Pig Case Study

Using

Movie Data Set

#	Document revision details	Date	Author
1	Initial Draft – version 2.0	25-December-2016	EduPristine

Overview

This document is targeted for trainers and anyone who is interested to solve the complete case study mentioned in Pig content v2.0.

Content

This document covers description, source and format of the data set. There are two main problem statement solved in this document. This document can be considered as instructor’s manual.

It is expected, trainer will explain each statement and its outcome, schema etc while solving case study.

Environment Requirement

In order to implement suggested solution, It is required to have access to Pig either running on any Hadoop Cluster. Hadoop Cluster can be Standalone, Pseudo Distributed running in your local machine, Virtual Machine or in Cloud.

Method to access environment depends on your Infrastructure setup, Other than that remaining should work as it is without any change in code/approach.

This case study has been tested on Cloudera VM 4.X, 5.X and Edu Pristine’s Cloud Instance of Hadoop

Data set overview

The sample Data-set in this tutorial is taken from MovieLens Labs.
The data set is stable benchmark dataset of 100,000 ratings from 1000 users on 1700 movies. Released in 4/1998.

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

Data Directory: movie-100k
There are multiple files in this directory and for the purpose of this case study we will use below mentioned data.

File:	**u.data**
Description:	Ratings data
Fields in File:	user id, item id, rating, timestamp
Separator:	Tab [\t]
File:	**u.item**
Description:	Movie data
Fields in File:	movie id, movie title, release date, video release date,IMDb URL, unknown, Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir,Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western
Separator:	Pipe [|]
File:	**u.user**
Description:	User demographic data
Fields in File:	user id, age, gender, occupation, zip code
Separator:	Pipe [|]

Instruction Manual for Trainers

This simple case study is developed to be delivered in Pig Session. This case study covers all major and important functions used in Pig like grouping, Filter and joining.

The method to deliver Pig session is, get the case study done by trainees while discussing the required topics.

1. Load data to HDFS

```
hadoop fs –put  /local path to data set/movie-100k      /desired-hdfs-path/
```

User may have data set in flash drive or client machine. First user needs to move data to VM/Cloud. Then put the data into HDFS.

2. Start Pig Grunt

```
> pig
```

Type “pig” and get into grunt – pig console.

3. First Pig Statement - Load data from u.user

users = LOAD '/user/training/movie-100k/u.user' USING PigStorage('|')   AS (userId, age, gender, occupation, zipCode);

At this stage data from u.user is loaded in "user".

4. Extract Simple Statistics

allUsers = GROUP users ALL;

Group users data set unconditionally – using ALL. This will help us to get count, average of some users attributes

stats = FOREACH allUsers GENERATE COUNT(users), AVG(users.age), SUM(users.age);

Iterate through allUsers using FOREACH. For each Tuple in this relation, generate new Tuple with three fields, Count(users), Average of (users.age) and Sum(users.age)

byGender = GROUP users BY gender;

DESCRIBE byGender; - Observe schema here and discuss with students

Group by gender. It will return a bag with two fields. First field is group key, second is another bag, which contains all tuple that belongs to particular key. Try ' DESCRIBE byGender' after above command to understand the schema.

```
genderStats = FOREACH byGender GENERATE group, COUNT(users), AVG(users.age);
```

Group by Gender and generate Tuple with three fields, group key, Count of users and average of users age.

```
programmers = FILTER users BY occupation == 'programmer';
```

Filter uses by Occupation

```
progsByAge = GROUP programmers BY age;

progCountsByAge = FOREACH progsByAge GENERATE group AS age,

COUNT(programmers) as NumProgs;

progsCountsByAgeSorted = ORDER progCountsByAge BY NumProgs DESC;

STORE stats INTO '/user/training/movie-100k/output/stats' USINg PigStorage('\t');

STORE genderStats INTO '/user/training/movie-100k/output/genderStats' USING
PigStorage('\t');

STORE     progsCountsByAgeSorted     INTO
'/user/training/movie100k/output/progsCountsByAgeSorted'
USING     PigStorage('\t');
```

5. Finding the Top 25 rated movies

```
votes = LOAD '/user/training/movie-100k/u.data' USING PigStorage('\t') AS
(userId,itemId,rating,timestamp);

movies = LOAD '/user/training/movie-100k/u.item' USING PigStorage('|') AS
    (movieId,movieTitle, releaseDate, videoReleaseDate, imdbURL, unknown, Action,
    Adventure, Animation, Childrens, Comedy, Crime, Documentary, Drama, Fantasy, FilmNoir,
    Horror, Musical, Mystery, Romance, SciFi,Thriller,War,Western);

movieVotesGroup = GROUP votes BY itemId;

movieVotes = FOREACH movieVotesGroup GENERATE FLATTEN(group) AS movieId,
                         AVG(votes.rating) AS avgRating, COUNT(votes) AS numVotes;

moviesWithVotes = JOIN movieVotes BY movieId, movies BY movieId;
```

Join two data sets by common field, movieId

```
moviesWithVotesSorted = ORDER moviesWithVotes BY movieVotes::avgRating DESC;

top25 = LIMIT moviesWithVotesSorted 25;

STORE top25 INTO '/user/training/movie-100k/output/topMovies' USING PigStorage('\t');
```

6. Are newer movies better ?

First prep data using Pig:

```
votes = LOAD '/user/training/movie-100k/u.data' USING PigStorage('\t') AS
(userId,itemId,rating,timestamp);

movies = LOAD '/user/training/movie-100k/u.item' USING PigStorage('|') AS    (movieId,
movieTitle, releaseDate, videoReleaseDate,    imdbURL, unknown, Action, Adventure,
Animation,Childrens,Comedy,Crime,Documentary,Drama,Fantasy,FilmNoir,
Horror,Musical,Mystery,Romance,SciFi,Thriller,War,Western);

movieVotesGroup = GROUP votes BY itemId;

movieVotes = FOREACH movieVotesGroup GENERATE FLATTEN(group) AS movieId,
                        AVG(votes.rating) AS avgRating, COUNT(votes) AS numVotes;
                        moviesWithVotes = JOIN movieVotes BY movieId,movies BY movieId;

movieRatings = FOREACH moviesWithVotes GENERATE movieVotes::movieId,
                        movieVotes::avgRating, movieVotes::numVotes,
                        REGEX_EXTRACT(movies::releaseDate,'(.*)-(.*)-(.*)',3), movies::movieTitle;

STORE movieRatings INTO '/user/training/movie-100k/output/movieRatings' USING
PigStorage('\t');
```