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

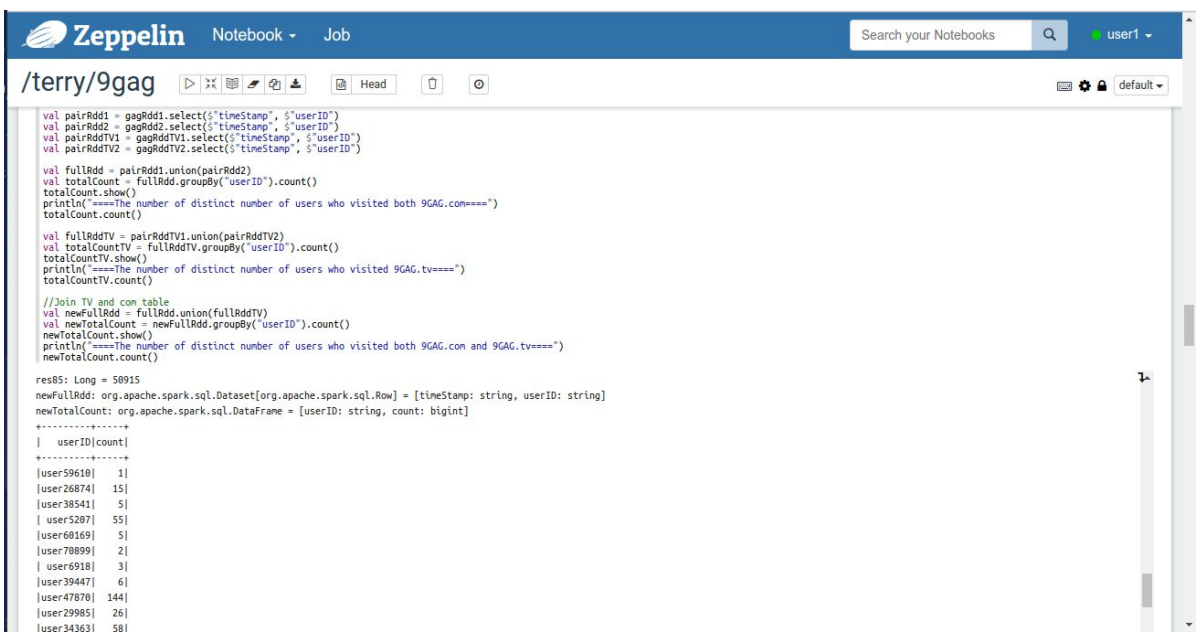
Question 1 (Coding)

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.



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.

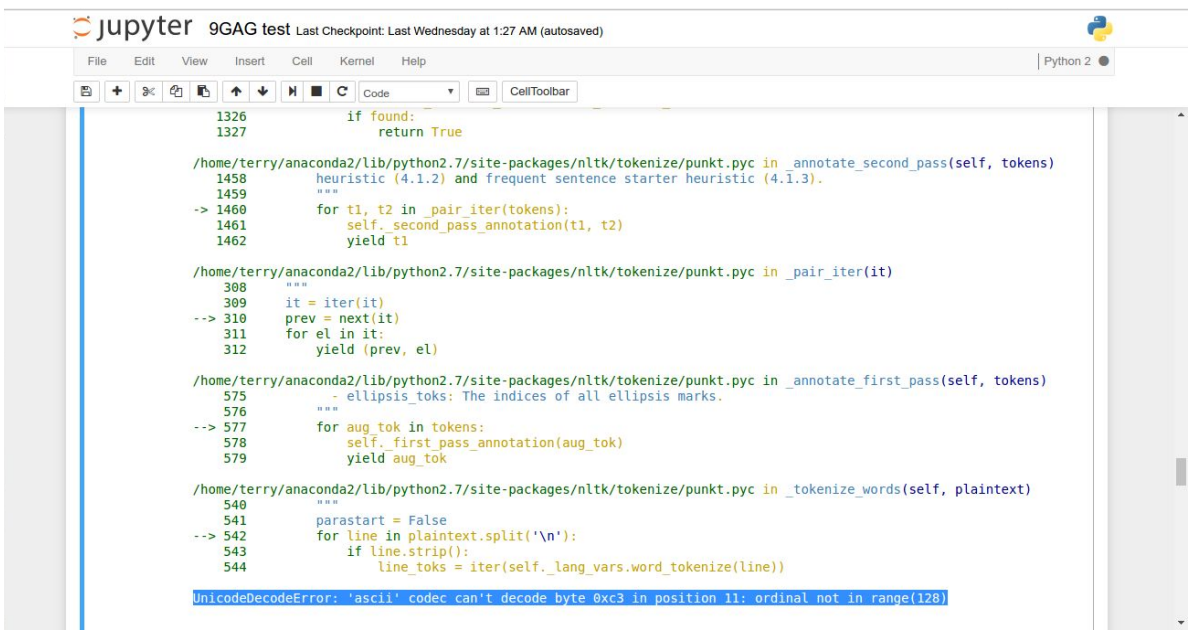
Question 3 (Coding)

I use ¼ 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:



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)

File Edit View Insert Cell Kernel Help Python 2

Code CellToolbar

```
#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
```