BIG DATA ETL EXERCISES

ABSTRACT

This document outlines the exercises to be undertaken during the ETL week of the Big Data course

QA Consulting Academy Team

Big Data Academy

Contents

Introduction 2
Practice exercises 2
Loading your data 2
Citations 2
Exercise 1: Using Pig for ETL processing 3
Working in the Grunt Shell 3
Complex Pig 4
Exercise 2: Recommending Movies with Pig 6
Extracting the Data 6
Transforming the Data 6
Loading the Data 6
Challenges 7

Introduction

This document will take you through some exercises to extract, transform, and load data. This will involve adding data to your cluster, cleaning it, and arranging it in a way that will be beneficial to analysts and developers.

For the practice exercises, you should work in your own user space in HDFS, as each one of you will be going through it individually. However for the project work, the data will span across your whole cluster so it will need to be in an area you can all access.

Practice exercises

The document goes through some practice exercises for Pig on the movielens database that should be easy enough to bring into your cluster. This should get you comfortable performing some of these commands and how to get started.

Loading your data

For this course you have a large amount of data of reviews from Amazon products between 1996 and 2014. It is up to you to investigate this data and decide how to lay it out - you may not always have access to a schema ready to explain everything to you.

It is currently in two main sections, the data and the metadata, both of which are in .json format. Use the **cat**, **tail**, or **head**, commands to check through your files and understand the format the data is in. From here you can start working in your teams to design your database structure and how you want to arrange it. Remember, you will all be analysing this individually, so it is suggested that you consider how the resources you have will be used.

Citations

This data is from the following two papers, and if you have time, it would be worth giving them a read.

Image-based recommendations on styles and substitutes
J. McAuley, C. Targett, J. Shi, A. van den Hengel
SIGIR, 2015

http://cseweb.ucsd.edu/~jmcauley/pdfs/sigir15.pdf

Inferring networks of substitutable and complementary products
J. McAuley, R. Pandey, J. Leskovec
Knowledge Discovery and Data Mining, 2015

http://cseweb.ucsd.edu/~jmcauley/pdfs/kdd15.pdf

Exercise 1: Using Pig for ETL processing

In the exercise, you will use Pig to work with the movielens dataset to extract and transform data.

Working in the Grunt Shell

Practice running Pig commands in the Grunt shell.

1. Before we get started you'll need to install pig. Use whichever machine you have designated as your 'client'.

   ```
   $ sudo yum install pig
   ```

   ```
   $ sudo apt-get install pig
   ```

2. We're going to work on the local file system to start out, so change directory to where you have stored the data from ml-latest. Start the Grunt shell in local mode.

   ```
   $ pig –x local
   ```

3. Load the data in the movies.csv file into Pig and dump it. It's worth noting here that Pig is 'lazy'. It won't actually load the data until we tell it to do something with it, so the load statement won't actually execute until the dump statement is sent.

   ```
   grunt> data = LOAD 'movies.csv' using PigStorage (',');
   ```

   ```
   grunt> DUMP data;
   ```

4. Only load the first two columns' worth of data then dump it.

   ```
   grunt> first_2_cols = LOAD 'movies.csv' using PigStorage (',') AS
   (mid:chararray, mname:chararray);
   ```

   ```
   grunt> DUMP first_2_cols;
   ```

5. Use the DESCRIBE command to review the schema of first_2_cols.

   ```
   grunt> DESCRIBE first_2_cols;
   ```

6. See what happens if you use the DESCRIBE command on data. Recall that you loaded data, you did not define a schema.

   ```
   grunt> DESCRIBE data;
   ```

7. End your Grunt session.

   ```
   grunt> QUIT;
   ```

Complex Pig

Let's try something a little trickier. We'll stick with the Grunt shell for now and move into creating scripts later, which will be useful for complicated tasks but let's just experiment with some commands first. Before you get started, move your movielens data into HDFS as we will no longer be working in local mode. If you run into issues, check you are putting the full path to the data files in Pig, e.g. 'tags.csv' may actually need to be something like '/user/name/moviedata/tags.csv'.

1. Load the tags.csv file into Pig and dump a few records to the screen.

```
$ pig

grunt> users = LOAD 'tags.csv' using PigStorage (',') AS (uid:chararray,
mid:chararray, tag:chararray, timestamp:chararray, gender:chararray);

grunt> udata = FOREACH users GENERATE uid, mid, gender;

grunt> limited = limit udata 5;

grunt> dump limited;
```

2. Now we'll group our users by gender and see if we can find any interesting trends.

```
grunt> gengr = group users by gender;

grunt> genc = foreach gengr generate group, COUNT(users);

grunt> dump genc;
```

   You'll notice we have a mix of female and male users, and you should also see one entry of (gender, 1), indicating the group of the header.

3. Next, let's load some ratings data.

```
grunt> ratings = LOAD 'ratings.csv' using PigStorage(',') as (uid:chararray,
mid:chararray, rating:int, timestamp:chararray);

grunt> rlimit = limit ratings 5;

grunt> dump rlimit;
```

4. Then let's join our users and ratings and perform a little analysis. Note that this job will take a long time! Consider either using a sample (sampdata = sample udata 0.01;) or running over a break. We're going to find what each gender gives as an average score.

```
grunt> urjoin = join users by uid, ratings by uid;

grunt> urgr = group urjoin by gender;
```

```
grunt> avgs = foreach urgr generate group, AVG(urjoin.rating);

grunt> dump avgs;
```

5. We're now going to try and even more complicated join after loading some more data. We'll also describe this join so we can see exactly what it's doing.

```
grunt> movie = load 'movies.csv' using PigStorage (',') AS (mid:chararray,
title:chararray, genres:chararray);

grunt> mlimit = limit movie 5;

grunt> dump mlimit;

grunt> urmjoin = join urjoin by ratings.mid, movie by mid;

grunt> describe urmjoin;

grunt> joinlimit = limit urmjoin 5;

grunt> dump joinlimit;

grunt> maindata = foreach urmjoin generate urjoin::users::uid,
urjoin::rating::rating, movie::genres;

grunt> mainlimit = limit maindata 5;

grunt> dump mainlimit;
```

6. We're now going to use a filter to look at a particular genre of movies.

```
grunt> mainfilt = filter maindata by (movie::genres matches ''.*Comedy.*');

grunt> maingroup = group mainfilt by movie::genres;

grunt> dump maingroup;
```

7. Then finally, we'll see if there seems to be a gender preference for this genre of movies.

```
grunt> genavg = foreach maingroup generate group, AVG(mainfilt.ratings);

grunt> dump genavg;
```

Exercise 2: Recommending Movies with Pig

The previous exercises could have easily been created as a Pig script and run on the command line. Simply create a document ending with .pig and run it, either locally as pig –x local pigscript.pig or on HDFS as pig pigscript.pig. The following steps will take you through creating a script to give movie recommendations to users, but will only hint at what steps you should take.

Extracting the Data

Before we begin we should get our data into a useful format. You can do this on the command line or with a script, but I suggest you keep it separate from your final script to transform the data else it will take some time to run repeatedly.

We want one document to access all the data we need, so use JOIN to bring together a document that will contain the user IDs, the movie IDs, the rating, and limit this to movies with a minimum of 30 ratings.

Finally, you will need to use a STORE statement to save this document, rather than just printing it to the console, e.g. STORE finalData INTO '/user/movies/output';

Transforming the Data

Now you want to see if you can find any kind of correlation between movie ratings. Try grouping movies together and seeing if they get good ratings from the same users. That is, if one user rated The Lion King as 5 as well as Toy Story, it could be that we should recommend Toy Story to those that watch The Lion King. If lots of users seem to do that then we should definitely recommend it. Try finding the top occurrences of these correlations.

Loading the Data

Finally, you should save your findings to an external document using the STORE statement.

Challenges

This week has introduced you to Pig. In your team you need to load your data into your cluster. Consider what will be useful for you as an analyst to have and how you should lay it out. You then need to start cleaning your data, make sure it is consistent and usable.

Try practicing with small samples of data to find effective ways of cleaning and sorting your data. Everyone needs to practice working with Pig and should deliver a script for transforming the data. Overall however, this needs to be done within your teams as the data needs to be accessible across the cluster, and creating multiple copies of cleaned data will quickly fill up HDFS. It may be that some have different ideas as to how this should be done, and you do need to reach a consensus within your group as how everything will be set up, but you may include your own ideas within your report.

Bear in mind that you have a large amount of data and many people working on it at once. When cleaning, testing, and analysing data you should first take a small sample of it and experiment with it locally before running a program on the full cluster. This will prevent the cluster from becoming overloaded with work.