CS6350: BIG DATA ANALYTICS and MANAGEMENT

Fall 2017

HW#3

Data Analytics and Stream Framework using Spark

Due: Nov 29 2017 by 11.59 p.m.

This homework consists of two parts. Here, we focus on K-means clustering, recommendation (collaborative filtering), classification and steam framework with Spark, Kafka.

Part A:

Q1 Clustering

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 the item-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

Q2. Use Collaborative filtering find the accuracy(MSE) 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 60% of the data for training and 40% for testing and report the accuracy of the model.

Due: Nov 29 at 11.55 p.m

Question 3

Advanced Analytics (Classification using Kernel Mean Matching)

In this part, you will use Kernel Mean Matching (KMM) algorithm for bias-correction in a dataset and find out the accuracy of classification algorithm.

What is Bias?

Many real-world applications exhibit scenarios where training and test data are drawn from the same distribution. But there might be some scenarios where training and test data are drawn from different distributions. This is referred to as Sample Selection Bias, that is, training data distribution are biased compared to test data distribution.

What is KMM?

Kernel Mean Matching (KMM) is a well-known method for bias correction by estimating density ratio between training and test data distribution. This mechanism re- weights training data instances so that their weighted data distribution resembles that of the observed test data distribution.

Dataset:

In this question you will use the app dataset.

App Dataset was collected by executing multiple Android apps on a Samsung Galaxy S device, running Android version 4.3.1. Total 30,000 apps were randomly selected from three different categories in Google Play Store. The categories include Finance, Communication, and Social. Apps were then installed and launched on the phone which is connected to the Internet via a wireless router. Each trace per app is collected over a 30-sec period passively using a mirroring switch at the wireless router. Various features from these traces and network packets are extracted to form the data collection.

The dataset file names are as follows:

datafile-
9xpoq4i8k20.c0.d4.C23.N2000.t16.T4.D1.E1.F1.G1.H1.I1.B16.J8.K300.L0.05.M100.A1.V0.P0.G0.l0.0.b600

datafile-
mjnh4rkek20.c0.d4.C23.N2000.t16.T4.D1.E1.F1.G1.H1.I1.B16.J8.K300.L0.05.M100.A1.V0.P0.G0.l0.0.b600

datafile-
qrsnc2a7k20.c0.d4.C23.N2000.t16.T4.D1.E1.F1.G1.H1.I1.B16.J8.K300.L0.05.M100.A1.V0.P0.G0.l0.0.b600

datafile-98m7fto4k100.c0.d4.C23.N2000.t16.T4.D1.E1.F1.G1.H1.I1.B16.J8.K300.L0.05.M100.A1.V0.P0.G0.I0.0.b600

datafile-451nedtok100.c0.d4.C23.N2000.t16.T4.D1.E1.F1.G1.H1.I1.B16.J8.K300.L0.05.M100.A1.V0.P0.G0.I0.0.b600

datafile-ayck0vm8k100.c0.d4.C23.N2000.t16.T4.D1.E1.F1.G1.H1.I1.B16.J8.K300.L0.05.M100.A1.V0.P0.G0.I0.0.b600

Every data file has two parts: train and test. At the start of the data file, each attribute name is listed beginning with the tag '@ATTRIBUTE'. The data starts after the tag '@DATA'. Columns are separated using '**,**' (comma) and the last column represents the app class.

Figure: screenshot of a portion of datafile named datafile-9xpoq4i8k20.c0.d4.C23.N2000.t16.T4.D1.E1.F1.G1.H1.I1.B16.J8.K300.L0.05.M100.A1.V0.P0.G0.I0.0.b600

Task for question 3:

Use KMM algorithm to do a bias correction on the app dataset. Then use this bias corrected data and do a classification to predict the app using any state-of-the-art machine learning classification algorithm. Measure the accuracy of the classification

on the bias corrected data. Also, apply that classification algorithm on biased dataset (without using KMM) and measure the accuracy. Compare two accuracies.

Source code of KMM is provided in the links section. You may use any machine learning packages or libraries (e.g. Scikit Learn, TensorFlow etc.)

What to submit for question 3:

1. Submit the source code that uses KMM and any classification algorithm to find out the accuracy of a biased dataset.
2. Report of your analysis for each dataset. Report should contain the accuracies for both classifications (with KMM and without KMM), the overall best result, all the parameters used and any assumptions you have made. Moreover, you should briefly mention which classification algorithm you have chosen and why.
3. A readme file containing instructions on how to run your code

Links:

Dataset: Download app dataset

KMM source code:
https://github.com/swarupchandra/multistream/blob/master/kmm.py

Useful paper about KMM:

- J Huang, A Gretton, KM Borgwardt, B Schölkopf, AJ Smola. Correcting sample selection bias by unlabeled data. In Advances in neural information processing systems, pages 601-608, 2006.

Q4. Spark Streaming and Kafka

You are required to implement the following framework using **Apache Spark Streaming, and Kafka (optional).** 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, #obama** and collect their location information (city or state or country). 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.

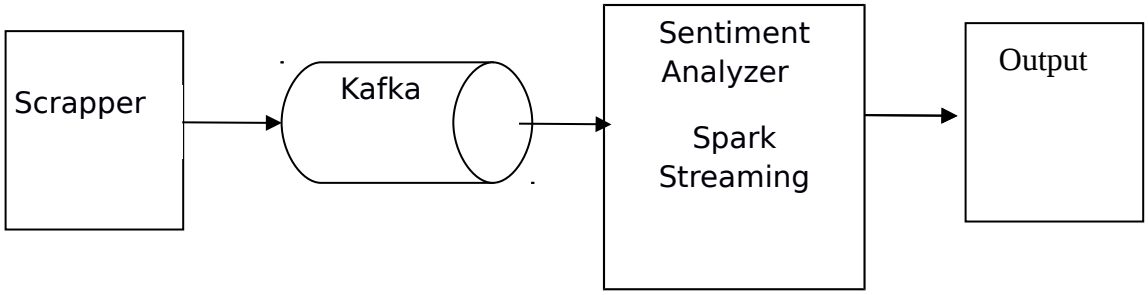

Figure: Sentiment analysis framework

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 JAVA/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**, **#obama**.

b. After getting tweets we will filter them based on their locations. If any tweet does not have location, we will discard them.

c. After filtering, we will send them (tweets with location) to Kafka in case if you use Python.

d. You should use Kafka API (producer) in your program (https://kafka.apache.org/090/documentation.html#producerapi)

e. You 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. 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 (java/scala), nltk(python) for sentiment analyzing. For example, you can add Stanford CoreNLP as an external library using SBT/Maven in your scala/java project. In python you can import nltk by installing it using pip.

  Sentiment analysis using Spark Streaming:

  In Spark Streaming, 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 above). Then for each hash tag, save the output with twitter itself.