MIDS W205

Exercise #	2	Title	Introduction to the Elements of a Streaming application.
Related Module(s)	8,9	Goal	Implement an end to end streaming app.
Last Updated	10/23/15	Expected duration	15-25 hours.

Introduction

Streaming applications may seem complex but understand how they operate will be critical for a data scientist. In this Exercise we will explore a streaming application analyzing Twitter data. In order to allow you to explore a more complex implementation in a limited amount of time you will be using an existing code base and add codes to it. You will use Streamparse as used in Lab 9 with a given topology. The application reads the stream of tweets from Twitter streaming API, parses them, counts the number of each words in the stream of tweets, and write the final results back to a Postgres table.

Scope and Goal

In this Exercise, you will capture and process live streaming Twitter data covering the following features:

- Data Streaming
- Capturing the live data
- Processing to get insights
- Store the final processing results to a relational database management system

Technologies used in this exercise:

Apache Storm, Amazon EC2, python, Twitter API, Streamparse, Postgres, and psycopg

Instructions, Resources, and Prerequisites

In the following table you will find references and resources related to programs and components used or you need to use.

Resource	What
http://storm.apache.org/documentation.html	Apache Storm Documentation
https://streamparse.readthedocs.org/en/latest/api.html	Stream Parse Documentation
https://dev.twitter.com/streaming/overview	Twitter Stream API
http://docs.tweepy.org/en/v3.4.0/ http://docs.tweepy.org/en/v3.4.0/streaming_how_to.html	Tweepy Documentation
http://initd.org/psycopg/	psycopg

In order to get exposed to a more substantial example of stream processing we will ask you to understand and deploy an existing Streamparse project and enhance it with more features rather than developing an application from scratch.

Use Case:

Catching and analyzing live twitter data around your business interest area can give you a deeper understanding of current social trends and demands. Older data can give you information on the mainstream trends over a certain period of time, but live data can give you exact and accurate insights real time. For example, say there is manager who manages live TV ads during a popular TV program, which is broadcasted every week. Basing on the Twitter trends at the time of the show live, the manager can decide which ad would be more contextual and engage viewers even more. This will ensure the viewer's interest in not only in the show but in the ads as well.

So, in this exercise, you will exactly capture live social tweets to see people's live interests, process it real time and actually summarize or aggregate to get insights. Figure 1 shows the overall architecture of the application as well as the topology of its storm component that you need to develop. The application reads the live stream of tweets from twitter in **Tweet-spout** using Tweepy library. The **Parse-tweet-bolt** parses the tweets, extracts the words from each parse tweets and emits the results**. Count-bolt** counts the number of each word in the received tuples and updates the counts associated with each words in the **Tweetwordcount** table inside the **Tcount** database.

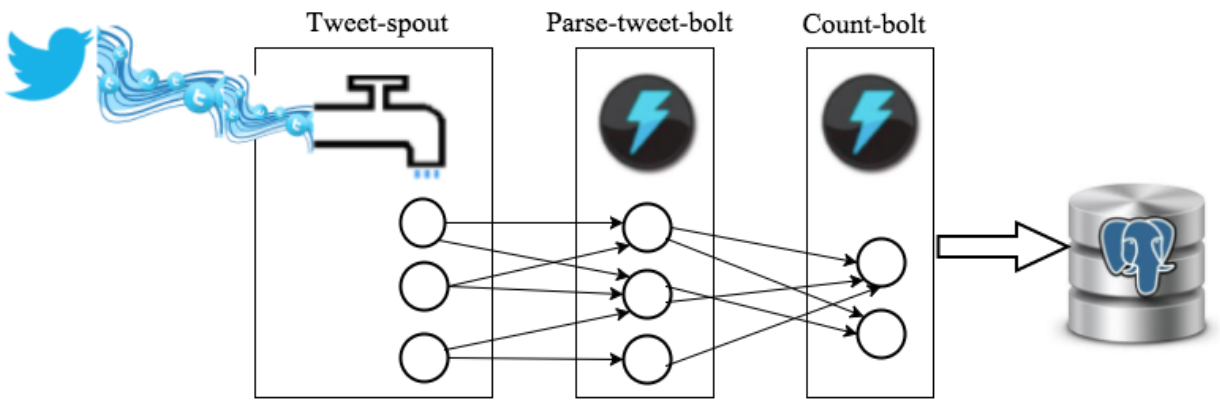

Figure 1: Application Topology

Overall Guideline - for all steps

In this section we provide the overall guideline of implementing the real time system using live twitter data as shown in Figure 1. You need to follow the below detail guideline for each of the steps of your implementation. You would use **ucbw205_complete_plus_postgres_PY2.7** AMI

for creating your own EC2 server for this exercise. You use your github account to store your scripts, data, etc.

Step-1. Environment and Tool Setup

1. Clone the Github Repository for this exercise from git@github.com:UC-Berkeley-I-School/w205-labs-exercises.git (If you want to just clone the exercise_2 directory, see appendix).
2. Create an EC2 instance using **ucbw205_complete_plus_postgres_PY2.7** and install Streamparse on it. You can use the instructions from Lab 9 to install the Streamparse. You may want to save this AMI once you configured it for the first time to save time in future re-launches.
3. Create a project called **Tweetwordcount** in Streamparse.
4. Copy the files from the **tweetwordcount** directory in your cloned repository and paste them in the corresponding folders in the new **Tweetwordcount** project. The description of the files in the code base is provided in Table 1.
5. Modify the **tweetwordcount.clj** to match the topology in Figure 1.

Note 1: You may not want to keep your EC2 server live all the time as you will run out of credit that way. So, you could save your work in github as you progress and when you make your sever alive, you can re -pull the code and use.

Name of the program	Location	Description
Tweets.py	exercise_2/tweetwordcount/src/spouts/	Tweet-spout
Parse.py	exercise_2/tweetwordcount/src/bolts/	Parse-tweet-bolt
Wordcount.py	exercise_2/tweetwordcount/src/bolts/	Count-bolt
Twittercredentials.py	exercise_2/	Twitter App Keys
hello-stream-twitter.py	exercise_2/	Sample twitter Stream program
tweetwordcount.clj	/exercise_2/tweetwordcount/topologies/	Topology for the program
psycopg-sample.py	exercise_2/	Sample codes on how to use psycopg
Finalresults.py	exercise_2/	Showing the results in the Postgres database

Table 1: Description of the files in the code base

Step-2. Twitter application setup

Once you have the topology configured with codes in the cloned repository, you need to modify the spout program (i.e Tweets.py) to pull the tweets from twitter streaming API. In order to get the tweets, you need to set up a twitter application and get access keys for pulling the tweets out of the twitter streaming API. You also need to install Tweepy for working

The following instructions will step you through the process of acquiring data from Twitter. The first thing that you need to do is to install Tweepy, which is a python library for accessing the Twitter API.

2.1 Install Tweepy

The easiest way to install Tweepy is by using pip:

```
$pip install tweepy
```

You may also use Git to clone the repository directly from Github and install it manually:

```
$git clone https://github.com/tweepy/tweepy.git
$cd tweepy
$python setup.py install
```

The next step is to get access to twitter data**.** Twitter data can be accessed over the Web by creating an application on their site and then using the access keys they provide for the application in your program.

2.2 Create an Application

Note: You will need to have a Twitter account to create an application.
To create an application:

1. Login to Twitter (https://www.twitter.com/).
2. Visit https://apps.twitter.com and click on "Create New App".
3. Fill in the application name, description, and Website. The name will be listed in your application list when you return this Website.
4. Agree to the terms and agreements and click on "Create your Twitter Application"

Once you have successfully created an application, it should take you to the newly created application. Here you must create access keys for subsequent operations by your application. To do so, use the following procedure:

1. Click on the "Keys and Access Tokens" tab.
2. Click on "Create my Access Token" near the bottom of the page.

The response should be relatively immediate. Now you have four things:

1. A consumer key that identifies your application.
2. A consumer secret that acts as a "password" for your application.
3. An access token that identifies your authorized access.
4. An access token secret that acts as a "password" for that authorized access.

At any point, you can revoke the access key or regenerated any of these values. To completely disable the application, you must delete the application. This does remove the consumer key, secret, and access tokens from Twitter's system and any program using them will immediately stop working.

2.3 Test your Application

In the code base that you cloned, hello-stream-twitter.py is a sample application that pulls tweets from the twitter streaming API. This program uses Tweepy to work with the streaming API. Use the hello-stream-twitter.py program to test your application. Change the code in Twittercredentials.py and insert your consumer key, consumer secret, access token, and access token secret. You should then be able to just run the program and get tweets:

```
$python hello-stream-twitter.py
```

The application will just return a count for number of tweets received. If you want to print the tweets modify the code accordingly.

Step-3. Application deployment

Now you have a streamparse project and a twitter application, your task is to write codes to connect necessary pieces together as to create a full stream tweetword count processing application as depicted in Figure 1. Your application has the following pieces:

1. A spout connected to twitter streaming API that pulls the tweets from twitter stream and emits them to the parse bolt. You need to modify the necessary codes to complete this part.
2. A tweet word count bolt that counts the number of words in the received tweets and updates the total counts in a corresponding table inside a Postgres database. For this part, you need to create a Postgres DB called **Tcount** and a table called **Tweetwordcount** table inside **Tcount**. You also need to modify the code in the **Wordcount.py** to updated the word counts in **Tweetwordcount** for each word in the tweet stream. To interact with Postgres, you can use **psycopg.** You can find a sample codes on how to use **psycopg** in **psycopg-sample.py**

Step-4. Reporting Scripts

In the previous step, you have developed a full twitter stream processing application. In this step, your task is to develop two simple reporting scripts that query the database and report on current status of data statistics. You need to develop two scripts:

1. `finalresults.py`

   The script gets as argument a `word` and when run outputs an integer that represents the total number of `word` occurrences in the stream. For example:

   ```
   $ python finalresults.py hello
   $ Total number of occurences of "hello": 10
   ```

   When the script is run without specifying an argument, it outputs all pairs of words found in the stream along with their total count of occurrences, sorted alphabetically in an ascending order, one word per line. For example:

   ```
   $ python finalresults.py
   ```

```
$ (<word1>, 2), (<word2>, 8), (<word3>, 6), (<word4>, 1), …
```

2. `histogram.py`
   The script gets as argument an integer k and when run outputs a histogram of the k words with the largest total number of occurrences in the stream. For example:

```
$ python histogram.py 3
$     <word2>: 8
      <word3>: 6
      <word1>: 2
```

   How your script handles ties? Make an assumption and discuss in your report.

Submissions and Assessment Criteria:

Submission Items:

1. Complete documentation of your system
   a. Folder name
   b. File name
   c. Description/Purpose
   d. Dependency on other files
2. Complete and fully functional spout and bolts programs based on the description above
3. End to End Run: Process Run - Screen shots (At least 3 of your choice)
4. **FinalResults.py** that
   a. Takes a word as an argument and shows the total counts of the word
   b. Has an option of printing all the final results of your processing
5. Histogram of top 30 words in your stream

Appendix

In this section we introduce some additional instructions. Some of these are from other labs and you may be able to re-use installations from those.

Installing Streamparse

First make sure you are using python version 2.7.x. If you do not, update to 2.7 by following the instructions in the appendix section.

Install `ez_setup`.

```
$sudo curl -o ez_setup.py https://bootstrap.pypa.io/ez_setup.py
```

```
$sudo python ez_setup.py
```

Use `ez_install` to install `pip`, which is a package manager for Python software.

```
$sudo /usr/bin/easy_install-2.7 pip
```

Then use `pip` to install `virtualenv`. `virtualenv` is a tool to create and manage dependencies for distinct Python environments. Streamparse uses `virtualenv` to manage all dependencies for individual Python Storm projects.

```
$sudo pip install virtualenv
```

Streamparse requires the build tool `lein` to resolve dependencies, so now you will install `lein`.

Note: In the video the installer fails to save to `/usr/bin`, so we have to move the `lein` file there. If the command succeeds you will not need to use the `mv lein /usr/bin` command.

```
$wget --directory-prefix=/usr/bin/
https://raw.githubusercontent.com/technomancy/leiningen/stable/bin/lein
```

If you check the permissions on the `lein` file, you will see it is not executable. This means the shell and operating system will not allow you to run it as a command.

```
$ls -l /usr/bin/lein
-rw-r--r-- 1 root root 12713 Oct 25 17:01 /usr/bin/lein
```

Use the following `chmod` command to turn on the executable permission for all users.

```
$ chmod a+x /usr/bin/lein
```

Check that it looks like what you expected.

```
$ls -l /usr/bin/lein
-rwxr-xr-x 1 root root 12713 Oct 25 17:01 /usr/bin/lein
```

The first time you run `lein`, it will install itself.

```
$sudo /usr/bin/lein

$lein version
WARNING: You're currently running as root; probably by accident.
```

```
Press control-C to abort or Enter to continue as root.
Set LEIN_ROOT to disable this warning.

Leiningen 2.5.3 on Java 1.7.0_79 Java HotSpot(TM) 64-Bit Server
```

Install streamparse.

```
$pip install streamparse
```

Now you have streamparse installed. It will greatly simplify the creation of Python Storm projects and help you get a simple example up and running quickly. The following command creates an installation of the `wordcount` example.

```
$sparse quickstart wordcount
```

After watching the video and understanding the structure of topology definitions and the actual spout and bolt, run the word-count example use the following commands:

```
$cd wordcount
$sparse run
```

Python version 2.7

If you run into issues running Streamparse, check your python version. You need to use python 2.7. Follow the instructions below, and you will need to rerun the streamparse installation.

Install the required version of Python.

```
$sudo yum install python27-devel –y
```

You can see that the Python in your execution path ($PATH) is still 2.6.X by checking the version again.

```
$python --version
Python 2.6.6
```

Rename the current version to reflect its correct version.

```
$mv /usr/bin/python /usr/bin/python266
```

Create a symbolic link from the file in the path to the version you want to execute.

```
$ln -s /usr/bin/python2.7 /usr/bin/python
```

Check that the link indeed refers to the intended version of Python.

```
$/usr/bin/python --version
Python 2.7.3
```

Check that the shell picks up the version of Python you intended.

```
$python --version
Python 2.7.3
```

Cloning a subdirectory

Git clones complete repos. If you only want to clone the `exercise_2` directory you need to use a “sparse checkout”. Follow the below instructions. They allow you to clone only the `exercise_2` sub directory.

```
$mkdir ex2
$cd ex2/
$git init
$git remote add -f origin git@github.com:UC-Berkeley-I-pwdSchool/w205-labs-exercises.git
$git config core.sparseCheckout true
$echo "exercise_2" >> .git/info/sparse-checkout
$git pull origin master
```