W205 – Storing and Retrieving Data

Exercise 2

Carlos Castro

Introduction

Here I describe the architecture, design and technical aspects of a streaming application that counts words from Tweets in real time.

There are multiple technical challenges in streaming real-time data from a source as massive as Twitter. Using streaming facilities such as Apache Storm allows this seemingly humongous task to be attainable.

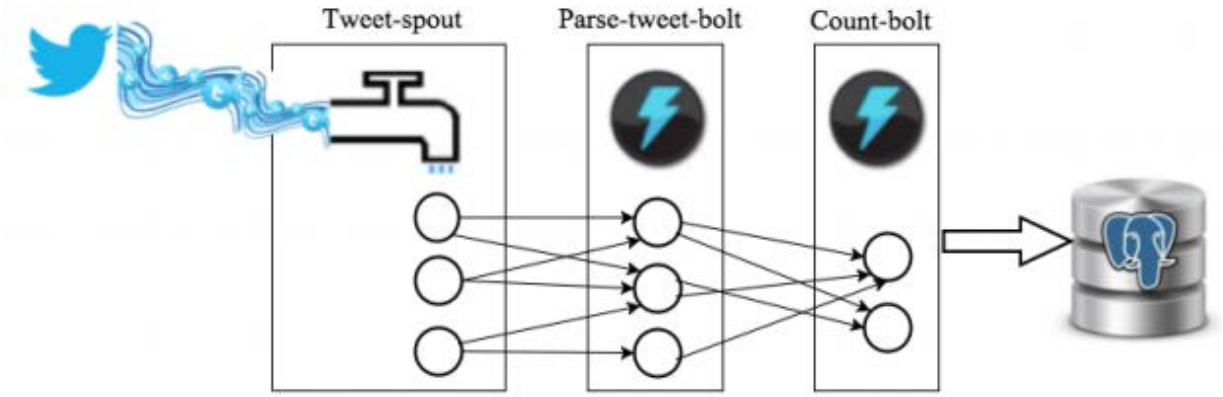
Despite the naturally interesting technical challenges, this architecture acts as basis of a framework with potential for amazingly impactful social and business applications. Some concrete examples include detection of cyber bullying, identification of psychopaths and early notification of natural disasters such as earthquakes.

Architecture

The diagram below depicts the high-level architecture and flow of our application. We use Apache Storm to process the real-time data stream from Twitter API. The real-time data from Twitter is digested in Spouts, which pass the data through a series of bolts.

The first bolt parses the tweet data and breaks it down into words, which are passed to the final bolt, which updates the counts for each word in a Postgres Sql database.

At any given time, the Postgres database holds the count for each word.



Dependencies

Here we list the external dependencies of our application. The Deployment section and deployment documentation in the repo explains how these dependencies can be set up to run the application.

- Storm:
  - **Description:** Apache storm is our streaming software
  - **Dependency Type:** Local machine Dependency
  - **Installation source:** Comes installed with the UCB MIDS W205 EX2-FULL AMI
- StreamParse:
  - **Description:** Libraries on top of apache storm that allow running and creating storm topologies, bots and spouts in a mainly python-based approach
  - **Dependency Type:** Local machine Dependency
  - **Installation source:** Comes installed with the UCB MIDS W205 EX2-FULL AMI
- Python
  - **Description:** Most of this project uses python, a high-level programming language
  - **Dependency Type:** Local machine Dependency
  - **Installation source:** Comes installed with the UCB MIDS W205 EX2-FULL AMI
- Postgres SQL
  - **Description:** SQL database to store words and their counts from Twitter.
  - **Dependency Type:** Local machine Dependency
  - **Installation source:** If not installed in the API, can be installed through running the script setup_ucb_complete_plus_postgres.sh from lab 2. See the deployment section under Readme.md at the repository's deployment folder for detailed setup steps
- Tweepy
  - **Description:** Python client libraries for the Twitter API
  - **Dependency type:** local machine dependency
  - **Installation source:** Tweepy is installed as part of the preparation for the project. See the deployment section under Readme.md at the repository's deployment folder for detailed setup steps
- Psycopg2
  - **Description:** Python client libraries Postgres SQL
  - **Dependency type:** local machine dependency
  - **Installation source:** Psycopg2 is installed as part of the preparation for the project. See the deployment section under Readme.md at the repository's deployment folder for detailed setup steps

- Twitter API
    - Description: External API to consume tweets in real-time
    - Dependency type: External API
    - Installation source: None. However, an API key and secret are necessary to connect to Twitter API. In the code we provide a working key and secret for sample

File Structure

Folders

Let's first observe a high level of the role of each folder in the repository

- **Deployment:** Contains deployment instructions, scripts, and general information
- **Docs:** Architecture documentation
- Screenshots: Captures of the application running in an AWS AMI
- **Serving_scripts:** scripts required in the project to gather information from the database through python
- **Tweetwordcount:** The streamparse project that collects Tweets from the Tweeter API, counts the words and writes them in a database

Files

File	Folder	Description
Readme.md	/exercise_2	Project description and structure
Readme.md	/exercise_2/deployment	Deployment steps
Architecture.pdf	/exercise_2/docs	Architecture & project documentation
tweetwordcount.clj	/exercise_2/tweetwordcount/topologies	Storm topology of the application
tweets.py	/exercise_2/tweetwordcount/src/spouts	Spout that consumes Twitter API to collect tweets in real-time
Parse.py	/exercise_2/tweetwordcount/src/bolts	Parses tweets and breaks them into words
Wordcount.py	/exercise_2/tweetwordcount/src/bolts	Counts words and saves in Postgres database
Finalresults.py	/exercise_2/serving_scripts	Script that retrieves the counts for all words or for a word
Histogram.py	/exercise_2/serving_scripts	Histogram showing words with counts between a min and max threshold
screenshot-servingScript-finalResults.JPG	/exercise_2/screenshots	Capture of the output of running the finalresutls serving script

screenshot-servingScript-histogram.JPG	/exercise_2/screenshots	Capture of the output of running the histogram serving script
screenshot-servingScript-histogramInputValidation.JPG	/exercise_2/screenshots	Capture of the error message from histogram script input validation when wrong parameters are passed
screenshot-streamparseRunningTwitterStream.JPG	/exercise_2/screenshots	Capture of the twitter scream from running streamparse
Plot.png	/exercise_2/data	Bar chart of the words with the top 20 word count
Barchart_data.csv	/exercise_2/data	Data for the words with the top 20 word count
setup_ucb_complete_plus_postgres.sh	/exercise_2/deployment	Script to set up postgres in the UCB full AMI