Exercise 2 - W205 - 6 - Angela Gunn

Directory & File Structure

Project files can be found in https://github.com/anggunn/MIDS-W205.git

The project itself is in exercise_2/ with EX2Tweetwordcount being the actual application, results being additional files to review the results from running the application and screenshots which shows screen captures of the application running.

I will focus specifically on the structure of EX2Tweetwordcount.

Name of file	Location	Description
Twittercredentials.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/	Twitter App Keys
createtable.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/	Script to create the table “Tweetwordcount” in a Postgres database “tcount”
EX2tweetwordcount.clf	MIDS-W205/exercise_2/Ex2Tweetwordcount/topologies/	Topology for the program
Tweets.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/src/spouts/	The tweet spout for the program
Parse.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/src/bolts/	The bolt for parsing the tweets.
Wordcount.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/src/bolts/	The bolt for processing the words from the Parse bolt and saving to the database.

Additional files included in the program architecture but do not require additional discussion are listed below:

Name of file	Location
config.json	MIDS-W205/exercise_2/Ex2Tweetwordcount/
fabfile.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/
tasks.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/
__init__.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/src/bolts/

__init__.py	MIDS-W205/exercise_2/Ex2Tweetwordcount/src/spouts/
tweetwordcount.txt	MIDS-W205/exercise_2/Ex2Tweetwordcount/virtualenvs/

Application Idea

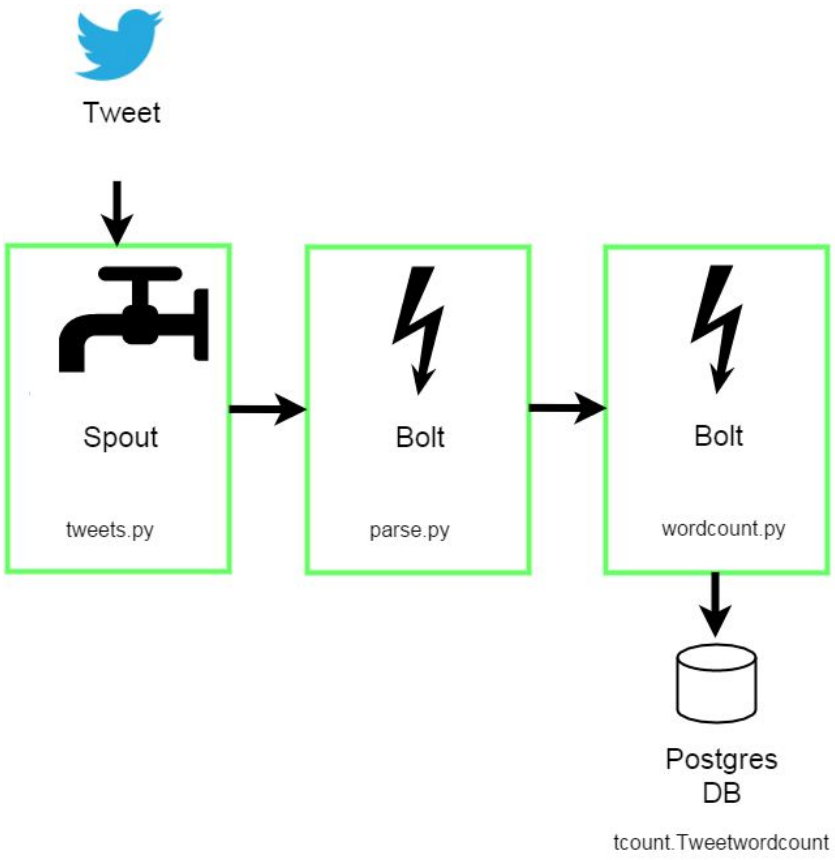
This application connects to a Twitter stream through Twitter's API and python's Tweepy library. It collects live tweets that show people's live interests, processes them in real time to get insights and aggregate the results in a database. Each unique word includes a count for the number of times the word has been encountered. Only alphanumeric characters appear in words stored, and

This application can be useful for analyzing frequency of keywords. For example, if we wanted to know how often the word "Trump" is being included in tweets, we can query the database and find the count.

Limitations of the application is it does not store the time of the tweet, and hence the time of the word's occurrence. It also does not indicate which words occurred together in the same tweet; there is no way of determining if "Donald" was in a tweet with the word "Trump" or the word "Duck."

Description of Architecture

The application contains a spout and two bolts. The spout, tweepy.py, collects the tweets from the live Twitter stream. The spout then sends the tweets to the bolt parse.py. This bolt takes the tweets and parses it into individual words, removing punctuation. From here, the resulting individual words are passed on to the second bolt, wordcount.py. This bolt does a final validation of the words before inserting them into the Postgres database table tcount.Tweetwordcount.



The original exercise diagram can be interpreted that multiple instances of each spout and bolt should be used. I originally implemented this architecture but ran into issues that the number of tweets coming in from my live stream were not in a volume appropriate for the capacity of that architecture. Reducing the architecture to just one instance of each spout and bolt was sufficient.

File Dependency

In order to run the application, the following python libraries must be installed:

Library	Description
pyscopg2	PostgresSQL adapter for the Python programming language
streamparse	Lets you run Python code against real-time streams of data via Apache Storm
tweepy	An easy-to-use Python library for accessing the Twitter API)

Information for Running Application

The application assumes it is being executed on the Amazon EC2 AMI **ucbw205_complete_plus_ postgres_PY2.7** which contains the appropriate versions of Python for executing the code.

This document assumes the reader has a basic knowledge of connecting to the AMI, starting the required services (metastore, postgres, hadoop) and working with postgres databases.

Please see the README.md file for specific instructions in executing the application.