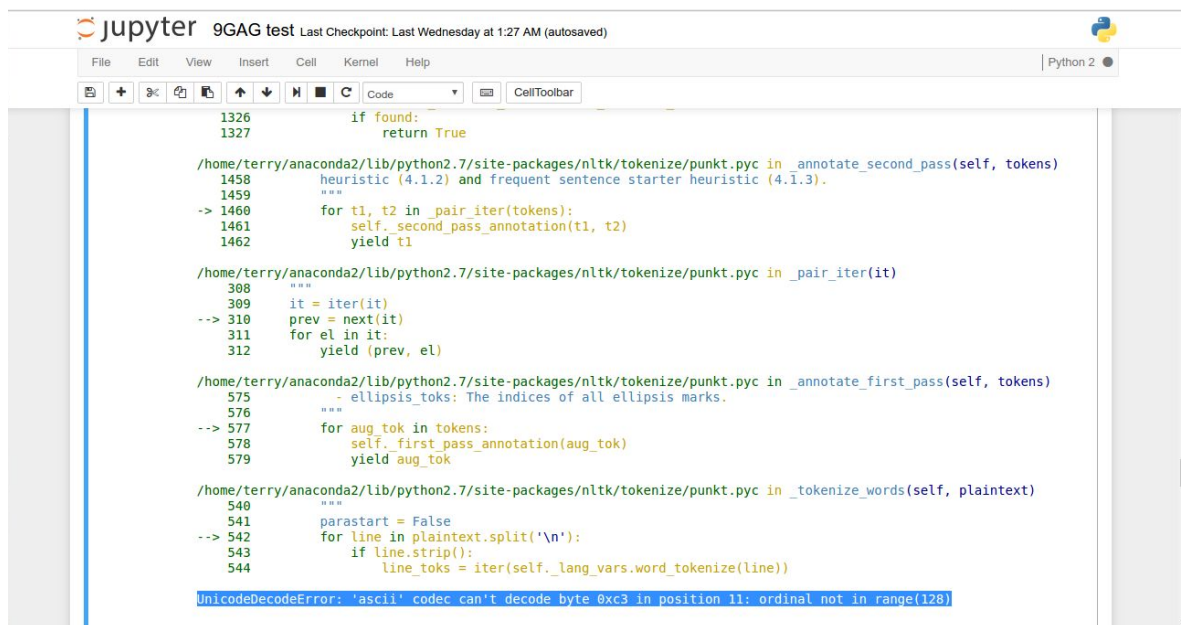
UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 11: ordinal not in range(128)

For solving this problem, need to use `.decode('utf-8', 'ignore')` for avoiding from error.

Further information:

<http://nedbatchelder.com/text/unipain.html>

This is the picture of error:



```
jupyter 9GAG test Last Checkpoint: Last Wednesday at 1:27 AM (autosaved)
File Edit View Insert Cell Kernel Help Python 2
Code CellToolbar

1326         if found:
1327             return True

/home/terry/anaconda2/lib/python2.7/site-packages/nltk/tokenize/punkt.py in _annotate_second_pass(self, tokens)
1458     heuristic (4.1.2) and frequent sentence starter heuristic (4.1.3).
1459     """
-> 1460     for t1, t2 in _pair_iter(tokens):
1461         self._second_pass_annotation(t1, t2)
1462     yield t1

/home/terry/anaconda2/lib/python2.7/site-packages/nltk/tokenize/punkt.py in _pair_iter(it)
308     """
309     it = iter(it)
-> 310     prev = next(it)
311     for el in it:
312         yield (prev, el)

/home/terry/anaconda2/lib/python2.7/site-packages/nltk/tokenize/punkt.py in _annotate_first_pass(self, tokens)
575     - ellipsis toks: The indices of all ellipsis marks.
576     """
-> 577     for aug_tok in tokens:
578         self._first_pass_annotation(aug_tok)
579     yield aug_tok

/home/terry/anaconda2/lib/python2.7/site-packages/nltk/tokenize/punkt.py in _tokenize_words(self, plaintext)
540     """
541     parastart = False
-> 542     for line in plaintext.split('\n'):
543         if line.strip():
544             line_toks = iter(self._lang_vars.word_tokenize(line))

UnicodeDecodeError: 'ascii' codec can't decode byte 0xc3 in position 11: ordinal not in range(128)
```

Most Informative Features

feature = u'love'	love : anger =	163.2 : 1.0
feature = u'happy'	love : boredo =	115.4 : 1.0
feature = u'hate'	hate : boredo =	103.7 : 1.0
feature = u'sad'	sadnes : fun =	92.3 : 1.0
feature = u'thanks'	happin : boredo =	79.2 : 1.0
feature = u'mothers'	love : hate =	78.4 : 1.0
feature = u'http'	neutra : boredo =	72.2 : 1.0
feature = u'day'	love : anger =	66.9 : 1.0

feature = u'good' happin : boredo = 64.5 : 1.0
feature = u'fun' fun : anger = 61.6 : 1.0

jupyter 9GAG test Last Checkpoint: Last Wednesday at 1:27 AM (autosaved) Python 2

```
#accuracy: 0.640243902439 with testing part of the training data
classifier2 = nltk.classify.DecisionTreeClassifier.train(trainfeats, entropy_cutoff=0, support_cutoff=0)
predict2 = classifier2.classify_many(testdata)
#print(predict2)

#BernoulliNB accuracy: 0.321646341463(approximately) with testing new data
#accuracy: 0.475609756098 with testing part of the training data
classif = SklearnClassifier(BernoulliNB()).train(trainfeats)
predict3 = classif.classify_many(testdata)
#print(predict3)

#SVM accuracy: 0.326219512195(approximately) with testing new data
#accuracy: 0.326219512195 with testing part of the training data
classif2 = SklearnClassifier(SVC(), sparse=False).train(trainfeats)
predict4 = classif2.classify_many(testdata)
#print(predict4)

print 'accuracy:', nltk.classify.util.accuracy(classifier, testfeats)
print 'accuracy:', nltk.classify.util.accuracy(classifier2, testfeats)
print 'accuracy:', nltk.classify.util.accuracy(classif, testfeats)
print 'accuracy:', nltk.classify.util.accuracy(classif2, testfeats)
```

Most Informative Features

feature = u'love'	love : anger =	214.9 : 1.0
feature = u'day'	love : anger =	186.6 : 1.0
feature = u'happy'	love : boredo =	94.2 : 1.0
feature = u'sad'	sadnes : enthus =	90.8 : 1.0
feature = u'hate'	hate : boredo =	75.1 : 1.0
feature = u'sucks'	hate : happin =	66.7 : 1.0
feature = u'thanks'	happin : boredo =	62.3 : 1.0
feature = u'mothers'	love : hate =	60.7 : 1.0
feature = u'today'	sadnes : anger =	60.4 : 1.0
feature = u'http'	neutra : boredo =	58.2 : 1.0