DATASCI W261, Machine Learning at Scale

Assignement: week #4

Lei Yang (mailto:leiyang@berkeley.edu) | Michael Kennedy (mailto:mkennedy@ischool.berkeley.edu) | Natarajan Krishnaswami (mailto:natarajan@krishnaswami.org)

Due: 2016-02-11, 8AM PST

HW4.0. Q&A

What is MrJob? How is it different to Hadoop MapReduce?

mrjob is an open-source Python 2.6+ package (https://github.com/Yelp/mrjob), originally developed by Yelp, that helps write and run Hadoop Streaming (http://hadoop.apache.org/docs/r2.7.2/hadoop-streaming/HadoopStreaming.html) jobs. mrjob fully supports Amazon's Elastic MapReduce (EMR) (https://aws.amazon.com/elasticmapreduce/) service, which allows buying time on a Hadoop cluster on an hourly basis. It also works with one's local Hadoop cluster. mrjob can also run directly via Python, for testing/debugging purpose. mrjob is a utility library that simplifies writing and running Hadoop Streaming jobs, developer doesn't need to handle many lower level tasks in Hadoop Streaming such as MapReduce job chaining, i/o management etc. User can instead focus on writing the analysis procedures for the data pipeline. In addition, mrjob provides granular control on job execution through various step definitions (https://pythonhosted.org/mrjob/job.html). Hadoop MapReduce is the execution engine that mrjob sends the MapReduce job to, and receives results from.

What are the mappint_init(), mapper_final(), combiner_final(), reducer_final() methods? When are they called?

- **mapper_init()** lets the user define any action the job needs to execute *before* the mapper starts processing any input.
- **mapper_final()**: When Hadoop Streaming stops sending data to the map task, mrjob calls mapper_final(). That function emits the local aggregation result for this task, which is a much smaller set of output lines than the mapper would have output.
- **combiner_final()**: when Hadoop Streaming stops sending data to the combine task
- **reducer_final()**: when Hadoop Streaming stops sending data to the reduce task

HW 4.1. Q&A

What is serialization in the context of MrJob or Hadoop? When it used in these frameworks?

What is serialization in the context of MrJob or Hadoop? When it used in these frameworks? Serialization is the process of converting input and output of tasks to raw bytes for Hadoop to distribute to the next task or to write as output. Serialization/deserialization is used by i/o management and data transfer, where job input/output with different format between mrjob and Hadoop are streaming through the data analysis pipeline during the job execution.

What is the default serialization mode for input and outputs for MrJob?

The default input protocol is RawValueProtocol, which reads and writes lines of raw text with no key. So by default, the first step in your job sees (None,

HW 4.2:

Recall the Microsoft logfiles data from the async lecture. The logfiles are described are located at:

- https://kdd.ics.uci.edu/databases/msweb/msweb.html (https://kdd.ics.uci.edu/databases/msweb/msweb.html)
- http://archive.ics.uci.edu/ml/machine-learning-databases/anonymous/ (http://archive.ics.uci.edu/ml/machine-learning-databases/anonymous/)

This dataset records which areas (Vroots) of www.microsoft.com each user visited in a one-week timeframe in Feburary 1998.

Here, you must preprocess the data on a single node (i.e., not on a cluster of nodes) from the format:

- C,"10001",10001 #Visitor id 10001
- V,1000,1 #Visit by Visitor 10001 to page id 1000
- V,1001,1 #Visit by Visitor 10001 to page id 1001
- V,1002,1 #Visit by Visitor 10001 to page id 1002
- C,"10002",10002 #Visitor id 10001
- V

- V

Note: #denotes comments

to the format:

- V,1000,1,C, 10001
- V,1001,1,C, 10001
- V,1002,1,C, 10001

Write the python code to accomplish this.

MrJob:

- no shuffling/sorting is needed for this task
- thus no reducer, only single mapper to preserve the order of C & V

In [2]:

```
%%writefile HW4_2.py
from mrjob.job import MRJob
from mrjob.step import MRStep

class ConvertLog(MRJob):
    # visitor name
    visitor = None
    # mapper
    def convert_mapper(self, _, line):
        # time of mapper being called
        self.increment_counter('HW4_2', 'lines', 1)
        # only emit lines start with C and V
        line = line.strip()
        if line[0] not in ['C', 'V']:
            return
        # process C and V lines
        if line[0] == 'C':
            # get the latest visitor suffix
            self.visitor = 'C,%s' %line.split(',')[2]
        else:
            # emit the desire output, no need for key
            yield None, '%s,%s' %(line, self.visitor)

    # MapReduce steps
    def steps(self):
        return [MRStep(mapper=self.convert_mapper)]

if __name__ == '__main__':
    ConvertLog.run()
```

```
Overwriting HW4_2.py
```

Execute MrJob

In [3]:

```
# run the job locally
!python HW4_2.py anonymous-msweb.data > HW4_2_results
# results sample
!head -10 HW4_2_results
```

```
no configs found; falling back on auto-configuration
no configs found; falling back on auto-configuration
creating tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_2.leiyang.20160207.164
624.031610

PLEASE NOTE: Starting in mrjob v0.5.0, protocols will be strict by default. It's recommended you r
un your job with --strict-protocols or set up mrjob.conf as described at https://pythonhosted.org/
mrjob/whats-new.html#ready-for-strict-protocols

writing to /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_2.leiyang.20160207.164624.031610/s
tep-0-mapper_part-00000
Counters from step 1:
  HW4_2:
    lines: 131666
Moving /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_2.leiyang.20160207.164624.031610/step-
0-mapper_part-00000 -> /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_2.leiyang.20160207.164
624.031610/output/part-00000
Streaming final output from /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_2.leiyang.2016020
7.164624.031610/output
removing tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_2.leiyang.20160207.164
```

```
624.031610
null    "V,1000,1,C,10001"
null    "V,1001,1,C,10001"
null    "V,1002,1,C,10001"
null    "V,1001,1,C,10002"
null    "V,1003,1,C,10002"
null    "V,1001,1,C,10003"
null    "V,1003,1,C,10003"
null    "V,1004,1,C,10003"
null    "V,1005,1,C,10004"
null    "V,1006,1,C,10005"
```

HW4.3: Find the 5 most frequently visited pages using MrJob from the output of 4.2 (i.e., transfromed log file).

MrJob Steps:

- mapper 1: emit V~count pairs
- reducer 1: count visit times n for each page V
- reducer 2: get 5 most frequently visited pages, with n sorted reversely on numeric order by the partitioner

In [254]:

```
%%writefile HW4_3.py
from mrjob.job import MRJob
from mrjob.step import MRStep

class FreqVisitPage(MRJob):
    n_freq, i = 5, 0

    def mapper_count(self, dummy, line):
        self.increment_counter('HW4_3','page_map',1)
        # get page id
        pID = line.strip().split(',')[1]
        yield pID.strip(), 1

    def reducer_count(self, page, count):
        self.increment_counter('HW4_3','page_count',1)
        yield page, sum(count)

    def mapper_sort(self, page, count):
        yield (page, count), None

    def reducer_sort(self, key, _):
        self.increment_counter('HW4_3','page_sort',1)
        if self.i < self.n_freq:
            self.i += 1
            yield key

    def steps(self):
        sort_jobconf = {  #key value pairs
            'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.mapreduce.lib.partiti
on.KeyFieldBasedComparator',
            'mapreduce.partition.keycomparator.options': '-k2,2nr',
            'mapreduce.job.maps': '2',
            'mapreduce.job.reduces': '1',
            'stream.num.map.output.key.fields': '2',
            'mapreduce.map.output.key.field.separator': ' ',
            'stream.map.output.field.separator': ' ',
        }

        count_jobconf = {
            'mapreduce.job.maps': '3',
            'mapreduce.job.reduces': '2',
        }

        return [MRStep(mapper=self.mapper_count, reducer=self.reducer_count, jobconf=count_jobcon
f)
                ,MRStep(mapper=self.mapper_sort, reducer=self.reducer_sort, jobconf=sort_jobconf)
               ]


if __name__ == '__main__':
    FreqVisitPage.run()
```

```
Overwriting HW4_3.py
```

Run the job:

In [255]:
```
### running the job locally
#!python HW4_3.py HW4_2_results > HW4_3_results

### running the job on hadoop
!python HW4_3.py HW4_2_results --hadoop-home '/usr/local/Cellar/hadoop/2.7.1/' -r hadoop > HW4_3_
results

### results
!cat HW4_3_results
```

```
no configs found; falling back on auto-configuration
no configs found; falling back on auto-configuration
creating tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_3.leiyang.20160207.205
944.473167
writing wrapper script to /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_3.leiyang.20160207.
205944.473167/setup-wrapper.sh
Using Hadoop version 2.7.1
Copying local files into hdfs:///user/leiyang/tmp/mrjob/HW4_3.leiyang.20160207.205944.473167/file
s/

PLEASE NOTE: Starting in mrjob v0.5.0, protocols will be strict by default. It's recommended you r
un your job with --strict-protocols or set up mrjob.conf as described at https://pythonhosted.org/
mrjob/whats-new.html#ready-for-strict-protocols

HADOOP: packageJobJar: [/var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/hadoop-unjar8087091006866
532002/] [] /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/streamjob2221887976295196557.jar tmpD
ir=null
Counters from step 1:
  (no counters found)
HADOOP: packageJobJar: [/var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/hadoop-unjar4072438874045
359374/] [] /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/streamjob6375479652895239471.jar tmpD
ir=null
Counters from step 2:
  (no counters found)
Streaming final output from hdfs:///user/leiyang/tmp/mrjob/HW4_3.leiyang.20160207.205944.473167/ou
tput
removing tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_3.leiyang.20160207.205
944.473167
deleting hdfs:///user/leiyang/tmp/mrjob/HW4_3.leiyang.20160207.205944.473167 from HDFS
"1008"  10836
"1034"  9383
"1004"  8463
"1018"  5330
"1017"  5108
```

HW4.4: Find the most frequent visitor of each page using MrJob and the output of 4.2 (i.e., transfromed log file). In this output please include the webpage URL, webpageID and Visitor ID.

Dev Notes:

- mapper 1: repeat HW4.2 mapper to conver log format first
- mapper 2: emit dummy key for webpage item, such that it alway comes immediately before website/visitor lines
    - for rows start with A, key is
    - partitioner: sort

In [292]:
```
%%writefile HW4_4.py
from mrjob.job import MRJob
from mrjob.step import MRStep

class FreqVisitor(MRJob):

    # member variables: visitor ID and url
    url = visitorID = None
    current_page = None
    current_max = 0

    # 1. mapper
    def convert_mapper(self, _, line):
        # time of mapper being called
        self.increment_counter('HW4_2', 'lines', 1)
        # only emit lines start with C and V
        line = line.strip()
        line = line.strip()
        if line[0] not in ['C', 'V', 'A']:
            return
        temp = line.split(',')
        # process A, C, and V lines
        if line[0] == 'C':
            # get the latest visitor ID
            self.visitorID = temp[2]
        elif line[0] == 'A':
            # emit V_pageID_*_url as key, dummy 1 as value
            yield 'V_%s_*_%s' %(temp[1], temp[4].strip('"')), 1
        else:
            # emit V_pageID_C_visitorID as key, 1 as value
            yield 'V_%s_C_%s' %(temp[1], self.visitorID), 1

    # 2. reducer to get count for each visitor on each page
    def count_reducer(self, key, value):
        temp = key.strip().split('_')
        # save webpage url for the following visisting records
        if temp[2] == '*':
            self.url = temp[3]
        else:
            yield key+'_'+self.url, sum(value)

    # 3. mapper for sorting: partition by page id, secondary sorting/ranking by count
    def rank_mapper(self, key, count):
        v, pID, c, cID, url = key.strip().split('_')
        yield 'V_%s' %pID, (count, 'C_%s - URL: %s' %(cID, url))

    # 4. reducer get max vistor of each page
    # NOTE: this implementation doesn't show all ties, just one of the record with the biggest co
unt
    def rank_reducer(self, key, value):
        # most frequent vistor of the webpage
        yield key, max(value)


    # 0. MapReduce steps
    def steps(self):
        count_jobconf = {  #key value pairs
            'mapreduce.job.maps': '1',
            'mapreduce.job.reduces': '1', # must only use 1 reducer to have the proper order
        }

        rank_jobconf = {
            #'mapreduce.job.output.key.comparator.class': 'org.apache.hadoop.mapreduce.lib.partit
ion.KeyFieldBasedComparator',
            #'mapreduce.partition.keycomparator.options': '-k1,1r', # reverse order page ID '-k2,
2nr',
            'mapreduce.job.maps': '2',
            'mapreduce.job.reduces': '1',
            #'stream.num.map.output.key.fields': '2',
            #'mapreduce.map.output.key.field.separator': '',
            #'stream.map.output.field.separator': ' ',
        }
        return [MRStep(mapper=self.convert_mapper, reducer=self.count_reducer, jobconf=count_jobc
onf)
                ,MRStep(mapper=self.rank_mapper, reducer=self.rank_reducer, jobconf=rank_jobconf)
               ]


if __name__ == '__main__':
    FreqVisitor.run()
```

```
Overwriting HW4_4.py
```

Run the job:

In [294]:

```
### running the job locally
#!python HW4_4.py anonymous-msweb.data > HW4_4_results

### running the job on hadoop
!python HW4_4.py anonymous-msweb.data --hadoop-home '/usr/local/Cellar/hadoop/2.7.1/' -r hadoop >
HW4_4_results

### results sample
!cat HW4_4_results
```

```
no configs found; falling back on auto-configuration
no configs found; falling back on auto-configuration
creating tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_4.leiyang.20160207.221
613.945000
writing wrapper script to /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_4.leiyang.20160207.
221613.945000/setup-wrapper.sh
Using Hadoop version 2.7.1
Copying local files into hdfs:///user/leiyang/tmp/mrjob/HW4_4.leiyang.20160207.221613.945000/file
s/

PLEASE NOTE: Starting in mrjob v0.5.0, protocols will be strict by default. It's recommended you r
un your job with --strict-protocols or set up mrjob.conf as described at https://pythonhosted.org/
mrjob/whats-new.html#ready-for-strict-protocols

HADOOP: packageJobJar: [/var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/hadoop-unjar8262919340003
780482/] [] /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/streamjob8665858738147888764.jar tmpD
ir=null
Counters from step 1:
  (no counters found)
HADOOP: packageJobJar: [/var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/hadoop-unjar4820819124527
597538/] [] /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/streamjob6003131137893457708.jar tmpD
ir=null
Counters from step 2:
  (no counters found)
Streaming final output from hdfs:///user/leiyang/tmp/mrjob/HW4_4.leiyang.20160207.221613.945000/ou
tput
removing tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_4.leiyang.20160207.221
613.945000
deleting hdfs:///user/leiyang/tmp/mrjob/HW4_4.leiyang.20160207.221613.945000 from HDFS
"V_1000"        [1, "C_42679 - URL: /regwiz"]
"V_1001"        [1, "C_42710 - URL: /support"]
"V_1002"        [1, "C_42592 - URL: /athome"]
"V_1003"        [1, "C_42709 - URL: /kb"]
"V_1004"        [1, "C_42707 - URL: /search"]
"V_1005"        [1, "C_42698 - URL: /norge"]
"V_1006"        [1, "C_42612 - URL: /misc"]
"V_1007"        [1, "C_42664 - URL: /ie"]
"V_1008"        [1, "C_42711 - URL: /msdownload"]
"V_1009"        [1, "C_42707 - URL: /windows"]
"V_1010"        [1, "C_42698 - URL: /vbasic"]
"V_1011"        [1, "C_42557 - URL: /officedev"]
"V_1012"        [1, "C_42650 - URL: /outlookdev"]
"V_1013"        [1, "C_42698 - URL: /vbasicsupport"]
"V_1014"        [1, "C_42698 - URL: /officefreestuff"]
"V_1015"        [1, "C_42626 - URL: /msexcel"]
"V_1016"        [1, "C_42626 - URL: /excel"]
"V_1017"        [1, "C_42692 - URL: /products"]
"V_1018"        [1, "C_42710 - URL: /isapi"]
"V_1019"        [1, "C_42311 - URL: /mspowerpoint"]
"V_1020"        [1, "C_42694 - URL: /msdn"]
"V_1021"        [1, "C_42523 - URL: /visualc"]
"V_1022"        [1, "C_42574 - URL: /truetype"]
"V_1023"        [1, "C_42665 - URL: /spain"]
"V_1024"        [1, "C_42692 - URL: /iis"]
"V_1025"        [1, "C_42657 - URL: /gallery"]
"V_1026"        [1, "C_42708 - URL: /sitebuilder"]
"V_1027"        [1, "C_42708 - URL: /intdev"]
"V_1028"        [1, "C_42063 - URL: /oledev"]
"V_1029"        [1, "C_42675 - URL: /clipgallerylive"]
"V_1030"        [1, "C_42707 - URL: /ntserver"]
"V_1031"        [1, "C_42667 - URL: /msoffice"]
"V_1032"        [1, "C_42700 - URL: /games"]
"V_1033"        [1, "C_41867 - URL: /logostore"]
"V_1034"        [1, "C_42705 - URL: /ie"]
"V_1035"        [1, "C_42710 - URL: /windowssupport"]
"V_1036"        [1, "C_42629 - URL: /organizations"]
"V_1037"        [1, "C_42650 - URL: /windows95"]
"V_1038"        [1, "C_42708 - URL: /sbnmember"]
"V_1039"        [1, "C_42656 - URL: /isp"]
"V_1040"        [1, "C_42698 - URL: /office"]
"V_1041"        [1, "C_42708 - URL: /workshop"]
"V_1042"        [1, "C_42467 - URL: /vstudio"]
"V_1043"        [1, "C_42626 - URL: /smallbiz"]
"V_1044"        [1, "C_42673 - URL: /mediadev"]
"V_1045"        [1, "C_42633 - URL: /netmeeting"]
"V_1046"        [1, "C_42702 - URL: /iesupport"]
"V_1048"        [1, "C_42541 - URL: /publisher"]
```

```
"V_1049"        [1, "C_42539 - URL: /supportnet"]
"V_1050"        [1, "C_42525 - URL: /macoffice"]
"V_1051"        [1, "C_41980 - URL: /scheduleplus"]
"V_1052"        [1, "C_42665 - URL: /word"]
"V_1053"        [1, "C_42688 - URL: /visualj"]
"V_1054"        [1, "C_42577 - URL: /exchange"]
"V_1055"        [1, "C_42424 - URL: /kids"]
"V_1056"        [1, "C_42507 - URL: /sports"]
"V_1057"        [1, "C_42667 - URL: /powerpoint"]
"V_1058"        [1, "C_42707 - URL: /referral"]
"V_1059"        [1, "C_42706 - URL: /sverige"]
"V_1060"        [1, "C_42659 - URL: /msword"]
"V_1061"        [1, "C_42576 - URL: /promo"]
"V_1062"        [1, "C_42331 - URL: /msaccess"]
"V_1063"        [1, "C_42337 - URL: /intranet"]
"V_1064"        [1, "C_42704 - URL: /activeplatform"]
"V_1065"        [1, "C_42641 - URL: /java"]
"V_1066"        [1, "C_42260 - URL: /musicproducer"]
"V_1067"        [1, "C_42456 - URL: /frontpage"]
"V_1068"        [1, "C_42025 - URL: /vbscript"]
"V_1069"        [1, "C_42613 - URL: /windowsce"]
"V_1070"        [1, "C_42692 - URL: /activex"]
"V_1071"        [1, "C_42621 - URL: /automap"]
"V_1072"        [1, "C_42446 - URL: /vinterdev"]
"V_1073"        [1, "C_42469 - URL: /taiwan"]
"V_1074"        [1, "C_42597 - URL: /ntworkstation"]
"V_1075"        [1, "C_42622 - URL: /jobs"]
"V_1076"        [1, "C_42701 - URL: /ntwkssupport"]
"V_1077"        [1, "C_42626 - URL: /msofficesupport"]
"V_1078"        [1, "C_42643 - URL: /ntserversupport"]
"V_1079"        [1, "C_42572 - URL: /australia"]
"V_1080"        [1, "C_42432 - URL: /brasil"]
"V_1081"        [1, "C_42694 - URL: /accessdev"]
"V_1082"        [1, "C_42240 - URL: /access"]
"V_1083"        [1, "C_42448 - URL: /msaccesssupport"]
"V_1084"        [1, "C_42516 - URL: /uk"]
"V_1085"        [1, "C_42577 - URL: /exchangesupport"]
"V_1086"        [1, "C_40233 - URL: /oem"]
"V_1087"        [1, "C_42577 - URL: /proxy"]
"V_1088"        [1, "C_42650 - URL: /outlook"]
"V_1089"        [1, "C_42598 - URL: /officereference"]
"V_1090"        [1, "C_42445 - URL: /gamessupport"]
"V_1091"        [1, "C_42650 - URL: /hwdev"]
"V_1092"        [1, "C_41717 - URL: /vfoxpro"]
"V_1093"        [1, "C_42025 - URL: /vba"]
"V_1094"        [1, "C_39554 - URL: /mshome"]
"V_1095"        [1, "C_42234 - URL: /catalog"]
"V_1096"        [1, "C_42566 - URL: /mspress"]
"V_1097"        [1, "C_42435 - URL: /latam"]
"V_1098"        [1, "C_42616 - URL: /devonly"]
"V_1099"        [1, "C_42543 - URL: /cio"]
"V_1100"        [1, "C_42568 - URL: /education"]
"V_1101"        [1, "C_41626 - URL: /oledb"]
"V_1102"        [1, "C_42546 - URL: /homeessentials"]
"V_1103"        [1, "C_41711 - URL: /works"]
"V_1104"        [1, "C_41560 - URL: /hk"]
"V_1105"        [1, "C_42281 - URL: /france"]
"V_1106"        [1, "C_41001 - URL: /cze"]
"V_1107"        [1, "C_38331 - URL: /slovakia"]
"V_1108"        [1, "C_42598 - URL: /teammanager"]
"V_1109"        [1, "C_42537 - URL: /technet"]
"V_1110"        [1, "C_41897 - URL: /mastering"]
"V_1111"        [1, "C_41318 - URL: /ssafe"]
"V_1112"        [1, "C_42445 - URL: /canada"]
"V_1113"        [1, "C_42682 - URL: /security"]
"V_1114"        [1, "C_41916 - URL: /servad"]
"V_1115"        [1, "C_36277 - URL: /hun"]
"V_1116"        [1, "C_40678 - URL: /switzerland"]
"V_1117"        [1, "C_41101 - URL: /sidewinder"]
"V_1118"        [1, "C_42598 - URL: /sql"]
"V_1119"        [1, "C_42453 - URL: /corpinfo"]
"V_1120"        [1, "C_10241 - URL: /switch"]
"V_1121"        [1, "C_41556 - URL: /magazine"]
"V_1122"        [1, "C_41995 - URL: /mindshare"]
"V_1123"        [1, "C_42708 - URL: /germany"]
"V_1124"        [1, "C_42667 - URL: /industry"]
"V_1125"        [1, "C_42237 - URL: /imagecomposer"]
"V_1126"        [1, "C_42030 - URL: /mediamanager"]
```

```
    "V_1127"        [1, "C_42699 - URL: /netshow"]
    "V_1128"        [1, "C_10286 - URL: /msf"]
    "V_1129"        [1, "C_27780 - URL: /ado"]
    "V_1130"        [1, "C_42603 - URL: /syspro"]
    "V_1131"        [1, "C_42418 - URL: /moneyzone"]
    "V_1132"        [1, "C_40053 - URL: /msmoneysupport"]
    "V_1133"        [1, "C_41992 - URL: /frontpagesupport"]
    "V_1134"        [1, "C_42479 - URL: /backoffice"]
    "V_1135"        [1, "C_42659 - URL: /mswordsupport"]
    "V_1136"        [1, "C_42364 - URL: /usa"]
    "V_1137"        [1, "C_42704 - URL: /mscorp"]
    "V_1138"        [1, "C_42572 - URL: /mind"]
    "V_1139"        [1, "C_41482 - URL: /k-12"]
    "V_1140"        [1, "C_42678 - URL: /netherlands"]
    "V_1141"        [1, "C_42605 - URL: /europe"]
    "V_1142"        [1, "C_42467 - URL: /southafrica"]
    "V_1143"        [1, "C_42286 - URL: /workshoop"]
    "V_1144"        [1, "C_41640 - URL: /devnews"]
    "V_1145"        [1, "C_39965 - URL: /vfoxprosupport"]
    "V_1146"        [1, "C_42269 - URL: /msp"]
    "V_1147"        [1, "C_42555 - URL: /msft"]
    "V_1148"        [1, "C_42697 - URL: /channel"]
    "V_1149"        [1, "C_39863 - URL: /adc"]
    "V_1150"        [1, "C_42197 - URL: /infoserv"]
    "V_1151"        [1, "C_41774 - URL: /mspowerpointsupport"]
    "V_1152"        [1, "C_42312 - URL: /rus"]
    "V_1153"        [1, "C_39053 - URL: /venezuela"]
    "V_1154"        [1, "C_42467 - URL: /project"]
    "V_1155"        [1, "C_42353 - URL: /sidewalk"]
    "V_1156"        [1, "C_41710 - URL: /powered"]
    "V_1157"        [1, "C_42001 - URL: /win32dev"]
    "V_1158"        [1, "C_42705 - URL: /imedia"]
    "V_1159"        [1, "C_41444 - URL: /transaction"]
    "V_1160"        [1, "C_41764 - URL: /visualcsupport"]
    "V_1161"        [1, "C_42263 - URL: /workssupport"]
    "V_1162"        [1, "C_42285 - URL: /infoservsupport"]
    "V_1163"        [1, "C_40475 - URL: /opentype"]
    "V_1164"        [1, "C_41702 - URL: /smsmgmt"]
    "V_1165"        [1, "C_42007 - URL: /poland"]
    "V_1166"        [1, "C_41248 - URL: /mexico"]
    "V_1167"        [1, "C_42650 - URL: /hwtest"]
    "V_1168"        [1, "C_42646 - URL: /salesinfo"]
    "V_1169"        [1, "C_42642 - URL: /msproject"]
    "V_1170"        [1, "C_41306 - URL: /mail"]
    "V_1171"        [1, "C_42374 - URL: /merchant"]
    "V_1172"        [1, "C_42129 - URL: /belgium"]
    "V_1173"        [1, "C_31767 - URL: /moli"]
    "V_1174"        [1, "C_40040 - URL: /nz"]
    "V_1175"        [1, "C_41737 - URL: /msprojectsupport"]
    "V_1176"        [1, "C_41737 - URL: /jscript"]
    "V_1177"        [1, "C_42437 - URL: /events"]
    "V_1178"        [1, "C_31500 - URL: /msdownload."]
    "V_1179"        [1, "C_41490 - URL: /colombia"]
    "V_1180"        [1, "C_35728 - URL: /slovenija"]
    "V_1181"        [1, "C_42083 - URL: /kidssupport"]
    "V_1182"        [1, "C_38633 - URL: /fortran"]
    "V_1183"        [1, "C_42613 - URL: /italy"]
    "V_1184"        [1, "C_42626 - URL: /msexcelsupport"]
    "V_1185"        [1, "C_41832 - URL: /sna"]
    "V_1186"        [1, "C_42539 - URL: /college"]
    "V_1187"        [1, "C_42036 - URL: /odbc"]
    "V_1188"        [1, "C_42506 - URL: /korea"]
    "V_1189"        [1, "C_41768 - URL: /internet"]
    "V_1190"        [1, "C_41570 - URL: /repository"]
    "V_1191"        [1, "C_41812 - URL: /management"]
    "V_1192"        [1, "C_38976 - URL: /visualjsupport"]
    "V_1193"        [1, "C_41518 - URL: /offdevsupport"]
    "V_1194"        [1, "C_40708 - URL: /china"]
    "V_1195"        [1, "C_40458 - URL: /portugal"]
    "V_1196"        [1, "C_11431 - URL: /ie40"]
    "V_1197"        [1, "C_42285 - URL: /sqlsupport"]
    "V_1198"        [1, "C_40310 - URL: /pictureit"]
    "V_1199"        [1, "C_11644 - URL: /feedback"]
    "V_1200"        [1, "C_42451 - URL: /benelux"]
    "V_1201"        [1, "C_42203 - URL: /hardware"]
    "V_1202"        [1, "C_41172 - URL: /advtech"]
    "V_1203"        [1, "C_42518 - URL: /danmark"]
    "V_1204"        [1, "C_40792 - URL: /msscheduleplus"]
```

```
"V_1204"        [1, "C_40791 - URL: /msscheduleplus"]
"V_1205"        [1, "C_41597 - URL: /hardwaresupport"]
"V_1206"        [1, "C_42321 - URL: /select"]
"V_1207"        [1, "C_42008 - URL: /icp"]
"V_1208"        [1, "C_41548 - URL: /israel"]
"V_1209"        [1, "C_42513 - URL: /turkey"]
"V_1210"        [1, "C_31871 - URL: /snasupport"]
"V_1211"        [1, "C_42344 - URL: /smsmgmtsupport"]
"V_1212"        [1, "C_42071 - URL: /worldwide"]
"V_1213"        [1, "C_37985 - URL: /corporate"]
"V_1214"        [1, "C_35031 - URL: /finserv"]
"V_1215"        [1, "C_42572 - URL: /developer"]
"V_1216"        [1, "C_41329 - URL: /vrml"]
"V_1217"        [1, "C_38711 - URL: /ireland"]
"V_1218"        [1, "C_42541 - URL: /publishersupport"]
"V_1219"        [1, "C_20439 - URL: /ads"]
"V_1220"        [1, "C_41611 - URL: /macofficesupport"]
"V_1221"        [1, "C_41273 - URL: /mstv"]
"V_1222"        [1, "C_42103 - URL: /msofc"]
"V_1223"        [1, "C_42649 - URL: /finland"]
"V_1224"        [1, "C_40025 - URL: /atec"]
"V_1225"        [1, "C_42453 - URL: /piracy"]
"V_1226"        [1, "C_41980 - URL: /msschedplussupport"]
"V_1227"        [1, "C_42435 - URL: /argentina"]
"V_1228"        [1, "C_40882 - URL: /vtest"]
"V_1229"        [1, "C_26913 - URL: /uruguay"]
"V_1230"        [1, "C_40928 - URL: /mailsupport"]
"V_1231"        [1, "C_41626 - URL: /win32devsupport"]
"V_1232"        [1, "C_37637 - URL: /standards"]
"V_1233"        [1, "C_14363 - URL: /vbscripts"]
"V_1234"        [1, "C_42626 - URL: /off97cat"]
"V_1235"        [1, "C_37345 - URL: /onlineeval"]
"V_1236"        [1, "C_38325 - URL: /globaldev"]
"V_1237"        [1, "C_32583 - URL: /devdays"]
"V_1238"        [1, "C_26885 - URL: /exceldev"]
"V_1239"        [1, "C_38020 - URL: /msconsult"]
"V_1240"        [1, "C_39891 - URL: /thailand"]
"V_1241"        [1, "C_34793 - URL: /india"]
"V_1242"        [1, "C_41937 - URL: /msgarden"]
"V_1243"        [1, "C_28084 - URL: /usability"]
"V_1244"        [1, "C_41835 - URL: /devwire"]
"V_1245"        [1, "C_31934 - URL: /ofc"]
"V_1246"        [1, "C_42323 - URL: /gamesdev"]
"V_1247"        [1, "C_30024 - URL: /wineguide"]
"V_1248"        [1, "C_18347 - URL: /softimage"]
"V_1249"        [1, "C_41914 - URL: /fortransupport"]
"V_1250"        [1, "C_37938 - URL: /middleeast"]
"V_1251"        [1, "C_40693 - URL: /referencesupport"]
"V_1252"        [1, "C_37561 - URL: /giving"]
"V_1253"        [1, "C_32638 - URL: /worddev"]
"V_1254"        [1, "C_20190 - URL: /ie3"]
"V_1255"        [1, "C_22918 - URL: /msmq"]
"V_1256"        [1, "C_30930 - URL: /sia"]
"V_1257"        [1, "C_40127 - URL: /devvideos"]
"V_1258"        [1, "C_30514 - URL: /peru"]
"V_1259"        [1, "C_21424 - URL: /controls"]
"V_1260"        [1, "C_21894 - URL: /trial"]
"V_1261"        [1, "C_36145 - URL: /diyguide"]
"V_1262"        [1, "C_37425 - URL: /chile"]
"V_1263"        [1, "C_27503 - URL: /services"]
"V_1264"        [1, "C_40427 - URL: /se"]
"V_1265"        [1, "C_39038 - URL: /ssafesupport"]
"V_1266"        [1, "C_35105 - URL: /licenses"]
"V_1267"        [1, "C_42071 - URL: /caribbean"]
"V_1268"        [1, "C_27503 - URL: /javascript"]
"V_1269"        [1, "C_41054 - URL: /business"]
"V_1270"        [1, "C_28493 - URL: /developr"]
"V_1271"        [1, "C_28493 - URL: /mdsn"]
"V_1272"        [1, "C_28493 - URL: /softlib"]
"V_1273"        [1, "C_28493 - URL: /mdn"]
"V_1274"        [1, "C_28493 - URL: /pdc"]
"V_1275"        [1, "C_28903 - URL: /security."]
"V_1276"        [1, "C_40810 - URL: /vtestsupport"]
"V_1277"        [1, "C_30111 - URL: /stream"]
"V_1278"        [1, "C_41317 - URL: /hed"]
"V_1279"        [1, "C_31062 - URL: /msgolf"]
"V_1280"        [1, "C_41643 - URL: /music"]
"V_1281"        [1, "C_37099 - URL: /intellimouse"]
"V_1282"        [1, "C_41244 - URL: /home"]
```

```
        "V_1282"        [1, "C_41244 - URL: /home"]
        "V_1283"        [1, "C_41033 - URL: /cinemania"]
        "V_1284"        [1, "C_41108 - URL: /partner"]
        "V_1295"        [1, "C_42616 - URL: /train"]
```

HW 4.5

Here you will use a different dataset consisting of word-frequency distributions for 1,000 Twitter users. These Twitter users use language in very different ways, and were classified by hand according to the criteria:

- 0: Human, where only basic human-human communication is observed.
- 1: Cyborg, where language is primarily borrowed from other sources (e.g., jobs listings, classifieds postings, advertisements, etc...).
- 2: Robot, where language is formulaically derived from unrelated sources (e.g., weather/seismology, police/fire event logs, etc...).
- 3: Spammer, where language is replicated to high multiplicity (e.g., celebrity obsessions, personal promotion, etc...)

Check out the preprints of our recent research, which spawned this dataset:

http://arxiv.org/abs/1505.04342 (http://arxiv.org/abs/1505.04342) http://arxiv.org/abs/1508.01843 (http://arxiv.org/abs/1508.01843)

The main data lie in the accompanying file:

topUsers_Apr-Jul_2014_1000-words.txt

and are of the form:

USERID,CODE,TOTAL,WORD1_COUNT,WORD2_COUNT,.... . .

where

USERID = unique user identifier CODE = 0/1/2/3 class code TOTAL = sum of the word counts

Using this data, you will implement a 1000-dimensional K-means algorithm in MrJob on the users by their 1000-dimensional word stripes/vectors using several centroid initializations and values of K.

Note that each "point" is a user as represented by 1000 words, and that word-frequency distributions are generally heavy-tailed power-laws (often called Zipf distributions), and are very rare in the larger class of discrete, random distributions. For each user you will have to normalize by its "TOTAL" column. Try several parameterizations and initializations:

- (A) K=4 uniform random centroid-distributions over the 1000 words
- (B) K=2 perturbation-centroids, randomly perturbed from the aggregated (user-wide) distribution
- (C) K=4 perturbation-centroids, randomly perturbed from the aggregated (user-wide) distribution
- (D) K=4 "trained" centroids, determined by the sums across the classes.

and iterate until a threshold (try 0.001) is reached. After convergence, print out a summary of the classes present in each cluster. In particular, report the composition as measured by the total portion of each class type (0-3) contained in each cluster, and discuss your findings and any differences in outcomes across parts A-D.

Note that you do not have to compute the aggregated distribution or the class-aggregated distributions, which are rows in the auxiliary file:

topUsers_Apr-Jul_2014_1000-words_summaries.txt

Extra Notes:

- For (A), we select 4 users randomly from a uniform distribution [1,...,1,000]
- For (B), (C), and (D) you will have to use data from the auxiliary file:

topUsers_Apr-Jul_2014_1000-words_summaries.txt

This file contains 5 special word-frequency distributions:

- (1) The 1000-user-wide aggregate, which you will perturb for initializations in parts (B) and (C), and
- (2-5) The 4 class-level aggregates for each of the user-type classes (0/1/2/3)

In part (D), just use the (row-normalized) class-level aggregates as 'trained' starting centroids (the training is already done for you!). In parts (B) and (C), you will have to perturb the 1000-user aggregate (after initially normalizing by its sum, which is also provided). So if in (B) you want to create 2 perturbations of the aggregate, start with (1), normalize, and generate 1000 random numbers uniformly from the unit interval (0,1) twice (for two centroids), using:

```
from numpy import random
numbers = random.sample(1000)
```

Take these 1000 numbers and add them (component-wise) to the 1000-user aggregate, and then renormalize to obtain one of your aggregate-perturbed initial centroids perturbed initial centroids.

Data pre-processing & Initialization

- (A) K=4 uniform random centroid-distributions over the 1000 words
- (B) K=2 perturbation-centroids, randomly perturbed from the aggregated (user-wide) distribution
- (C) K=4 perturbation-centroids, randomly perturbed from the aggregated (user-wide) distribution
- (D) K=4 "trained" centroids, determined by the sums across the classes.

In [40]:

```
import numpy as np
import math

# load the raw data
raw_data = np.genfromtxt('topUsers_Apr-Jul_2014_1000-words.txt', delimiter=',')
# normalize
train_data = [a[3:]/a[2] for a in raw_data]
# save as training data file
np.savetxt('HW4_5_train_data.csv', train_data, delimiter = ",")

# load the auxiliary file
!tail -n +2 topUsers_Apr-Jul_2014_1000-words_summaries.txt | cut -d ',' -f 3-2000 > HW4_5_aux.txt
aux_data = np.genfromtxt('HW4_5_aux.txt', delimiter=',')
aux_norm = [a[1:]/a[0] for a in aux_data]

# get centroids initialization A
K, n = 4, 1000
idx = np.random.randint(n, size=4)
centroids = [train_data[i] for i in idx]
np.savetxt('HW4_5_init_A.txt', centroids, delimiter = ",")

# get centroids initialization B
K, n = 2, 1000
centroids = [[a+p for a, p in zip(aux_norm[0], np.random.sample(n))] for i in range(K)]
norm = np.sum(centroids, axis = 1)
cen_norm = [c/n for c,n in zip(centroids, norm)]
np.savetxt('HW4_5_init_B.txt', cen_norm, delimiter = ",")

# get centroids initialization C
K, n = 4, 1000
centroids = [[a+p for a, p in zip(aux_norm[0], np.random.sample(n))] for i in range(K)]
norm = np.sum(centroids, axis = 1)
cen_norm = [c/n for c,n in zip(centroids, norm)]
np.savetxt('HW4_5_init_C.txt', cen_norm, delimiter = ",")

# get centroids initialization D
centroids = aux_norm[1:]
np.savetxt('HW4_5_init_D.txt', centroids, delimiter = ",")

print 'Data pre-processing done!'
```

```
Data pre-processing done!
```

MrJob to provide *one* training iteration for k-mean model

- ready for running on Hadoop
- assuming a file *Centroids.txt*, that stores centroids coordinates, will be updated on Hadoop cluster by the training handler/driver after each iteration
- **input**: a csv of all training sample coordinates
- **output**: centroids coordinates

In [88]:

```
%%writefile HW4_5_Kmeans.py

import numpy as np
from mrjob.job import MRJob
from mrjob.step import MRStep
from itertools import chain
import subprocess

#Calculate find the nearest centroid for data point
def MinDist(datapoint, centroid_points):
    datapoint = np.array(datapoint)
    centroid_points = np.array(centroid_points)
    diff = datapoint - centroid_points
    diffsq = diff*diff
    # Get the nearest centroid for each instance
    # get the nearest centroid for each instance
    minidx = np.argmin(list(diffsq.sum(axis = 1)))
    return minidx

#Check whether centroids converge
def stop_criterion(centroid_points_old, centroid_points_new,T):
    oldvalue = list(chain(*centroid_points_old))
    newvalue = list(chain(*centroid_points_new))
    Diff = [abs(x-y) for x, y in zip(oldvalue, newvalue)]
    Flag = True
    for i in Diff:
        if(i>T):
            Flag = False
            break
    return Flag

class MRKmeans(MRJob):
    centroid_points=[]
    # TODO: figure out why set from mapper_init doesn't preserve value
    k, n = 2, 1000

    def steps(self):
        return [
            MRStep(mapper_init = self.mapper_init,
                   mapper=self.mapper,
                   combiner = self.combiner,
                   reducer=self.reducer)
               ]

    #load centroids info from file
    def mapper_init(self):
        hadoopPath = 'Centroids.txt'
        localPath = '/Users/leiyang/GitHub/mids/w261/HW4-Questions/ref/Centroids.txt'
        cat = subprocess.Popen(["cat", hadoopPath], stdout=subprocess.PIPE)
        self.centroid_points = [map(float, s.strip().split(',')) for s in cat.stdout]

    #load data and output the nearest centroid index and data point
    def mapper(self, _, line):
        D = (map(float,line.split(',')))
        yield int(MinDist(D,self.centroid_points)), (D,1)

    #Combine sum of data points locally
    def combiner(self, idx, inputdata):
        num = 0
        sum_d = [0]*self.n
        # iterate through the generator
        for d,c in inputdata:
            num += c
            sum_d = [a+b for a,b in zip(sum_d, d)]
        yield idx, (sum_d, num)

    #Aggregate sum for each cluster and then calculate the new centroids
    def reducer(self, idx, inputdata):
        num = [0]*self.k
        centroids = [[0]*self.n for i in range(self.k)]
        # iterate through the generator
        for d, c in inputdata:
            num[idx] += c
            centroids[idx] = [a+b for a,b in zip(centroids[idx], d)]
        # recalculate centroids
        centroids[idx] = [a/num[idx] for a in centroids[idx]]
        yield idx, (centroids[idx])

if __name__ == '__main__':
    MRKmeans.run()
```

```
Overwriting HW4_5_Kmeans.py
```

K-mean training driver

- control the training process
- update Centroids.txt file from Hadoop clusters after each iteration

In [84]:

```
%%writefile HW4_5_Kmeans_driver.py
#!/usr/bin/python

#%reload_ext autoreload
```

```
#%autoreload 2
import numpy as np
from HW4_5_Kmeans import MRKmeans, stop_criterion

# create MrJob: specify input, available file, hadoop home
mr_job = MRKmeans(args=['HW4_5_train_data.csv', '--file', 'Centroids.txt',
                        '--hadoop-home', '/usr/local/Cellar/hadoop/2.7.1/', '-r', 'hadoop'])

# load initial centroids
centroid_points = np.genfromtxt('Centroids.txt', delimiter=',')

# Update centroids iteratively
i = 1
while(1):
    # save previous centoids to check convergency
    centroid_points_old = centroid_points[:]
    np.savetxt('c_old', centroid_points_old, delimiter=',')
    print "Iteration "+str(i)+" ..."
    with mr_job.make_runner() as runner:
        runner.run()
        # stream_output: get access of the output
        for line in runner.stream_output():
            key,value = mr_job.parse_output_line(line)
            centroid_points[key] = value

    i += 1
    # put the latest centroids in the file for next iteration
    np.savetxt('Centroids.txt', centroid_points, delimiter = ",")

    # check stopping criterion
    if stop_criterion(np.genfromtxt('c_old', delimiter=','), centroid_points, 0.001):
        break

print '\nK-means training completed after %d iterations!' %(i-1)
```

```
Overwriting HW4_5_Kmeans_driver.py
```

Find cluster composition

In [49]:

```
%%writefile HW4_5_get_composition.py
#!/usr/bin/python

import numpy as np
import math
from HW4_5_Kmeans import MinDist

# load the centroids
centroids = np.genfromtxt('Centroids.txt', delimiter=',')
compo = [[0,0,0,0] for i in range(len(centroids))]

# load the raw data
raw_data = np.genfromtxt('topUsers_Apr-Jul_2014_1000-words.txt', delimiter=',')

# find the cluster for each point, and record code count
for r in raw_data:
    code, point = r[1], r[3:]/r[2]
    compo[int(MinDist(point, centroids))][int(code)] += 1

# show results
for i in range(len(compo)):
    print '\n'
    for j in range(4):
        print 'Centroid %d - code %d: %d (%.2f%%)' %(i, j, compo[i][j], 100.0*compo[i][j]/sum(comp
o[i]))
```

```
Overwriting HW4_5_get_composition.py
```

Execution: (A) K=4 uniform random centroid-distributions over the 1000 words

- run MrJob to train k-means model
- check composition of each cluster

In [85]:

```
!cat HW4_5_init_A.txt > Centroids.txt
!python HW4_5_Kmeans_driver.py
!python HW4_5_get_composition.py
```

```
Iteration 1 ...
No handlers could be found for logger "mrjob.runner"
Iteration 2 ...
Iteration 3 ...
Iteration 4 ...
Iteration 5 ...
Iteration 6 ...
Iteration 7 ...

K-means training completed after 7 iterations!


Centroid 0 - code 0: 67 (45.89%)
Centroid 0 - code 1: 0 (0.00%)
Centroid 0 - code 2: 12 (8.22%)
Centroid 0 - code 3: 67 (45.89%)


Centroid 1 - code 0: 1 (0.76%)
Centroid 1 - code 1: 88 (66.67%)
Centroid 1 - code 2: 39 (29.55%)
Centroid 1 - code 3: 4 (3.03%)


Centroid 2 - code 0: 683 (94.73%)
Centroid 2 - code 1: 3 (0.42%)
Centroid 2 - code 2: 3 (0.42%)
Centroid 2 - code 3: 32 (4.44%)


Centroid 3 - code 0: 1 (100.00%)
Centroid 3 - code 1: 0 (0.00%)
Centroid 3 - code 2: 0 (0.00%)
Centroid 3 - code 3: 0 (0.00%)
```

Execution: (B) K=2 perturbation-centroids, randomly perturbed from the aggregated (user-wide) distribution

In [89]:

```
!cat HW4_5_init_B.txt > Centroids.txt
!python HW4_5_Kmeans_driver.py
!python HW4_5_get_composition.py
```

```
Iteration 1 ...
No handlers could be found for logger "mrjob.runner"
Iteration 2 ...
Iteration 3 ...
Iteration 4 ...

K-means training completed after 4 iterations!


Centroid 0 - code 0: 751 (86.62%)
Centroid 0 - code 1: 3 (0.35%)
Centroid 0 - code 2: 14 (1.61%)
Centroid 0 - code 3: 99 (11.42%)


Centroid 1 - code 0: 1 (0.75%)
Centroid 1 - code 1: 88 (66.17%)
Centroid 1 - code 2: 40 (30.08%)
Centroid 1 - code 3: 4 (3.01%)
```

Execution: (C) K=4 perturbation-centroids, randomly perturbed from the aggregated (user-wide) distribution

In [86]:

```
!cat HW4_5_init_C.txt > Centroids.txt
!python HW4_5_Kmeans_driver.py
!python HW4_5_get_composition.py
```

```
Iteration 1 ...
No handlers could be found for logger "mrjob.runner"
Iteration 2 ...
Iteration 3 ...
Iteration 4 ...
Iteration 5 ...
Iteration 6 ...

K-means training completed after 6 iterations!


Centroid 0 - code 0: 0 (0.00%)
Centroid 0 - code 1: 2 (13.33%)
Centroid 0 - code 2: 13 (86.67%)
Centroid 0 - code 3: 0 (0.00%)


Centroid 1 - code 0: 751 (87.53%)
Centroid 1 - code 1: 3 (0.35%)
Centroid 1 - code 2: 5 (0.58%)
Centroid 1 - code 3: 99 (11.54%)


Centroid 2 - code 0: 1 (1.32%)
Centroid 2 - code 1: 35 (46.05%)
Centroid 2 - code 2: 36 (47.37%)
Centroid 2 - code 3: 4 (5.26%)


Centroid 3 - code 0: 0 (0.00%)
Centroid 3 - code 1: 51 (100.00%)
Centroid 3 - code 2: 0 (0.00%)
Centroid 3 - code 3: 0 (0.00%)
```

Execution: (D) K=4 "trained" centroids, determined by the sums across the classes.

In [87]:

```
!cat HW4_5_init_D.txt > Centroids.txt
!python HW4_5_Kmeans_driver.py
!python HW4_5_get_composition.py
```

```
Iteration 1 ...
No handlers could be found for logger "mrjob.runner"
Iteration 2 ...
Iteration 3 ...
Iteration 4 ...
Iteration 5 ...

K-means training completed after 5 iterations!


Centroid 0 - code 0: 749 (93.16%)
Centroid 0 - code 1: 3 (0.37%)
Centroid 0 - code 2: 14 (1.74%)
Centroid 0 - code 3: 38 (4.73%)


Centroid 1 - code 0: 0 (0.00%)
Centroid 1 - code 1: 51 (100.00%)
Centroid 1 - code 2: 0 (0.00%)
Centroid 1 - code 3: 0 (0.00%)


Centroid 2 - code 0: 1 (1.22%)
Centroid 2 - code 1: 37 (45.12%)
Centroid 2 - code 2: 40 (48.78%)
Centroid 2 - code 3: 4 (4.88%)


Centroid 3 - code 0: 2 (3.17%)
Centroid 3 - code 1: 0 (0.00%)
Centroid 3 - code 2: 0 (0.00%)
Centroid 3 - code 3: 61 (96.83%)
```

HW4.6 (OPTIONAL) Scaleable K-MEANS++

- Read the following paper entitled "Scaleable K-MEANS++" located at:

http://theory.stanford.edu/~sergei/papers/vldb12-kmpar.pdf (http://theory.stanford.edu/~sergei/papers/vldb12-kmpar.pdf)

- In MrJob, implement K-MEANS|| and compare with a random initialization for the dataset above. Report on the number passes over the training data, and time required to run all clustering algorithms.

No code, just implementation notes

A MapReduce job to perform one iteration of finding centroid:

- **mapper_init():** load current centroids set C
- **mapper():** calculate $d^2(x, C)$, and emit:
  - $(*, d^2(x, C), None)$, order inversion to calcuate $\sum_{x \in X} d^2(x, C)$
  - $(id, d^2(x, C), point\_coordinates)$ in numerical ascending order
- **single reducer():**
  - calculate $\sum_{x \in X} d^2(x, C)$ with the emits with $*$ keys
  - generate a random value L such that $0 < L \leq \sum_{x \in X} d^2(x, C)$
  - choose a new center c_i, selecting the $c_i = x_j$ such that $\sum_{m=1}^{j-1} d^2(x_m, C) < L \leq \sum_{m=1}^{j} d^2(x_m, C)$

A driver program to control the loops:

1. randomly select a point from X, save to a file
2. run mrjob in loop
   - specify the centroids file to the job
   - after each run, collect centroids from the reducer, update the file
   - check stopping condition
3. merge centroids to have the final K, if necessary

In [ ]:

HW4.7 (OPTIONAL) Canopy Clustering

An alternative way to intialize the k-means algorithm is the canopy clustering. The canopy clustering algorithm is an unsupervised pre-clustering algorithm introduced by Andrew McCallum, Kamal Nigam and Lyle Ungar in 2000. It is often used as preprocessing step for the K-means algorithm or the Hierarchical clustering algorithm. It is intended to speed up clustering operations on large data sets, where using another algorithm directly may be impractical due to the size of the data set.

For more details on the Canopy Clustering algorithm see:

https://en.wikipedia.org/wiki/Canopy_clustering_algorithm (https://en.wikipedia.org/wiki/Canopy_clustering_algorithm)

- Plot the initialation centroids and the centroid trajectory as the Canopy Clustering based K-MEANS algorithm iterates.
- Repeat this for a random initalization (i.e., pick a training vector at random for each inital centroid) of the kmeans algorithm.
- Comment on the trajectories of both algorithms.
- Report on the number passes over the training data, and time required to run both clustering algorithms.
- Also report the rand index score for both algorithms and comment on your findings.

4.7.1

- Apply your implementation Canopy Clustering based K-MEANS algorithm to the dataset in HW 4.5 and compare to the a random initalization (i.e., pick a training vector at random for each inital centroid) of the kmeans algorithm.
- Report on the number passes over the training data, and time required to run both clustering algorithms.
- Also report the rand index score for both algorithms and comment on your findings.

MrJob for Canopy Clustering - *(still in code debug)*

job to get all canopies

- **mapper**: creates canopies on parallel for each chunk of data the mapper gets [ref (http://www.kamalnigam.com/papers/canopy-kdd00.pdf)]
  - as record streams through, decide if it:
    - belongs to the inner circle of any existin canopy
    - belongs to the outer circle of any existin canopy

    - belongs to a new canopy
  - for each role the record plays, emit a line such that:
    - all inner and/or outer circle points of a canopy will arrive reducer consecutively (through partitioner)
    - *key*: canopy ID - can use message ID
    - *value*: (point coordinates, 1)
  - **note**: the mapper will maintain a collection of canopy centroids, given that we want more mappers, caching centroid is presumably ok
- **combiner**: calculate intermediate canopy centroid
  - need to maintain record count if adopt
- **single reducer**: get all canopies and yield the centroids
  - get centroid for each canopy by averaging all inner and outer circle points
- merge centroids if necessary to have the final K as K-mean initialization

In [34]:
```
%%writefile HW4_7_Canopy.py

#import numpy as np
from mrjob.job import MRJob
from mrjob.step import MRStep
from itertools import chain
from math import pow

class MRCanopy(MRJob):
    centroids = {}
    # TODO: figure out why set from mapper_init doesn't preserve value
    k, n = 2, 1000
    T1, T2 = 0.001, 0.0001

    def steps(self):
        return [
            # job 1
            MRStep(mapper=self.mapper1
                   ,combiner = self.combiner1
                   ,reducer=self.reducer1
                  )
               ]

    # Load data and output each point with the canopy it belongs to or defines
    def mapper1(self, _, line):
        D = [map(float, line.strip().split(','))]
        name, point = str(D[0]), D[1:]
        yield name, (point, 1)
        if len(self.centroids) == 0:
            # first point, add canopy
            self.centroids[name] = point
            #print 'first'
            yield name, (point, 1)
        else:
            # compare with each centroid to determine status
            isNew = True
            for cen in self.centroids:
                # calculate distance
                dist = sum([pow(a-b, 2) for a,b in zip(self.centroids[cen], point)])
                # inside inner circle of cen
                if self.T2 > dist:
                    isNew = False
                    #print 'inner'
                    yield cen, (point, 1)
                # in between circles of cen
                elif self.T2 < dist and dist < self.T1:
                    #print 'close'
                    yield cen, (point, 1)
            # not in anyone's inner circle, add new canopy
            if isNew:
                self.centroids[name] = point
                #print 'new'
                yield name, (point, 1)


    # Combine sum of data points locally
    def combiner1(self, name, inputdata):
        num = 0
        sum_d = [0]*self.n
        # iterate through the generator
        for d,c in inputdata:
            num += c
            sum_d = [a+b for a,b in zip(sum_d, d)]
            sum_d = [a+b for a,b in zip(sum_d, d)]
        yield name, (sum_d, num)

    # Aggregate sum for each canopy, and then calculate the new centroids
    def reducer1(self, name, inputdata):
        num = 0 #[0]*self.k
        centroid = [0]*self.n
        current_canopy = None
        if current_canopy != name:
            # aggregation completes for previous canopy, emit
            centroid = [a/num for a in centroid]
            yield current_canopy, centroid
            # reset for new canopy
            current_canopy = name
            num = 0
            centroid = [0]*self.n
        # iterate through the generator
        for d, c in inputdata:
            num += c
            centroid = [a+b for a,b in zip(centroid, d)]


if __name__ == '__main__':
    MRCanopy.run()
```

```
Overwriting HW4_7_Canopy.py
```

Get training data from HW4.5

In [13]:

```
import numpy as np
# load the raw data
raw_data = np.genfromtxt('topUsers_Apr-Jul_2014_1000-words.txt', delimiter=',')
# normalize
train_data = [np.insert(a[3:]/a[2],0,a[0]) for a in raw_data]
# save as training data file
np.savetxt('HW4_7_train_data.csv', train_data, delimiter = ",")
```

Execute the MrJob:

In [35]:

```
!python HW4_7_Canopy.py HW4_7_train_data.csv > debug
```

```
no configs found; falling back on auto-configuration
no configs found; falling back on auto-configuration
creating tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_7_Canopy.leiyang.20160
210.041424.388486

PLEASE NOTE: Starting in mrjob v0.5.0, protocols will be strict by default. It's recommended you r
un your job with --strict-protocols or set up mrjob.conf as described at https://pythonhosted.org/
mrjob/whats-new.html#ready-for-strict-protocols

writing to /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_7_Canopy.leiyang.20160210.041424.3
88486/step-0-mapper_part-00000
Counters from step 1:
  (no counters found)
Moving /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_7_Canopy.leiyang.20160210.041424.38848
6/step-0-mapper_part-00000 -> /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_7_Canopy.leiyan
g.20160210.041424.388486/output/part-00000
Streaming final output from /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_7_Canopy.leiyang.
20160210.041424.388486/output
removing tmp directory /var/folders/tx/5ldq67q511q8wqwqkvptnxd00000gn/T/HW4_7_Canopy.leiyang.20160
210.041424.388486
```

stop yarn, hdfs, and job history

In [1]:

```
!/usr/local/Cellar/hadoop/2*/sbin/stop-yarn.sh
```