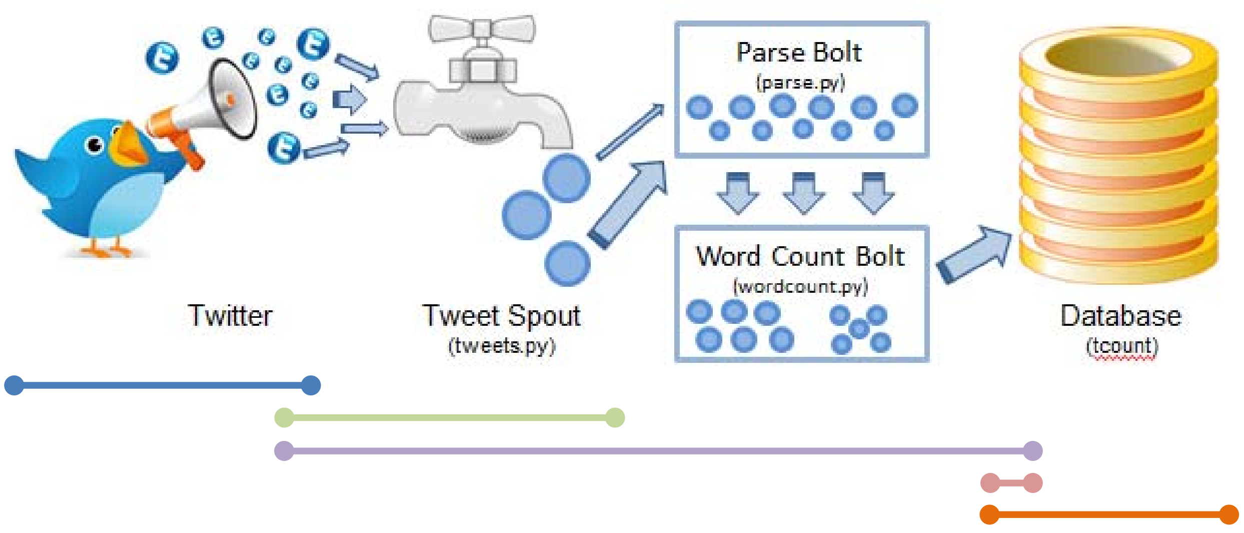
Timothy Alt
w205 – Storing and Retrieving Data, Exercise 2
Design/Architecture Documentation

Application Introduction and Overview

In this Exercise 2, we are tasked with developing a start-to-finish application that reads a stream of tweets from Twitter, parses the tweets into individual words, performs a word count, and stores the results in a database. Using Amazon Web Services – Elastic Compute Cloud (EC2), we will use Apache Storm, Tweepy, and Streamparse to acquire and navigate the tweet stream, the Twitter API to interface with Twitter, and PsycoPG/Postgres to populate and store the resulting data. Python programming will be used throughout and independently for data processing and analysis.

Architecture Overview

Figure 1 below is used to visualize the flow of information through the system for this exercise, as well as illustrate which system and technology is used during each step of the flow. A detailed explanation of the architecture and processing is in the next section.



🟦	🟩	🟪	🟥	🟧
Twitter	Twitter API	Apache Storm	PsycoPG	Postgres
	Tweepy	Streamparse		
	Python			
	Amazon Web Services – EC2			

Figure 1: Application Architecture and Topology

The Setup
To begin, we use the provided AMI (UCB MIDS w205 EX2-FULL) to create an EC2 instance, create a data directory, and mount a drive to that directory for our analysis. The startup scripts for Postgres are downloaded and installed, as well as PsycoPG and Tweepy. Also needed beforehand is a Twitter account, which includes the necessary information for setting up the Twitter API.

Next, a Streamparse project named extweetwordcount is created. Since we modified several of the default files for our project, our new files will need to be moved into various specific folders for our application to run correctly. The file structure is pictured below in Figure 2. Within the extweetcount folder, the changes were:

- topologies folder (the clojure file, deleted and replaced with extweetwordcount.clj),
- src/spouts folder (the python file, deleted and replaced with tweets.py), and
- src/bolts folder (the python file, deleted and replaced with parse.py and wordcount.py).

There are other files that were created as part of the Streamparse setup. Please leave these in place. Only delete/replace the files listed above and diagramed below in Figure 2.

Additional Python files will also need to be moved to the extweetwordcount folder that are not included in the Streamparse setup. These include:

- twittercredentials.py and hello-stream-twitter.py, to test our Twitter API,
- psycoph-sample.py, to create and test our Postgres database and table, and
- finalresults.py and histogram.py, to perform our final analysis.

Once the files are in place, run psycopg-sample.py to create and test the Postgres database and table. Also run hello-stream-twitter.py to make sure the Twitter API and credentials work as expected. Only when these are executed successfully can we begin streaming data and collecting information. Specifically, psycopg.py must be run in order for the word count bolt (wordcount.py), as well as our Python programs for analysis, to be successful.

Figure 2: File Structure and Directory

exercise_2/extweetwordcount/

- o src
  - o bolts
    - parse.py
    - wordcount.py
  - o spouts
    - tweets.py
- o topologies
  - extweetwordcount.clj
- Twittercredentials.py
- Hello-stream-twitter.py
- Psycopg.py
- Finalresults.py
- Histogram.py

The Application and Analysis
Once the setup is complete, we have all the pieces we need to run the application. The Twitter API and spout are configured to gather tweets. The bolts will parse the tweets and perform word counts, as well as input the data into a database table. Running Streamparse in the extweetwordcount folder will begin the streaming and collection of data. After a minute or so of data collection, the streaming can be broken by Control-C.

At this point, the Python programs finalresults.py and histogram.py can be run. "Finalresults.py" as written will return all words in the database alphabetically with each of their word counts. If one word is entered after the filename, only that word and its respective count will be returned. "Histogram.py" must be run with two integers passed as arguments (python histogram.py 2, 10). The results will be all words that have counts between those two integers, inclusively. Figure 3 below shows a histogram of the top 20 words from the analysis performed.

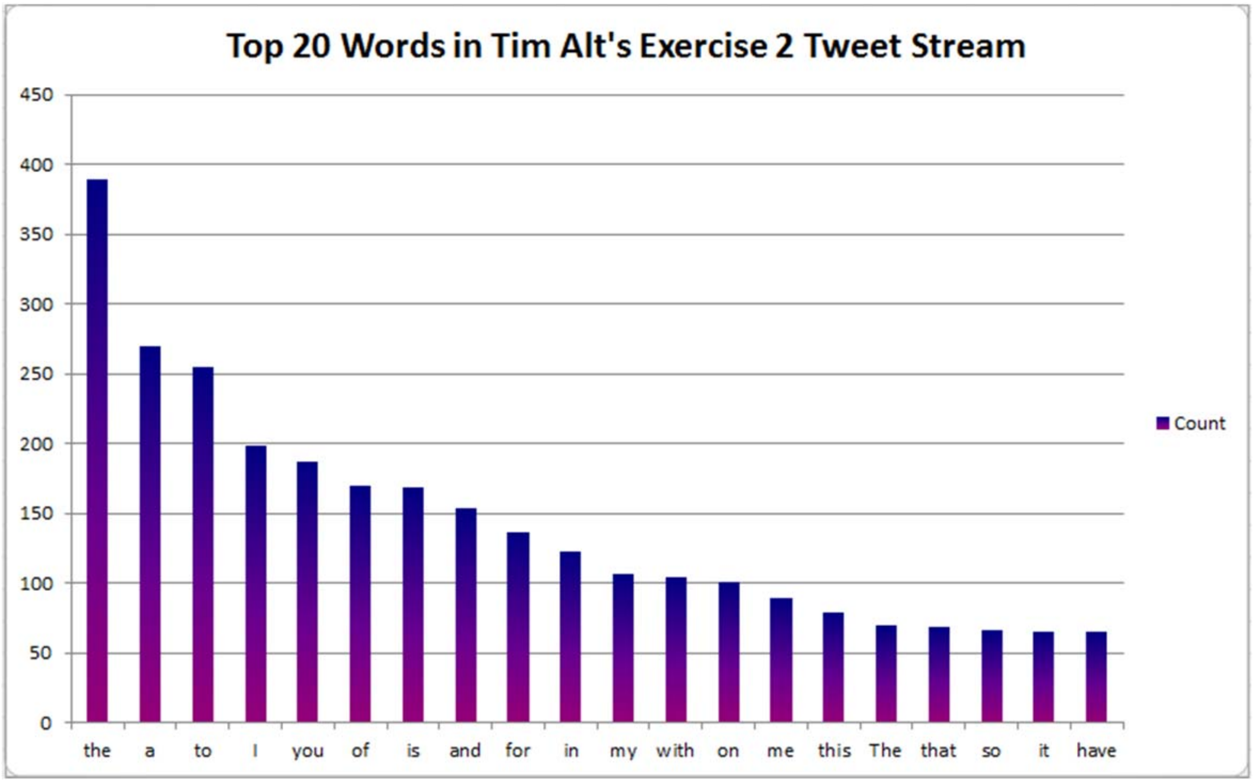


Figure 3: Top 20 Words from Tim's Analysis