IEMS 5730 Spring 2019 Homework 2

Name: Wenli SONG

Student No.: 1155114524

I declare that the assignment submitted on Elearning system is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/ .

Signed (Student Wenli SONG) Date: 8 March 2019

Name Wenli SONG SID 1155114524

Q1: Pig Setup and Basic Operations

(a) Install Pig under my Hadoop cluster

Download Pig, extract the gzipped file and move it to my home directory.

```
wget http://ftp.cuhk.edu.hk/pub/packages/apache.org/pig/latest/pig-0.17.0.tar.gz
tar -xvzf pig-0.17.0.tar.gz
mv pig-0.17.0 ~/pig
```

Set environment variables by command `vi ~/.bashrc`

```
export PIG_HOME=/home/ubuntu/pig
export PIG_CONF_DIR=$PIG_HOME/conf
export PIG_CLASSPATH=$HADOOP_CONF_DIR
export PATH=$PIG_HOME/bin:$PATH
```

Verify Pig Installation by command `pig -h`, and result are shown as below

```
ubuntu@ip-172-31-13-50:~$ pig -h

Apache Pig version 0.17.0 (r1797386)
compiled Jun 02 2017, 15:41:58

USAGE: Pig [options] [-] : Run interactively in grunt shell.
       Pig [options] -e[xecute] cmd [cmd ...] : Run cmd(s).
       Pig [options] [-f[ile]] file : Run cmds found in file.
  options include:
    -4, -log4jconf - Log4j configuration file, overrides log conf
    -b, -brief - Brief logging (no timestamps)
    -c, -check - Syntax check
    -d, -debug - Debug level, INFO is default
    -e, -execute - Commands to execute (within quotes)
    -f, -file - Path to the script to execute
    -g, -embedded - ScriptEngine classname or keyword for the ScriptEngine
    -h, -help - Display this message. You can specify topic to get help for that topic.
        properties is the only topic currently supported: -h properties.
    -i, -version - Display version information
    -l, -logfile - Path to client side log file; default is current working directory.
    -m, -param_file - Path to the parameter file
    -p, -param - Key value pair of the form param=val
    -r, -dryrun - Produces script with substituted parameters. Script is not executed.
    -t, -optimizer_off - Turn optimizations off. The following values are supported:
            ConstantCalculator - Calculate constants at compile time
            SplitFilter - Split filter conditions
            PushUpFilter - Filter as early as possible
            MergeFilter - Merge filter conditions
            PushDownForeachFlatten - Join or explode as late as possible
            LimitOptimizer - Limit as early as possible
            ColumnMapKeyPrune - Remove unused data
            AddForEach - Add ForEach to remove unneeded columns
            MergeForEach - Merge adjacent ForEach
            GroupByConstParallelSetter - Force parallel 1 for "group all" statement
            PartitionFilterOptimizer - Pushdown partition filter conditions to loader implemen
ting LoadMetaData
            PredicatePushdownOptimizer - Pushdown filter predicates to loader implementing Loa
dPredicatePushDown
            All - Disable all optimizations
        All optimizations listed here are enabled by default. Optimization values are case ins
ensitive.
    -v, -verbose - Print all error messages to screen
    -w, -warning - Turn warning logging on; also turns warning aggregation off
    -x, -exectype - Set execution mode: local|mapreduce|tez, default is mapreduce.
    -F, -stop_on_failure - Aborts execution on the first failed job; default is off
    -M, -no_multiquery - Turn multiquery optimization off; default is on
    -N, -no_fetch - Turn fetch optimization off; default is on
    -P, -propertyFile - Path to property file
    -printCmdDebug - Overrides anything else and prints the actual command used to run Pig, in
cluding
                     any environment variables that are set by the pig command.
19/02/21 08:16:47 INFO pig.Main: Pig script completed in 122 milliseconds (122 ms)
```

Question (b) (c) (d)

Download two datasets.

```
wget http://storage.googleapis.com/books/ngrams/books/googlebooks-eng-all-1gram-20120701-a.gz \
http://storage.googleapis.com/books/ngrams/books/googlebooks-eng-all-1gram-20120701-b.gz
```

Upzip two datasets.

```
gunzip googlebooks-eng-all-1gram-20120701-a.gz \
googlebooks-eng-all-1gram-20120701-b.gz
```

Copy them to hdfs

```
hdfs dfs -copyFromLocal googlebooks-eng-all-1gram-20120701-* .
```

Write Script: `vi 1-q.pig`

1.Merge two files into on table

2.Compute average number of occurrences per year
3.Output the 20 bigrams with the highest average number of occurrences per year along with their corresponding average values sorted in the descending order

```
A = LOAD 'googlebooks-eng-all-1gram-20120701-a' as (bigram:chararray,year:int,match_count:int,volume_c
B = LOAD 'googlebooks-eng-all-1gram-20120701-b' as (bigram:chararray,year:int,match_count:int,volume_c
merged = UNION A, B;
groups = GROUP merged BY bigram;
avgs = FOREACH groups GENERATE group as bigram, AVG(merged.match_count) as avg_count;
sorted = ORDER avgs BY avg_count DESC;
top_20 = LIMIT sorted 20;
STORE top_20 INTO 'output-1' USING PigStorage('\t');
```

Execute script in Grunt shell `pig -x mapreduce`

```
exec q-1.pig
```

Check result `hdfs dfs -cat output-1/*`

```
and     2.593207744E7
and_CONJ        2.5906234451764707E7
a       1.6665890811764706E7
a_DET   1.6645121127058823E7
as      6179734.075294117
be      5629591.52
be_VERB 5621156.232941177
as_ADP  5360443.872941176
by      5294067.04
by_ADP  5272951.997647059
are     4298564.341176471
are_VERB        4298561.303529412
at      3676050.1529411767
at_ADP  3670625.785882353
an      2979272.7411764706
an_DET  2977977.8870588234
but     2471102.4964705883
but_CONJ        2468978.0564705883
all     2189962.722352941
all_DET 2161257.294117647
```

Q2: Hive Setup and Basic Operations

(a) Install Hive under my Hadoop cluster

```
wget http://ftp.cuhk.edu.hk/pub/packages/apache.org/hive/stable-2/apache-hive-2.3.4-bin.tar.gz
tar -xvzf apache-hive-2.3.4-bin.tar.gz
mv apache-hive-2.3.4-bin/ ~/hive

export HIVE_HOME=/home/ubuntu/hive
export HIVE_CONF_DIR=$HIVE_HOME/conf
export PATH=$HIVE_HOME/bin:$PATH

wget https://dev.mysql.com/get/mysql-apt-config_0.8.12-1_all.deb
dpkg -i mysql-apt-config_0.8.12-1_all.deb
sudo apt update
sudo apt install mysql-server
cp /usr/share/java/mysql-connector-java-8.0.15.jar $HIVE_HOME/lib/

cd $HIVE_HOME
schematool -dbType mysql -initSchema
```

Question (b)

Write script `vi q-2.sql`

```
CREATE TABLE BIGRAM (
  bigram STRING,
  year INT,
  match_count INT,
  volume_count INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t';
DESCRIBE BIGRAM;
load data inpath 'googlebooks-eng-all-1gram-20120701-a' into table BIGRAM;
load data inpath 'googlebooks-eng-all-1gram-20120701-b' into table BIGRAM;
INSERT OVERWRITE DIRECTORY 'output-2'
select bigram, SUM(match_count)/COUNT(match_count) avg
from BIGRAM GROUP BY bigram SORT BY avg DESC LIMIT 20;
```

Execute script

```
hive -f q-2.sql
```

Check result `hdfs dfs -cat output-2/*`

```
and 2.593207744E7
and_CONJ  2.5906234451764707E7
a 1.6665890811764706E7
a_DET 1.6645121127058823E7
as  6179734.075294117
be  5629591.52
be_VERB 5621156.232941177
as_ADP  5360443.872941176
by  5294067.04
by_ADP  5272951.997647059
are 4298564.341176471
are_VERB  4298561.303529412
at  3676050.1529411767
at_ADP  3670625.785882353
an  2979272.7411764706
an_DET  2977977.8870588234
but 2471102.4964705883
but_CONJ  2468978.0564705883
all 2189962.722352941
all_DET 2161257.294117647
```

Overall Running Time

Pig	Hive
4 min 12 s	6 min 32s

Q3: Word Counting on a Storm Cluster

(a) Multi-node Storm cluster setup

Download dataset

```
wget --user bigdata --password spring2019bigdata http://mobitec.ie.cuhk.edu.hk/ierg4330Spring2019/home
```

setup Zookeeper

```
wget http://ftp.cuhk.edu.hk/pub/packages/apache.org/zookeeper/stable/zookeeper-3.4.12.tar.gz
tar -xvzf zookeeper-3.4.12.tar.gz
mv zookeeper-3.4.12 ~/zookeeper
```

config `vi ~/zookeeper/conf/zoo.cfg`

```
tickTime=2000
dataDir=/home/ubuntu/zookeeper/data
clientPort=2181
initLimit=5
syncLimit=2
server.1=ip-172-31-13-50:2888:3888
server.2=ip-172-31-15-23:2888:3888
server.3=ip-172-31-1-146:2888:3888
server.4=ip-172-31-0-227:2888:3888
```

```
mkdir /home/ubuntu/zookeeper/data
cd /home/ubuntu/zookeeper
## execute on each cluster node respectively
sh -c "echo '1' > /home/ubuntu/zookeeper/data/myid"
sh -c "echo '2' > /home/ubuntu/zookeeper/data/myid"
sh -c "echo '3' > /home/ubuntu/zookeeper/data/myid"
sh -c "echo '4' > /home/ubuntu/zookeeper/data/myid"

~/zookeeper/bin/zkServer.sh start
~/zookeeper/bin/zkServer.sh status
```

Download Storm

```
wget http://ftp.cuhk.edu.hk/pub/packages/apache.org/storm/apache-storm-1.2.2/apache-storm-1.2.2.tar.gz
tar -xvzf apache-storm-1.2.2.tar.gz
mv apache-storm-1.2.2 ~/storm
```

Configuration

```
## storm
export STORM_HOME=/home/ubuntu/storm
export PATH=$STORM_HOME/bin:$PATH

vi $STORM_HOME/conf/storm.yaml
storm.zookeeper.servers:
    - "ip-172-31-13-50"
    - "ip-172-31-15-23"
    - "ip-172-31-1-146"
    - "ip-172-31-0-227"
nimbus.seeds: ["ip-172-31-13-50"]
storm.local.dir: "/home/ubuntu/storm/storm-local"
supervisor.slots.ports:
    - 6700
    - 6701
    - 6702
    - 6703

storm nimbus &
storm supervisor & (this command excute on other nodes)
storm ui &
```

setup Maven

```
wget http://ftp.cuhk.edu.hk/pub/packages/apache.org/maven/maven-3/3.6.0/binaries/apache-maven-3.6.0-bi
tar -xvzf apache-maven-3.6.0-bin.tar.gz
mv apache-maven-3.6.0 ~/maven
## maven
export MAVEN_HOME=/Users/songwenli/maven
export PATH=$MAVEN_HOME/bin:$PATH
mvn -v
```

use Maven to include Storm as a development dependency

`vi $STORM_HOME/examples/storm-starter/pom.xml`

```
    <dependency>
      <groupId>org.apache.storm</groupId>
      <artifactId>storm-core</artifactId>
      <version>1.2.2</version>
      <scope>provided</scope>
    </dependency>
```

Run the StatefulTopology example to validate the successful installation of your multi-node cluster

```
cd $STORM_HOME/examples/storm-starter
mvn clean install -DskipTests=true
mvn package
storm jar $STORM_HOME/examples/storm-starter/target/storm-starter-*.jar \
storm.starter.StatefulTopology
```

(b) Find Frequent Word: At-most-once model

Code:

WordCountTopology.java

```
package com.mycompany.app;

import com.codahale.metrics.Counter;

import org.apache.storm.Config;
import org.apache.storm.LocalCluster;
import org.apache.storm.StormSubmitter;
import org.apache.storm.topology.TopologyBuilder;
import org.apache.storm.tuple.Fields;

public class WordCountTopology {
    private static final String READ_SPOUT_ID = "read-spout";
        private static final String SPLIT_BOLT_ID = "split-bolt";
        private static final String COUNT_BOLT_ID = "count-bolt";
        private static final String REPORT_BOLT_ID = "report-bolt";
    private static final String TOPOLOGY_NAME = "word-count-topology";

    static Counter emitCounter;
    static Counter ackCounter;

    public static void main(String[] args) throws Exception {
        TopologyBuilder builder = new TopologyBuilder();

        builder.setSpout(READ_SPOUT_ID, new ReadFile(), 1);

        builder.setBolt(SPLIT_BOLT_ID, new SplitSentence(), 8).shuffleGrouping(READ_SPOUT_ID);
        builder.setBolt(COUNT_BOLT_ID, new WordCount(), 12).fieldsGrouping(SPLIT_BOLT_ID, new Fields("
        builder.setBolt(REPORT_BOLT_ID, new Report()).globalGrouping(COUNT_BOLT_ID);

        Config conf = new Config();
        // conf.setDebug(true);
        conf.registerMetricsConsumer(org.apache.storm.metric.LoggingMetricsConsumer.class, 1);
        // LocalCluster cluster = new LocalCluster();
        // cluster.submitTopology(TOPOLOGY_NAME, conf, builder.createTopology());
        StormSubmitter.submitTopology(TOPOLOGY_NAME, conf, builder.createTopology());
        // Thread.sleep(30000);
        // cluster.shutdown();
    }
}
```

ReadFile.java

```
package com.mycompany.app;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Map;

import org.apache.storm.spout.SpoutOutputCollector;
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.OutputFieldsDeclarer;
import org.apache.storm.topology.base.BaseRichSpout;
import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Values;
import org.apache.storm.utils.Utils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ReadFile extends BaseRichSpout {

    private static final long serialVersionUID = 1L;

    private static final Logger LOG = LoggerFactory.getLogger(ReadFile.class);

    SpoutOutputCollector collector;
    private FileReader fileReader;
    BufferedReader br;
    private String fileName = "/home/ubuntu/StormData.txt";
    private boolean completed = false;

    public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {
        try {
            fileReader = new FileReader(fileName);
            br = new BufferedReader(fileReader);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        this.collector = collector;
        WordCountTopology.emitCounter = context.registerCounter("emitCounter");
        WordCountTopology.ackCounter = context.registerCounter("ackCounter");
    }

    public void nextTuple() {
        if (completed) {
            Utils.sleep(300000);
            return;
        }

        String line;
        try {
            if ((line = br.readLine()) != null) {
                LOG.debug("Emitting tuple: {}", line);
                this.collector.emit(new Values(line));
                WordCountTopology.emitCounter.inc();
```

```
            } else {
                System.out.print("complete!--------------------");
                completed = true;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void ack(Object id) {
    }

    public void fail(Object id) {
    }

    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("sentence"));
    }
}
```

SplitSentence.java

```
package com.mycompany.app;

import java.util.Map;

import org.apache.storm.task.OutputCollector;
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.OutputFieldsDeclarer;
import org.apache.storm.topology.base.BaseRichBolt;
import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Tuple;
import org.apache.storm.tuple.Values;

public class SplitSentence extends BaseRichBolt {
    private static final long serialVersionUID = 1L;
    private OutputCollector collector;

    public void prepare(Map stormConf, TopologyContext context, OutputCollector collector) {
        this.collector = collector;
    }

    public void execute(Tuple input) {
        WordCountTopology.ackCounter.inc();
        String line = input.getStringByField("sentence");
        String[] words = line.split(" ");
        for (String word : words) {
            if (word.trim().isEmpty())
                continue;
            this.collector.emit(new Values(word.toLowerCase()));
            WordCountTopology.emitCounter.inc();
        }
    }

    public void declareOutputFields(OutputFieldsDeclarer declarer) {
        declarer.declare(new Fields("word"));
    }
}
```

Report.java

```
package com.mycompany.app;

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

import org.apache.storm.task.OutputCollector;
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.OutputFieldsDeclarer;
import org.apache.storm.topology.base.BaseRichBolt;
import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Tuple;
import org.apache.storm.tuple.Values;

public class Report extends BaseRichBolt {
    private static final long serialVersionUID = 1L;
    private HashMap<String, Integer> counts = null;
    private String outputFile = "/home/ubuntu/output-word-count.txt";
    private String outputCountFile = "/home/ubuntu/count.txt";
    private OutputCollector collector;

        public void prepare(Map config, TopologyContext context,
                        OutputCollector collector) {
        this.counts = new HashMap<String, Integer>();
        this.collector = collector;
        }

    public void execute(Tuple tuple) {
        WordCountTopology.ackCounter.inc();
                String word = tuple.getStringByField("word");
                Integer count = tuple.getIntegerByField("count");
        this.counts.put(word, count);

        System.out.println(word + "\t" + this.counts.get(word));
        System.out.println("Emit count:\t" + WordCountTopology.emitCounter.getCount());
        System.out.println("Ack count:\t" + WordCountTopology.ackCounter.getCount());
        }

        public void declareOutputFields(OutputFieldsDeclarer declarer) {
                // declarer.declare(new Fields("word", "count"));
        }

    public void cleanup() {
        List<String> keys = new ArrayList<String>();
        keys.addAll(this.counts.keySet());
        try {
            PrintWriter out = new PrintWriter(outputFile);
            for (String key : keys) {
                out.println(key + "\t" + this.counts.get(key));
            }
```

```
            out.close();
            out = new PrintWriter(outputCountFile);
            out.println("Emit count:");
            Long emitCount = WordCountTopology.emitCounter.getCount();
            out.println(emitCount);
            out.println("Ack count:");
            Long ackCount = WordCountTopology.ackCounter.getCount();
            out.println(ackCount);
            out.println("Fail count:");
            out.println(emitCount - ackCount);
            out.close();
        } catch (Exception e) {

        }
    }
}
```

Execute and check result

```
mvn -B archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DgroupId=com.mycompany.app \
  -DartifactId=wordcount

mvn clean install -DskipTests=true
mvn package
storm jar ./target/wordcount-1.0-SNAPSHOT.jar com.mycompany.app.WordCountTopology

sort -rVk 2 ~/output-word-count.txt -o ~/output-word-count.txt
head -n 10 ~/output-word-count.txt
```

Top 10

```
the     86178
and     71392
of      49250
to      28035
i       23545
in      20895
that    20800
a       19126
he      14970
for     14923
```

Count

Emit count	Ack count	Fail count
3224720	3224720	0

Q4: Discover Twitter Trending Hashtags using Storm

(a) Find popular hash tags using the multi-node Storm cluster

NOTE: This section is not fully completed.

Start project

```
mvn -B archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DgroupId=com.mycompany.app \
  -DartifactId=twitter
```

Configure pom.xml

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-assembly-plugin</artifactId>
      <configuration>
        <archive>
          <manifest>
            <mainClass>com.mycompany.app.TwitterHashtagStorm</mainClass>
          </manifest>
        </archive>
        <descriptorRefs>
          <descriptorRef>jar-with-dependencies</descriptorRef>
        </descriptorRefs>
      </configuration>
      <executions>
        <execution>
          <id>make-assembly</id> <!-- this is used for inheritance merges -->
          <phase>package</phase> <!-- bind to the packaging phase -->
          <goals>
            <goal>single</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.storm</groupId>
    <artifactId>storm-core</artifactId>
    <version>1.2.2</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.twitter4j</groupId>
    <artifactId>twitter4j-stream</artifactId>
    <version>4.0.7</version>
  </dependency>
    <dependency>
    <groupId>org.twitter4j</groupId>
    <artifactId>twitter4j-async</artifactId>
    <version>4.0.7</version>
  </dependency>
  <dependency>
    <groupId>org.twitter4j</groupId>
    <artifactId>twitter4j-core</artifactId>
    <version>4.0.7</version>
```

```
    </dependency>
  </dependencies>
```

Code (**NOTE**: Some code are from network)
TwitterHashtagStorm.java

```
package com.mycompany.app;

import org.apache.storm.Config;
import org.apache.storm.LocalCluster;
import org.apache.storm.StormSubmitter;
import org.apache.storm.topology.TopologyBuilder;
import org.apache.storm.tuple.Fields;

public class TwitterHashtagStorm {

        public static void main(String[] args) throws Exception {
                String consumerKey = "QzjpEe7Mk4c8W9mmrppPfEazI";
                String consumerSecret = "QBcRBN8HJIKT5nm5ngWD9rl4s3FPV2Wpoava0tX1MjYTEaLAC9";
                String accessToken = "906367537159573504-qH0zLIw0Av7ig6NLL75zwW6Qk42KIkE";
                String accessTokenSecret = "WlOYHBBz9ce99fmyi9j78SAH8MzZ9g0UYarWm18wW5I5v";

                String[] keyWords = { "trump", "Trump", "TRUMP" };

                Config conf = new Config();
                // config.setDebug(true);

                TopologyBuilder builder = new TopologyBuilder();
                builder.setSpout("twitter-spout",new TwitterSampleSpout(consumerKey, consumerSecret, a

                builder.setBolt("twitter-hashtag-reader-bolt", new HashtagReaderBolt(),9).shuffleGroup

                builder.setBolt("twitter-hashtag-counter-bolt", new HashtagCounterBolt()).fieldsGroupi

                // LocalCluster cluster = new LocalCluster();
                StormSubmitter.submitTopology("TwitterHashtagStorm", conf,builder.createTopology());
                // Thread.sleep(30000);
                // cluster.shutdown();
        }
}
```

TwitterSampleSpout.java

```
package com.mycompany.app;

import java.util.Map;
import java.util.concurrent.LinkedBlockingQueue;

import twitter4j.*;
import twitter4j.conf.ConfigurationBuilder;

import org.apache.storm.Config;
import org.apache.storm.spout.SpoutOutputCollector;
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.OutputFieldsDeclarer;
import org.apache.storm.topology.base.BaseRichSpout;
import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Values;
import org.apache.storm.utils.Utils;

@SuppressWarnings("serial")
public class TwitterSampleSpout extends BaseRichSpout {

        private SpoutOutputCollector collector;
        private LinkedBlockingQueue<Status> queue = null;
        private TwitterStream twitterStream;
        private String consumerKey;
        private String consumerSecret;
        private String accessToken;
        private String accessTokenSecret;
        private String[] keyWords;

        public TwitterSampleSpout() {
        }

        public TwitterSampleSpout(String consumerKey, String consumerSecret,String accessToken, String
                this.consumerKey = consumerKey;
                this.consumerSecret = consumerSecret;
                this.accessToken = accessToken;
                this.accessTokenSecret = accessTokenSecret;
                this.keyWords = keyWords;
        }

        public void open(Map conf, TopologyContext context, SpoutOutputCollector collector) {
                queue = new LinkedBlockingQueue<Status>(1000);
                this.collector = collector;

                ConfigurationBuilder cb = new ConfigurationBuilder();

                cb.setDebugEnabled(true).setOAuthConsumerKey(consumerKey).setOAuthConsumerSecret(consu

                this.twitterStream = new TwitterStreamFactory(cb.build()).getInstance();
                StatusListener listener = new StatusListener(){

                        public void onException(Exception ex) {

                        }
```

```
                public void onStatus(Status status) {
                        queue.offer(status);
                }

                public void onDeletionNotice(StatusDeletionNotice statusDeletionNotice) {

                }

                public void onTrackLimitationNotice(int numberOfLimitedStatuses) {

                }

                public void onScrubGeo(long userId, long upToStatusId) {

                }

                public void onStallWarning(StallWarning warning) {

                }

        };
        this.twitterStream.addListener(listener);

        if (keyWords.length == 0) {
                this.twitterStream.sample();
        }else {
                FilterQuery query = new FilterQuery().track(keyWords);
                this.twitterStream.filter(query);
        }
}

public void nextTuple() {
        Status ret = queue.poll();
        if (ret == null) {
                Utils.sleep(50);
        } else {
                this.collector.emit(new Values(ret));
        }
}

public void close() {
        this.twitterStream.shutdown();
}

public Map<String, Object> getComponentConfiguration() {
        Config ret = new Config();
        ret.setMaxTaskParallelism(1);
        return ret;
}

public void ack(Object id) {
}

public void fail(Object id) {
```

```
        }

        public void declareOutputFields(OutputFieldsDeclarer declarer) {
                declarer.declare(new Fields("tweet"));
        }

}
```

HashtagReaderBolt.java

```
package com.mycompany.app;

import java.util.HashMap;
import java.util.Map;

import twitter4j.*;
import twitter4j.conf.*;

import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Values;

import org.apache.storm.task.OutputCollector;
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.IRichBolt;
import org.apache.storm.topology.OutputFieldsDeclarer;
import org.apache.storm.tuple.Tuple;

public class HashtagReaderBolt implements IRichBolt {
   private static final long serialVersionUID = 1L;
   private OutputCollector collector;

   public void prepare(Map conf, TopologyContext context, OutputCollector collector) {
      this.collector = collector;
   }

   public void execute(Tuple tuple) {
      Status tweet = (Status) tuple.getValueByField("tweet");
      for(HashtagEntity hashtage : tweet.getHashtagEntities()) {
         // System.out.println("Hashtag: " + hashtage.getText());
         this.collector.emit(new Values(hashtage.getText()));
      }
   }

   public void cleanup() {}

   public void declareOutputFields(OutputFieldsDeclarer declarer) {
      declarer.declare(new Fields("hashtag"));
   }

   public Map<String, Object> getComponentConfiguration() {
      return null;
   }

}
```

HashtagCounterBolt.java

```
package com.mycompany.app;

import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Values;
import org.apache.storm.Config;
import org.apache.storm.Constants;
import org.apache.storm.task.OutputCollector;
import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.IRichBolt;
import org.apache.storm.topology.OutputFieldsDeclarer;
import org.apache.storm.tuple.Tuple;

public class HashtagCounterBolt implements IRichBolt {
   private static final long serialVersionUID = 1L;
   Map<String, Integer> counterMap;
   private OutputCollector collector;

   public void prepare(Map conf, TopologyContext context, OutputCollector collector) {
      this.counterMap = new HashMap<String, Integer>();
      this.collector = collector;
   }

   public void execute(Tuple tuple) {
      try {
         if (tuple.getSourceComponent().equals(Constants.SYSTEM_COMPONENT_ID)
               && tuple.getSourceStreamId().equals(Constants.SYSTEM_TICK_STREAM_ID)) {
            List<String> keys = new ArrayList<String>();
            keys.addAll(this.counterMap.keySet());
            try {
                PrintWriter out = new PrintWriter("/home/ubuntu/twitter_stream.txt");
                for (String key : keys) {
                    out.println(key + "\t" + this.counterMap.get(key));
                }
                out.close();
            } catch (Exception e) {

            }
         }

         String key = tuple.getString(0);
         if(!counterMap.containsKey(key)){
            counterMap.put(key, 1);
         } else {
            Integer c = counterMap.get(key) + 1;
            counterMap.put(key, c);
         }
         collector.ack(tuple);
```

```
      } catch (Exception e) {
         //TODO: handle exception
      }
   }

   public void cleanup() {
      for(Map.Entry<String, Integer> entry:counterMap.entrySet()){
         System.out.println("Result: " + entry.getKey()+" : " + entry.getValue());
      }
   }

   public void declareOutputFields(OutputFieldsDeclarer declarer) {
      declarer.declare(new Fields("hashtag"));
   }

   public Map<String, Object> getComponentConfiguration() {
      Config conf = new Config();
      conf.put(conf.TOPOLOGY_TICK_TUPLE_FREQ_SECS, 600);
      return conf;
   }

}
```

Run

```
mvn clean install -DskipTests=true
mvn package
storm jar ./target/twitter-1.0-SNAPSHOT-jar-with-dependencies.jar com.mycompany.app.TwitterHashtagStor
sort -rVk 2 ~/twitter_stream.txt -o ~/twitter_stream.txt
```

Result, Top 10 frequent word. (After running 1 hour 30 mins)

```
Trump   1471
ThursdayThoughts        1116
MAGA    682
RussianCollusion        458
trump   232
Socialust       188
Resist  167
KAG     162
Trump2020       158
WWG1WGA 146
```