Achitecture of extweetwordcount

**Dependencies**

Apache Storm

Amazon EC2

Python

Twitter API

Streamparse

Postgres

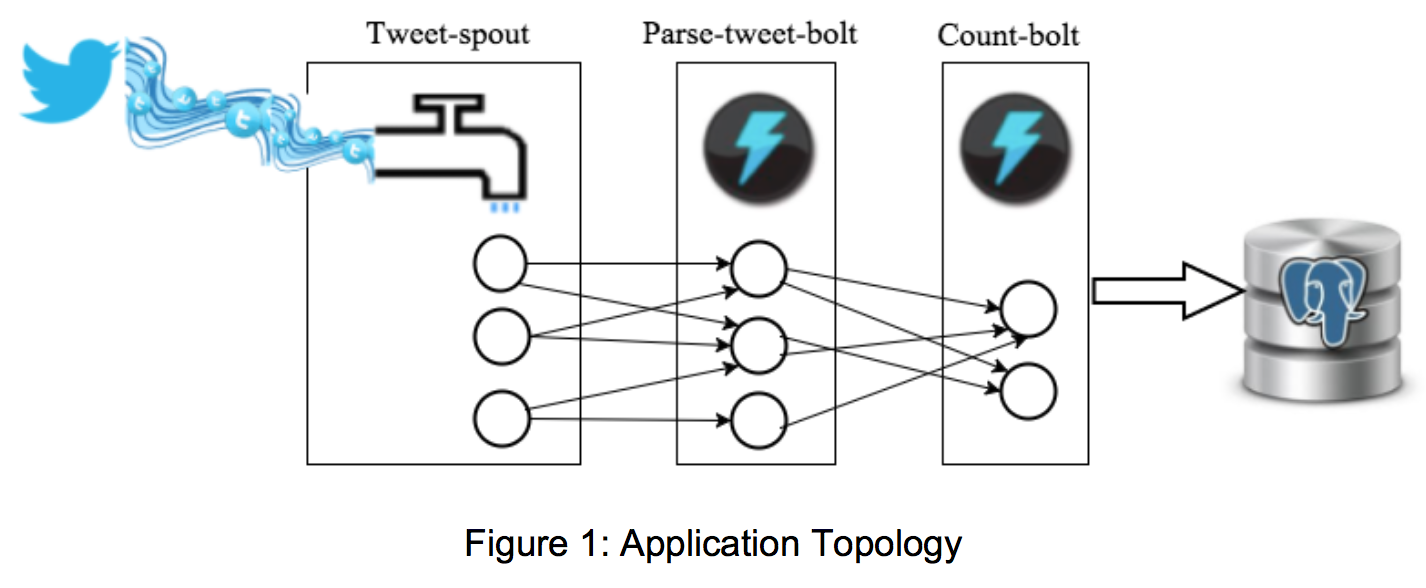
PsycoPG

Tweepy

**Overview**

Streaming applications may seem complex, but understanding how they operate is critical for a data scientist. In this exercise, we will explore a streaming application that analyzes Twitter data. In order to explore a complex implementation in a short period of time, you will develop your application using an existing codebase. You will use Streamparse, as seen in Lab 6, with a given topology. The application reads the stream of tweets from the Twitter streaming API, parses them, counts the occurrences of each word in the stream of tweets, and writes the final results back to a Postgres database.

Figure 1 shows the overall architecture of the application. Figure 1 also shows the Storm topology that you need to develop for the application. Using the Tweepy library, the application reads the live stream of tweets from Twitter in the **tweet-spout** component. The **parse-tweet-bolt** parses the tweets, extracts the words from each parsed tweet, and emits the words to the next bolt component (**count-bolt**) in the topology. The **count-bolt** counts the number of each word in the received tuples, and updates the counts associated with each word in the **tweetwordcount** table inside the **tcount** Postgres database.



**File Structure**

The folder structure looks as follows:

extweetwordcount

src: contains the spout and bolts needed for the project

bolts: contains the bolts needed for the project

parse.py: parses the tweets and extracts words

wordcount.py: updates counts for each word and stores into tcount, a Postgres database.

spouts: contains the spouts needed for the project

tweets.py: collects live stream of tweets from Twitter

topologies: this folder contains the code to describe how to use the bolts and spouts

tweetwordcount.clj: code that informs the project how to use the bolts and spouts

virtualenvs

wordcount.txt: always required for streamparse projects

config.json: project configuration

fabfile.py: can use this to perform custom actions prior to or post topology submission.

project.clj: description of project

tasks.py: can use this to perform custom actions prior to or post topology submission

README: contains instructions on how to use this

finalresults.py: script to output a full list of words with at least 1 count or search for a specific word.

histogram.py: script will output a list of words between specified min and max counts.

prerun.py: this will drop the tcount data if it exists, and create a fresh new version.