**CS6350**

**Big data Management Analytics and Management**

**Fall 2020**

**Homework 3**

**Submission Deadline: November 11th, 11:59 p.m.**

**Spark Streaming and Visualization**

Q1.

You are required to implement the following framework using Apache Spark

Streaming, Kafka (optional), Elastic, and Kibana. The framework performs SENTIMENT analysis of particular hash tags in twitter data in real-time. For example, we want to do the sentiment analysis for all the tweets for #trump, #coronavirus. Note that if you implement this framework with Scala, there is no need for Kafka and you can connect to twitter via the internal API. But if you want to implement it with Python, Kafka is required. Be careful about the Scala version compatibility.

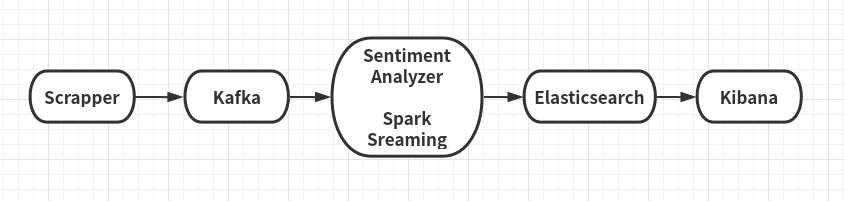


Figure: Sentiment analysis framework

The above framework has the following components:

**1. Scrapper (for python, but Scala needs to produce same result)**

The scrapper will collect all tweets and sends them to Kafka for analytics. The scraper will be a standalone program written in PYTHON and should perform the followings:

a. Collecting tweets in real-time with particular hash tags. For example, we

will collect all tweets with #trump, #coronavirus.

b. After filtering, we will send them to Kafka in case if you use Python.

c. You should use Kafka API (producer) in your program

(https://kafka.apache.org/090/documentation.html#producerapi)

d. Your scrapper program will run infinitely and should take hash tag as input parameter while running.

**2. Kafka (for Python)**

You need to install Kafka and run Kafka Server with Zookeeper. You should create a dedicated channel/topic for data transport

**3. Spark Streaming**

In Spark Streaming, you need to create a Kafka consumer (for python, shown in the class for streaming) and periodically collect filtered tweets (required for both Scala and python) from scrapper. For each hash tag, perform sentiment analysis

using Sentiment Analyzing tool (discussed below).

**3. Sentiment Analyzer**

Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker.

For example,

“President Donald Trump approaches his first big test this week from a

position of unusual weakness.” - has positive sentiment.

“Trump has the lowest standing in public opinion of any new president in

modern history.” - has neutral sentiment.

“Trump has displayed little interest in the policy itself, casting it as a

thankless chore to be done before getting to tax-cut legislation he values

more.” - has negative sentiment.

The above examples are taken from CNBC news:

http://www.cnbc.com/2017/03/22/trumps-first-big-test-comes-as-hes-in-an-

unusual-position-of-weakness.html

You can use any third-party sentiment analyzer like Stanford CoreNLP

(Scala), NLTK(python) for sentiment analyzing. For example, you can

add Stanford CoreNLP as an external library using SBT/Maven in your

Scala project. In python you can import NLTK by installing it using pip.

**4. Elasticsearch**

You need to install the Elasticsearch and run it to store the tweets and their sentiment information for further visualization purpose.

You can point [http://localhost:9200](http://localhost:9200/) to check if it’s running.

For further information, you can refer:

https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html

**5. Kibana**

Kibana is a visualization tool that can explore the data stored in Elasticsearch. In this assignment, instead of directly output the result, you are supposed to use the visualization tool to show your tweets sentiment classification result in a real-time manner.

Please see the documentation for more information:

<https://www.elastic.co/guide/en/kibana/current/getting-started.html>

What to submit:

1. Python/Scala code

2. Screenshots of your visualization charts

**Recommendation Systems:**

Q2. Using spark machine learning library spark-mlib, **use KMeans to cluster the movies using the ratings given by the user,** that is, use the item-user matrix from **itemusermat File provided** as input to your program.

**Dataset description.**

**Dataset: Itemusermat File.**

The **itemusermat file contains** the ratings given to each movie by the users in **Matrix format.** The file contains the ratings by users for 1000 movies.

Each line contains the movies id and the list of ratings given by the users.

A rating of 0 is used for entries where the user did not rate a movie.

From the sample below, user1 did not rate movie 2, so we use a rating of 0.

A sample **Itemusermat file** with theitem-user matrix is shown below.

|  |  |  |
| --- | --- | --- |
|  | user1 | user2 |
| movie1 | 4 | 3 |
| movies2 | 0 | 2 |

Set the number of clusters (**k**) to 10

Your Scala/python code should produce the following output:

* For each cluster, **print any 5 movies in the cluster. Your output should contain the movie\_id, movie title, genre, and the corresponding cluster** it belongs to. **Note:** Use the **movies.dat** file to obtain the movie title and genre.

For example

**cluster: 1**

**123, Star Wars, sci-fi**

**Q3.** Use Collaborative filtering find the accuracy of ALS model accuracy. Use **ratings.dat** file. It contains User id :: movie id :: ratings :: timestamp. Your program should report the accuracy of the model.

For details follow the link: http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html

**Please use 70% of the data for training and 30% for testing and report the accuracy of the model.**