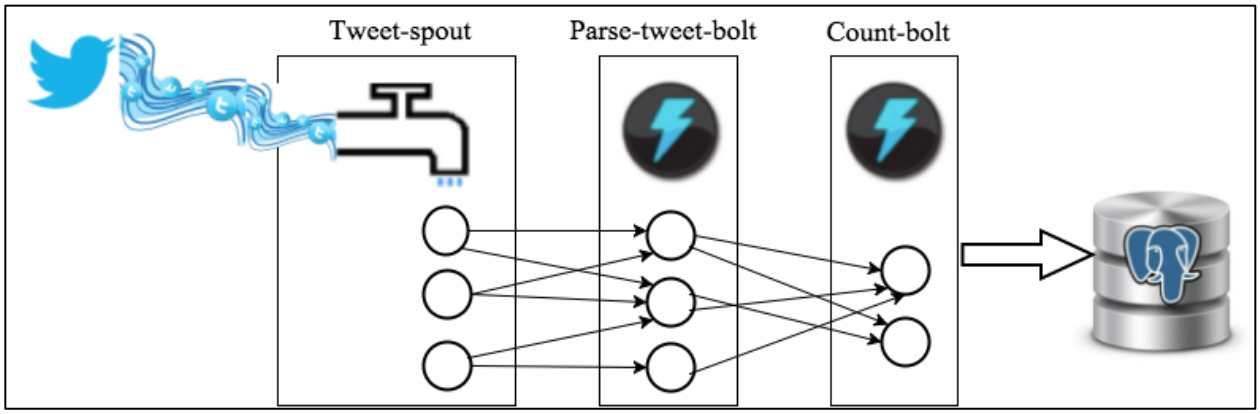
Application Architecture for Exercise 2

FILE STRUCTURE

```
----Exercise-2-Subject-205-Real Time Data Processing Using Apache Storm.pdf
  |
  -- EX2Tweetwordcount
     ├── README.md
     ├── Readme.txt
     ├── config.json
     ├── error_log.txt
     ├── fabfile.py
     ├── finalresults.py
     ├── histogram.py
     ├── project.clj
     ├── src
     │   ├── bolts
     │   │   ├── __init__.py
     │   │   ├── parse.py
     │   │   └── wordcount.py
     │   └── spouts
     │       ├── __init__.py
     │       └── tweets.py
     ├── tasks.py
     ├── topologies
     │   └── tweetwordcount.clj
     └── virtualenvs
         └── wordcount.txt
```

APPLICATION IDEA

This Exercise was intended to process real-time language data from Twitter. The data would stream in through Tweets, be partitioned into separate words, be counted, and then be stored and updated as word-count pairs in a database. The streaming application would run thanks to Apache Storm and streamparse, and the database would be PostgreSQL. A diagram of the application concept is presented below.



DESCRIPTION OF ARCHITECTURE

The Storm topology illustrated above contains a data stream from Twitter that feeds directly into 3 spouts. The Tweets are then parsed by 3 bolts so that they are segmented into words. Each of the words are summed up on 2 bolts, which then use psycopg2 to feed the words and their sums into the database, which is updated as new Tweets are parsed and counts are found.

FILE DEPENDENCIES

The application relies on the tweetwordcount.clj topology specification, which in turn points to the spout and bolt files: tweets.py, parse.py, and wordcount.py.

RUNNING THE APPLICATION

In order to run the application, Storm, streamparse, tweepy, psycopg2, and Postgres should be installed. Then, a Twitter account should be made and the application credentials entered into the tweets.py file. The Postgres daemon should be started and a Tcount database with a Tweetwordcount table should be created. Then, from within the EX2Tweetwordcount directory, one should type `sparse run` to activate streamparse. These instructions can also be found in the Readme.