PB(9/30/2017)

Principles of Big Data Management Phase-1

Fall 2017

Submitted by: (Team 13)

Pragathi Thammaneni
Sridevi Mallipudi
Fatema Hasta

Objective:

- The main aim of this phase is to develop a system to store, analyze, and visualize a social network data.
- Tasks:
  1. Collect social network data (e.g. tweets) in JSON format.
  2. Extracting the hashtags, URLs from the collected tweets.
  3. Store the text content (e.g. tweet's text) from the data into a file in HDFS.
  4. Run a Word Count program in Apache Spark, Apache Hadoop on extracted hashtags, URLs files and store the output and log files locally.

Applications/Software's Used: Apache Spark, Hadoop, Scala, Cloudera, Twitter Developer Account, Python.

Collecting tweets from Twitter:

- Firstly, we have created a developer account in Twitter using below link. https://apps.twitter.com/
- Below are the variables that contains the user credentials to access Twitter API

  ➢ API ACCESS_TOKEN = " 1329248834-R3X8SrLzchMQyBOSRtV1QSJYfis2lidRex4RXeq"
  ➢ ACCESS_SECRET = " g6pwJFjRJTQK2cRi01rGV3kpr6C9v7sQNTjrtPLtiETc4"
  ➢ CONSUMER_KEY = " l82JLsLffFzGMrytXhF4DAnf2"
  ➢ CONSUMER_SECRET = " Fh4985bvDNeF2RurZsseuDYBnbFv3cLmpdyyizFapBkr1uS954"
- We have written python program that is used to fetch tweets in JSON format. (TweetsExtract.py)
- The extracted file in JSON format contains all the tweet details such as id, created at, text, URL, hashtags etc.
- From JSON tweets file only the hashtags and URLs content is extracted using scala programs each for hastags and URLs. Then fetched details are stored in files.

Run the Word Count program in Apache Spark and Apache Hadoop the extracted hashtags/URLs and collect the output and log files:

1. Word Count Program in Apache Spark:

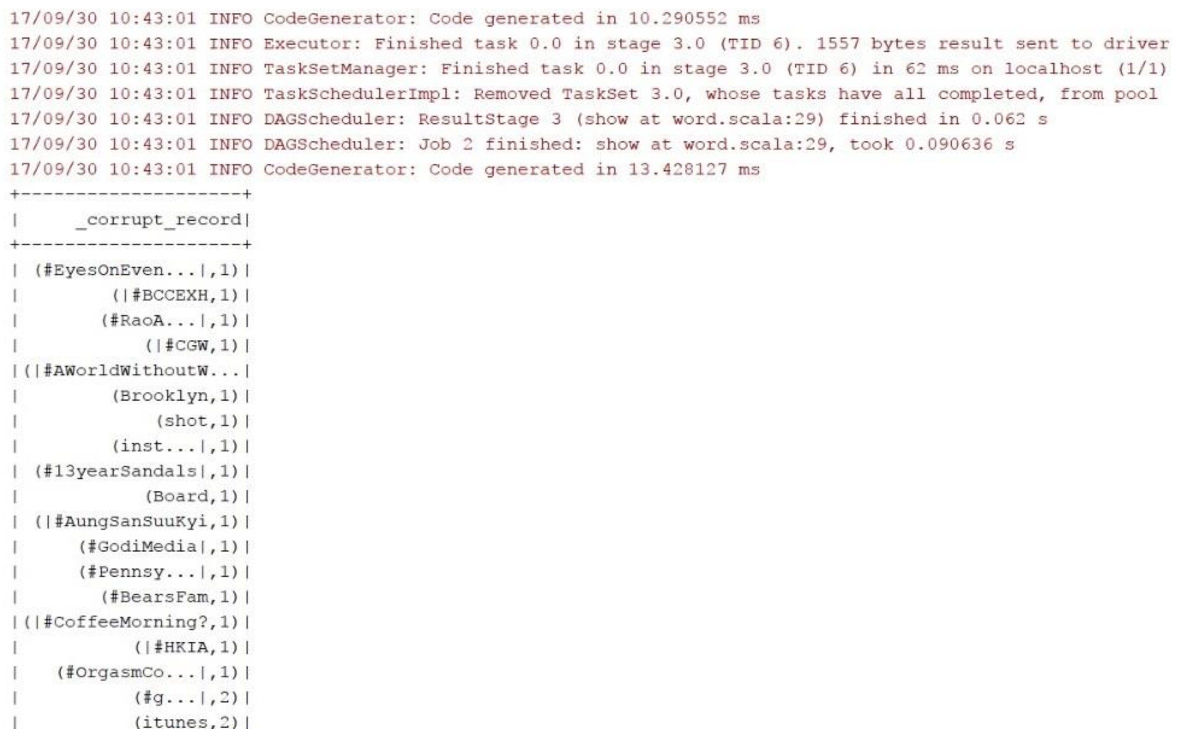
- We have used IntelliJ IDEA software to run word count program on extracted hastags/URLs using Apache Spark and Scala.
- In IntelliJ IDEA, Spark is integrated on Scala to execute word count program on extracted hashtags and URLs data.

  ➢ **val inputFile = "C:\\Users\\Praga\\Desktop\\hashtagExtractionOutput.txt"**
  ➢ **val input = sc.textFile(inputFile)**
    - sc.textFile in Spark, creates a RDD with each line as an element. For example, if there are 10 files in folder, 10 partitions will be created.
    - RDD means Resilient Distributed Datasets which are immutable and partitioned collection of records, which can only be created by coarse grained operations such as map, filter, group by etc.

- RDDs can only be created by reading data from a stable storage such as HDFS or by transformations on existing RDDs.

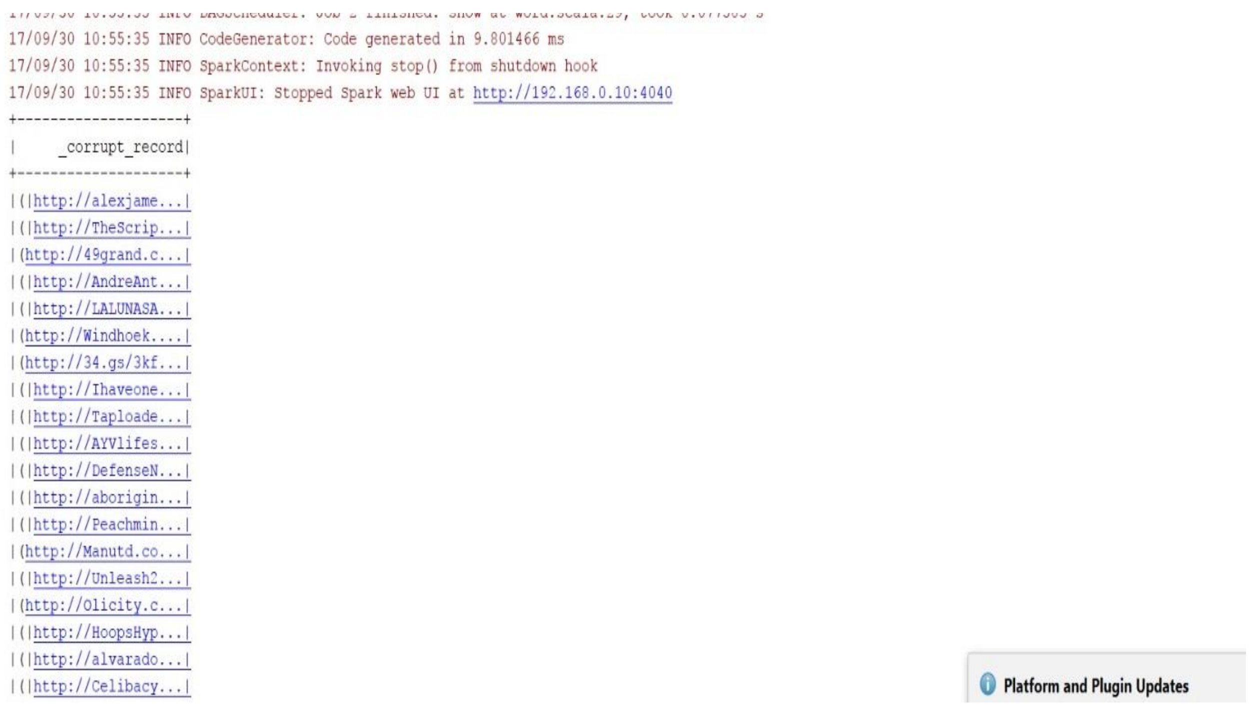
- **val words = input.flatMap(line => line.split(" "))**
- **val counts = words.map(word => (word, 1)).reduceByKey{case (x, y) => x + y}**
  - Once the file is read and stored in a variable textfile MapReduce function is used to execute word count program
  - MapReduce works by breaking the processing into 2 phases Map Phase and Reduce Phase.
  - Each phase has key-value pairs as input and output, by specifying two functions Map Function and Reduce Function.

- **val outputFile = "C:\\Users\\Praga\\Desktop\\outputFile"**
- **counts.saveAsTextFile(outputFile)**
  - Once the reduce function group the words with respective count, the data is stored in outputFile.



```
17/09/30 10:43:01 INFO CodeGenerator: Code generated in 10.290552 ms
17/09/30 10:43:01 INFO Executor: Finished task 0.0 in stage 3.0 (TID 6). 1557 bytes result sent to driver
17/09/30 10:43:01 INFO TaskSetManager: Finished task 0.0 in stage 3.0 (TID 6) in 62 ms on localhost (1/1)
17/09/30 10:43:01 INFO TaskSchedulerImpl: Removed TaskSet 3.0, whose tasks have all completed, from pool
17/09/30 10:43:01 INFO DAGScheduler: ResultStage 3 (show at word.scala:29) finished in 0.062 s
17/09/30 10:43:01 INFO DAGScheduler: Job 2 finished: show at word.scala:29, took 0.090636 s
17/09/30 10:43:01 INFO CodeGenerator: Code generated in 13.428127 ms
+--------------------+
|     _corrupt_record|
+--------------------+
| (#EyesOnEven...|,1)|
|         (|#BCCEXH,1)|
|        (#RaoA...|,1)|
|            (|#CGW,1)|
|(|#AWorldWithoutW...|
|         (Brooklyn,1)|
|             (shot,1)|
|        (inst...|,1)|
| (#13yearSandals|,1)|
|            (Board,1)|
| (|#AungSanSuuKyi,1)|
|     (#GodiMedia|,1)|
|     (#Pennsy...|,1)|
|        (#BearsFam,1)|
|(|#CoffeeMorning?,1)|
|           (|#HKIA,1)|
|   (#OrgasmCo...|,1)|
|           (#g...|,2)|
|           (itunes,2)|
```

Fig 1: Sample Word Count Output for hashtags



```
17/09/30 10:55:35 INFO CodeGenerator: Code generated in 9.801466 ms
17/09/30 10:55:35 INFO SparkContext: Invoking stop() from shutdown hook
17/09/30 10:55:35 INFO SparkUI: Stopped Spark web UI at http://192.168.0.10:4040
+--------------------+
|     _corrupt_record|
+--------------------+
|(|http://alexjame...|
|(|http://TheScrip...|
|(http://49grand.c...|
|(|http://AndreAnt...|
|(|http://LALUNASA...|
|(http://Windhoek....|
|(http://34.gs/3kf...|
|(|http://Ihaveone...|
|(|http://Taploade...|
|(|http://AYVlifes...|
|(|http://DefenseN...|
|(|http://aborigin...|
|(|http://Peachmin...|
|(http://Manutd.co...|
|(|http://Unleash2...|
|(http://Olicity.c...|
|(|http://HoopsHyp...|
|(|http://alvarado...|
|(|http://Celibacy...|
```

Fig 2: Sample Word Count Output for URLs

2. Word Count Program in Apache Hadoop:

In order to run the word count program using Apache Hadoop we have written the code in java. In the program TokenizerMapper class is created by extending the Mapper class and the map function is implemented by overriding the map method in the Mapper class. The mapper functions takes a key-value pair as an input and outputs a key-values pair as an output.

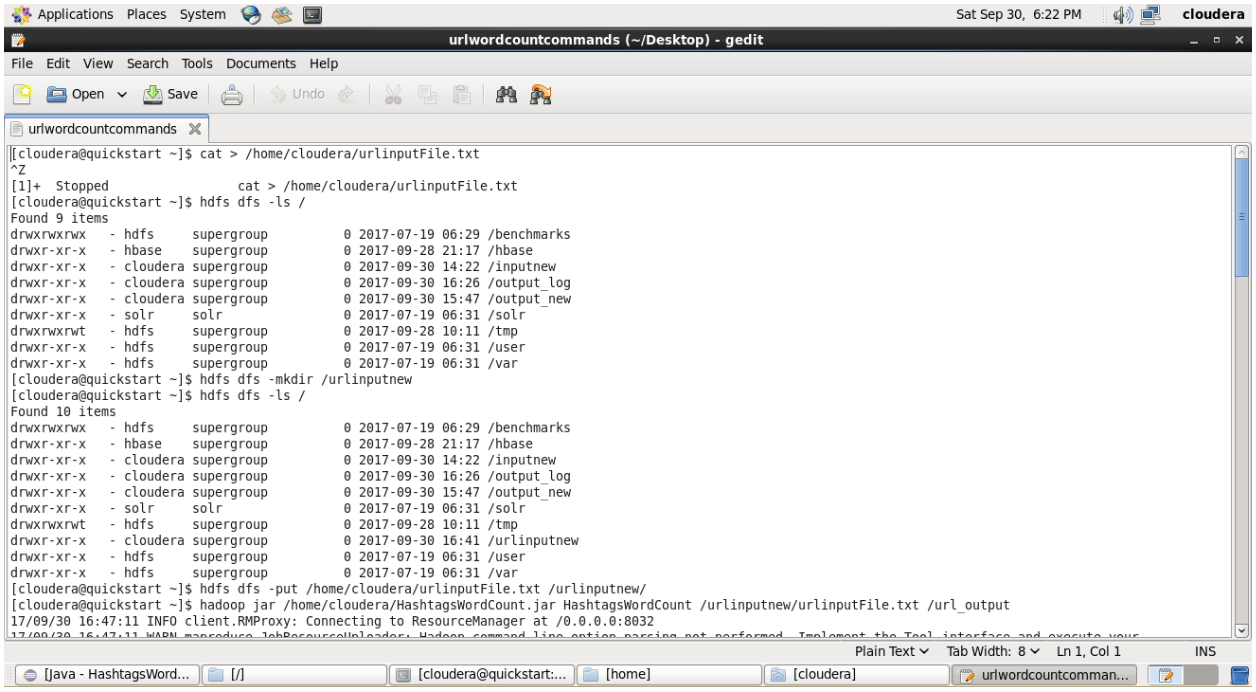
And IntSumReducer class is created by extending the org.apache.hadoop.mapreduce.Reducer class and the reduce method is implemented by overriding the reduce method from the Reducer class. The reduce function collects all the intermediate key-value pairs generated by the multiple map functions and will sum up all the occurrences of each word and output a key-value pair for each word in the text document.

And also the main method is used to setup all necessary configurations and runs the mapreduce job.

Then the .jar file is exported to cloudera to execute the word count for Hashtags and URLs using Apache Hadoop. Below are the steps followed:

- Created the input file for the MapReduce Program using below command and loaded extracted URLs into it.
  - $ cat > /home/cloudera/urlinputFile.txt
- Then a folder is created in HDFS and the input file is moved from local to HDFS using below commands.
  - $ hdfs dfs -mkdir /urlinputnew
  - $ hdfs dfs -put /home/cloudera/urlinputFile.txt /urlinputnew/

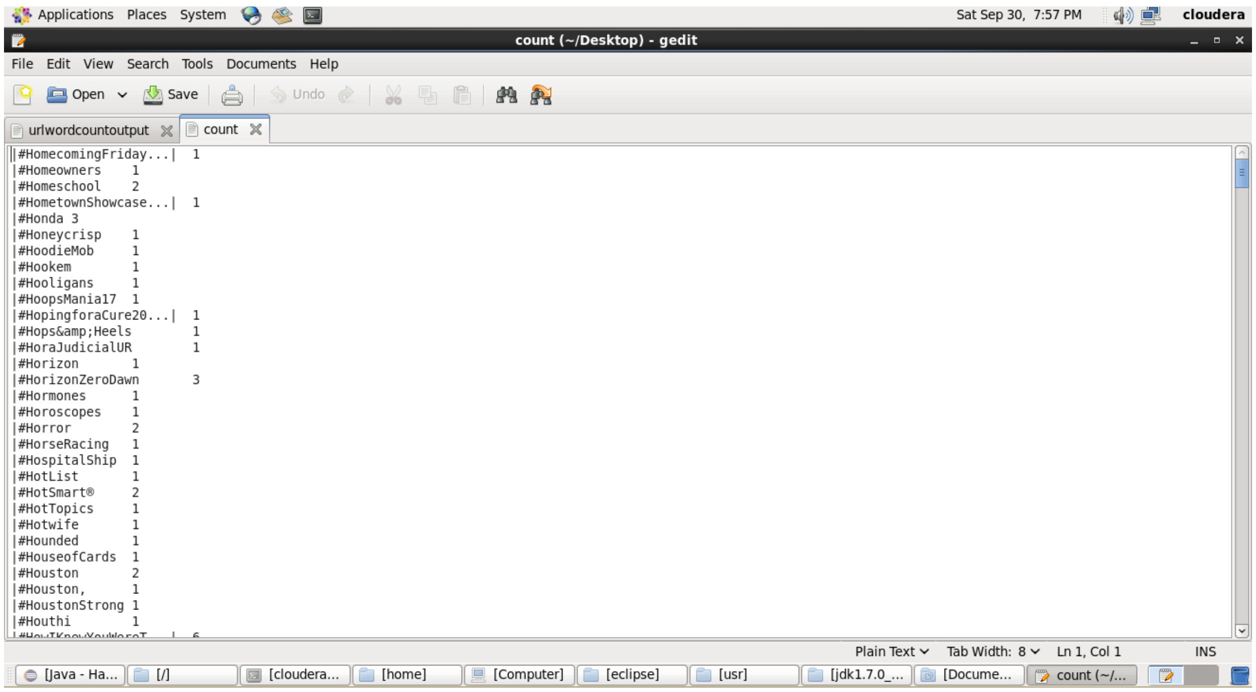
- To view the file moved to HDFS
  - ➢ $ hdfs dfs -cat /urlinputnew/urlinputFile.txt
- Then MapReduce program was run on Hadoop
  - ➢ $ hadoop jar /home/cloudera/HashtagsWordCount.jar HashtagsWordCount /urlinputnew/urlinputFile.txt /url_output
- To view the output File
  - ➢ $ hdfs dfs -ls /url_output
- Finally output is viewed in console using below command
  - ➢ $ hdfs dfs -cat /url_output/part-r-00000



```
[cloudera@quickstart ~]$ cat > /home/cloudera/urlinputFile.txt
^Z
[1]+  Stopped                 cat > /home/cloudera/urlinputFile.txt
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 9 items
drwxrwxrwx   - hdfs     supergroup          0 2017-07-19 06:29 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2017-09-28 21:17 /hbase
drwxr-xr-x   - cloudera supergroup          0 2017-09-30 14:22 /inputnew
drwxr-xr-x   - cloudera supergroup          0 2017-09-30 16:26 /output_log
drwxr-xr-x   - cloudera supergroup          0 2017-09-30 15:47 /output_new
drwxr-xr-x   - solr     solr                0 2017-07-19 06:31 /solr
drwxrwxrwt   - hdfs     supergroup          0 2017-09-28 10:11 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2017-07-19 06:31 /user
drwxr-xr-x   - hdfs     supergroup          0 2017-07-19 06:31 /var
[cloudera@quickstart ~]$ hdfs dfs -mkdir /urlinputnew
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 10 items
drwxrwxrwx   - hdfs     supergroup          0 2017-07-19 06:29 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2017-09-28 21:17 /hbase
drwxr-xr-x   - cloudera supergroup          0 2017-09-30 14:22 /inputnew
drwxr-xr-x   - cloudera supergroup          0 2017-09-30 16:26 /output_log
drwxr-xr-x   - cloudera supergroup          0 2017-09-30 15:47 /output_new
drwxr-xr-x   - solr     solr                0 2017-07-19 06:31 /solr
drwxrwxrwt   - hdfs     supergroup          0 2017-09-28 10:11 /tmp
drwxr-xr-x   - cloudera supergroup          0 2017-09-30 16:41 /urlinputnew
drwxr-xr-x   - hdfs     supergroup          0 2017-07-19 06:31 /user
drwxr-xr-x   - hdfs     supergroup          0 2017-07-19 06:31 /var
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/urlinputFile.txt /urlinputnew/
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/HashtagsWordCount.jar HashtagsWordCount /urlinputnew/urlinputFile.txt /url_output
17/09/30 16:47:11 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/09/30 16:47:11 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your
```

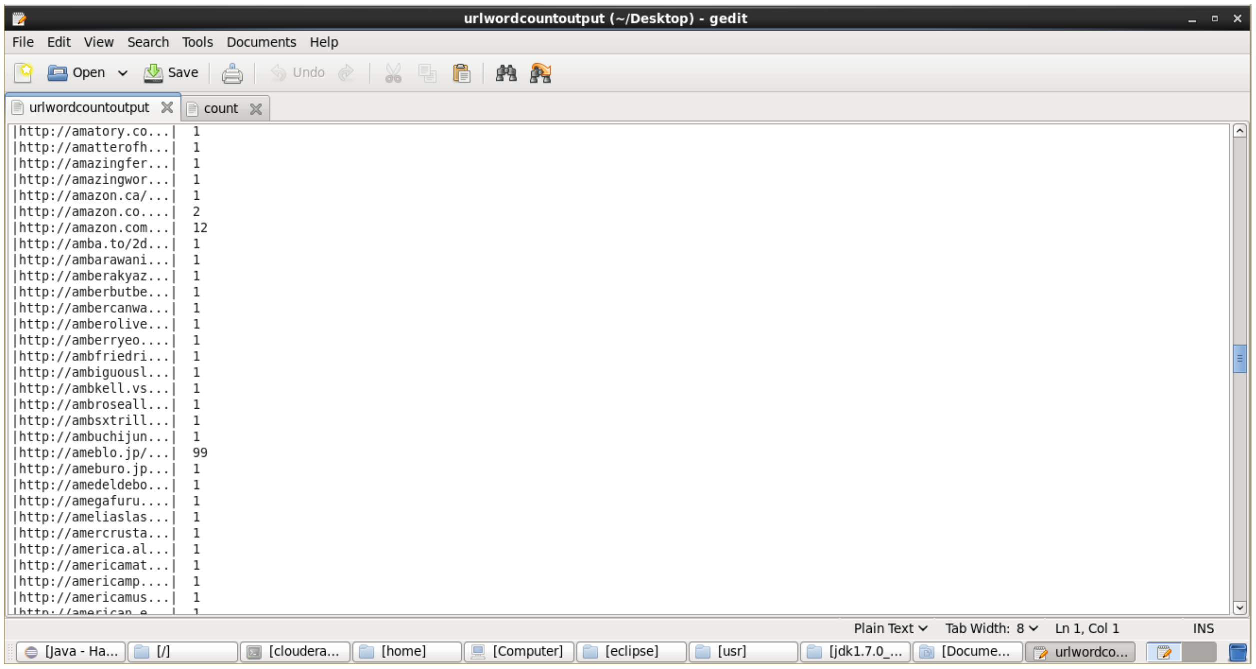
Applications Places System
Sat Sep 30, 6:22 PM
cloudera
urlwordcountcommands (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
urlwordcountcommands
Plain Text Tab Width: 8 Ln 1, Col 1 INS
[Java - HashtagsWord... [/] [cloudera@quickstart:... [home] [cloudera] urlwordcountcomman...

Fig 3: HDFS Commands


Applications Places System
Sat Sep 30, 7:57 PM
cloudera
count (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
urlwordcountoutput
count
Plain Text Tab Width: 8 Ln 1, Col 1 INS

```
|#HomecomingFriday...|  1
|#Homeowners    1
|#Homeschool    2
|#HometownShowcase...|  1
|#Honda 3
|#Honeycrisp    1
|#HoodieMob     1
|#Hookem        1
|#Hooligans     1
|#HoopsMania17  1
|#HopingforaCure20...|  1
|#Hops&amp;Heels        1
|#HoraJudicialUR        1
|#Horizon       1
|#HorizonZeroDawn       3
|#Hormones      1
|#Horoscopes    1
|#Horror        2
|#HorseRacing   1
|#HospitalShip  1
|#HotList       1
|#HotSmart®     2
|#HotTopics     1
|#Hotwife       1
|#Hounded       1
|#HouseofCards  1
|#Houston       2
|#Houston,      1
|#HoustonStrong 1
|#Houthi        1
```

Fig 4: Hashtags word count output


urlwordcountoutput (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
urlwordcountoutput
count
Plain Text Tab Width: 8 Ln 1, Col 1 INS

```
|http://amatory.co...|  1
|http://amatterofh...|  1
|http://amazingfer...|  1
|http://amazingwor...|  1
|http://amazon.ca/...|  1
|http://amazon.co....|  2
|http://amazon.com...|  12
|http://amba.to/2d...|  1
|http://ambarawani...|  1
|http://amberakyaz...|  1
|http://amberbutbe...|  1
|http://ambercanwa...|  1
|http://amberolive...|  1
|http://amberryeo....|  1
|http://ambfriedri...|  1
|http://ambiguousl...|  1
|http://ambkell.vs...|  1
|http://ambroseall...|  1
|http://ambsxtrill...|  1
|http://ambuchijun...|  1
|http://ameblo.jp/...|  99
|http://ameburo.jp...|  1
|http://amedeldebo...|  1
|http://amegafuru....|  1
|http://ameliaslas...|  1
|http://amercrusta...|  1
|http://america.al...|  1
|http://americamat...|  1
|http://americamp....|  1
|http://americamus...|  1
```

Fig 5: URLs word count output