



Cloud Computing and Big Data Ecosystems Design

Trending Topics in Twitter

- **Description:** The goal of this assignment is to implement a Java application that reads tweets from both the Twitter Streaming API and preloaded log file and finds the trending topics for a given set of languages and periods. The application will run in a distributed environment and must use Apache Kafka to store the tweets and avoid information lost and Storm to find out the trending topics. The output of the application is a set of files, one for each language, with the most used hashtags (top-k) in between two occurrences of a given hashtag.
- **Deadline:** 16th December 2016
- **Where:** All the required files must be uploaded to Moodle by the deadline. The file must be named ID.rar (ID is the id of the students provided by the instructor). If you have developed the twitterApp and storm topology in 2 different maven projects, the structure of your delivery will be:

- ID.rar
 - twitterApp
 - pom.xml
 - src/
 - trandingTopology
 - pom.xml
 - src/
 - kafkaTopics.sh

in the case you developed everything in the same project:

- ID.rar
 - masterApp
 - pom.xml
 - src/
 - kafkaTopics.sh

The TwitterApp must be configured with the **appassembler** maven plugin. The name of the script to launch the twitterApp must be **startTwitterApp.sh**. The main class of the tradingTopology must be **master2016.Top3App** (master2016 is the package name).

- **Groups:** The project is implemented by 2 persons of the same master program. Groups must be registered sending an email to mpatino@fi.upm.es and vvianello@fi.upm.es with subject: "master2016 project1" by **11/11/2016**. The group id will be assigned replying to this email.
 - In the email you must provide: full name and id (passport) number of the two students of the group and name of the master program.
- **Requirements:**
 - The application must be developed using the versions of the software: Oracle Java 7, Storm 1.0.2, Kafka 2.11-0.10.1.0 and deployed using Ubuntu 14.04.
 - Live tweets must be downloaded using the streaming API from Twitter using the URL <https://stream.twitter.com/1.1/statuses/sample.json>
 - Entities provide meta-information about tweets <https://dev.twitter.com/overview/api/entities>. This information is in JSON format and is used to process tweets by the java application
 - The log file with tweets has the following format:
 - 1 tweet per line
 - Tweets have the same JSON format as the ones returned by the streaming API
- **To be Released:**
 - Script to start the Java application used to store tweets in Kafka.
 - Script name: **startTwitterApp.sh**
 - Script parameters:
 - mode: 1 means read from file, 2 read from the Twitter API.
 - apiKey: key associated with the Twitter app consumer.
 - apiSecret: secret associated with the Twitter app consumer.
 - tokenValue: access token associated with the Twitter app.
 - tokenSecret: access token secret.
 - Kafka Broker URL: String in the format IP:port corresponding with the Kafka Broker
 - Filename: path to the file with the tweets (the path is related to the filesystem of the node that will be used to run the Twitter app)
 - The script will be always invoked with 7 parameters with both modes even if some parameters will not be used according with the mode.
 - The script **startTwitterApp.sh** must be tested on Ubuntu nodes and must be created using the appassembler maven plugin.
 - Script named **kafkaTopics.sh** used to create the needed topics in Kafka. It will receive a comma separated list of languages. Invocation example: `./kafkaTopics.sh es,it,en,pt`

Storm Topology

- **Description:** Storm topology for calculating the three more common hashtags (top3) in a set of languages over a configurable period of time (conditional window).
- **Requirements:**
 - The conditional window keeps all the tuples between two occurrences of a given word in the same language.
 - Top3 algorithm:
 - Given a set of languages, the Top3 algorithm calculates the three most used hashtags for a given language in between two occurrences of a given hashtag (defined by the conditional window).
 - The **lang** field of the tweets identifies the language. The path for the language in the JSON structure representing a tweet is “root-> lang”
 - The JSON representation of the tweet contains a field named **hashtags**. The path in the JSON is “root->entitites->hashtags->text”.
 - Topology output:
 - For each given language the topology must produce an output file named **lang_ID.log**, where the *lang* value is the one in the tweet and the ID is the number associated with the students implementing the work.
 - Each line in the file describes the top 3 hashtags in the last window interval using the format: counter, lang, tophashtag1, frequencyhashtag1, tophashtag2, frequencyhashtag2, tophashtag3, frequencyhashtag3
 - Where counter identifies the iteration of the conditional window.
 - If in a given window interval the list is not complete (only 1-2 hashtags are received in the interval), the output will be “null” as value for the hashtag and 0 for the counter.
 - Topology:
 - Name of the main class must be: master2016.Top3App
 - Parameters:
 1. langList: String with the list of languages (“lang” values) we are interested in and the associated special token. The list is in CSV format, example: en:house,pl:universidade,ar:carro,es:ordenador
 2. Kafka Broker URL: String IP:port of the Kafka Broker.
 3. topologyName: String identifying the topology in the Storm Cluster.
 4. Folder: path to the folder used to store the output files (the path is relative to the filesystem of the node that will be used to run the Storm Supervisor)
- **To be Released:**
 - Jar file with the topology ready to be deployed into a Storm cluster.

- **Output example:**

Let us consider the following input sequence where we are interested in the language **es** with word **ordenador** and language **en** with word **house**:

es, casa
es, ordenador -> start (es)
es, coche
en, football
en, house -> start (en)
es, casa
es, casa
en, football
en, messi
en, football
es, coche
es, casa
es, libro
en, messi
en, house -> stop (en)
es, work
es, ordenador -> stop (es)
en, what
es, concierto
es, ordenador -> start (es)

The output would be:

File: es_X.log

1,es,casa,3,coche,2,libro,1

File: en_X.log

1,en,football,2,messi,2,null,0 -> "football" is the top1 and "messi" the top2 because when there is a tie the ranking is done according with the alphabetic order.