MIDS W205: Storing and Retrieving Data
Exercise 2: Twitter Count Application
Dustin Matthew Burt

Directory and file structure

```
Dmburt/w205
|-- README.md
|-- finalresults.py
|-- histogram.py
|-- plot.png
|
|-- _screenshots
|   |-- finalresults-all.png
|   |-- finalresults-given-word.png
|   |-- histogram.png
|   `-- tweetcounter-running.png
|
|-- _build
|   |-- classes
|   |   `-- META-INF
|   |       `-- maven
|   |           `-- EXTwoTweetwordcount
|   |               `-- EXTwoTweetwordcount
|   |                   `-- pom.properties
|   `-- stale
|       `-- leiningen.core.classpath.extract-native-dependencies
|
|-- _resources
|   `-- resources
|       |-- bolts
|       |   |-- __init__.py
|       |   |-- old_wordcount.py
|       |   |-- parse.py
|       |   `-- wordcount.py
|       `-- spouts
|           |-- __init__.py
|           `-- tweets.py
|
|-- config.json
|-- fabfile.py
|-- finalresults.py
|-- histogram.py
|-- logs
|-- project.clj
|-- src
|   |-- bolts
|   |   |-- __init__.py
|   |   |-- old_wordcount.py
|   |   |-- parse.py
|   |   `-- wordcount.py
|   |
|   `-- spouts
```

(continued from previous page)

```
        <spouts>
|       |-- __init__.py
|       `-- tweets.py
|
|-- tasks.py
|-- topologies
|   `-- tweetwordcount.clj
|
`-- virtualenvs
    `-- tweetwordcount.txt
```

File dependencies

- All processes should run on UCB W205 Spring Ex 2 image
- Requires streamparse, Python 2.7, PostgreSQL
- All streamparse files included in repository

Startup

1) Initialize environment with **start.sh**
    a. Must be set to executable: chmod +x start.sh
2) Run **create_database.py**
    a. This script will create both the database and the *wordcount* table.
3) cd /EXTwoTweetwordcount
4) sparse run
    a. This is not a timer-limited application. Kill the process after a few minutes.

Description of architecture

Storm architecture:

- Spouts:
  - Tweets.py
    Connects to Twitter via tweepy Python library
- Bolts:
  - Parse.py
    Receives data feed from Tweets.py *spout*
    Splits tweet into individual words and removes:
    - *Twitter hash tags, user mentions, and retweets (i.e., #, @, RT)*
    - *URLs*
    - *Punctuation*
  - Wordcount.py
    Receives data from Parse.py *bolt*
    Executes three transactions on local PostgresSQL database while reading stream
    For data in queue:
    - *Query PostgresSQL database to find if word is already in the database table*
      - *If it is not, insert word (and count of 1).*
      - *If it is, update existing word and current count.*