Assignment 3: Big Data Technologies

Instructions

Deadline

Please upload all your results **before February 10, 2017**!

Submission guidelines

Upload a single .zip/.tar.gz file named

`VU_BI_2016W_Assignment3_<group no>.zip/.tar.gz`

that contains all your deliverables, i.e.:

1. A single PDF document that describes your solution for all parts of the assignment.

   Filename: `VU_BI_2016W_Assignment3_<group no>.pdf`

2. All solution folders (cf. deliverables outlined for each exercise):
   - `1a_AvgReviewScoreAmazon`
   - `1b_SentimentAmazon`
   - `2_MovieLensHive`

Please make sure you list all team members on the title page and the page headers of your solution document. It's sufficient to upload the final solution of the group once (i.e., one person for all members of the group).[1]

Questions

Please post general questions in the TUWEL discussion forum of the course. You can also discuss problems and issues you are facing there. We appreciate if you help other students out with general problems or questions regarding the tools used (and may take that into account in case you are short a few points for a better grade). For obvious reasons, however, please do not post any solutions there.

Please direct other questions (particularly those that are not of general interest for other students) to this semester's BI tutor, Gerta Sheganaku:

`gerta.sheganaku@tuwien.ac.at`

You can also contact me directly (with very specific questions) at:

`elmar.kiesling@tuwien.ac.at`

Please use subject line `BI_2016_<your subject>`.

[1] Otherwise, the latest upload will be graded.

Assignment 3: Big Data Technologies

The goal of this assignment is to obtain hands-on experience with Hadoop, a widely-used platform for big data analyses and applications. The first exercise will focus on the "nuts and bolts" of Hadoop programming and teach you how to implement and run MapReduce programs based on the Hadoop Java framework. The second exercise will then move up the Hadoop stack and cover the Hive data warehouse infrastructure, which provides an SQL-like interface and allows users to query large data stored in a distributed filesystem.

Preliminaries: Lab Setup

You can solve this assignment on a single node cluster in a virtual machine on your local computer. The easiest way to set up a local Hadoop instance is to download a preconfigured Hadoop stack set up in a virtual machine. We tested the exercise on Cloudera's Quickstart VM (available for VMware and Virtualbox), available from [1]. Other Hadoop distributions or a local installation on your native OS should work too, but were not tested. Note that the Cloudera VM (like practically all virtual machines provided by other Hadoop distributors) requires a 64-bit host OS. It also includes an Eclipse IDE and a sample project that comes with the necessary dependencies configured.

Resources

[1] http://go.cloudera.com/vm-download

Assignment 3: Big Data Technologies

I. ⏎ MapReduce product review analysis in Java

The aim of this exercise is to get some initial hands-on experience with MapReduce programming in Java. You will implement two MapReduce programs to (i) efficiently calculate average review scores for a large number of products and (ii) conduct a very basic sentiment analysis on product review texts. You will likely use a single machine virtual machine cluster to develop and deploy your Map Reduce programs. Therefore, we'll use "small" subsets of Amazon product review data for testing purposes. However, you should design for near-linear scale-up so that your algorithms can be deployed on a large cluster to process very large amounts of data efficiently (e.g., the > 140 million Amazon reviews in the full data set).

Prerequisites
Before you get started, download data sets for three product categories of your choice from

`http://jmcauley.ucsd.edu/data/amazon/`

We will be using both the "ratings only" csv data (part a) and the "5-core" dense subsets (part b). Please download the "small" subsets available for experimentation.

Once you have downloaded three csv ("ratings only") and three json ("5-core") files for the three chosen product categories, copy them into your MapReduce programming environment (e.g., /home/cloudera/data/), extract them, and copy them into hdfs, e.g.:

```
hadoop fs -mkdir /amazon
hadoop fs –copyFromLocal <path> /amazon
```

Place the files in two subfolders named `/amazon/5core` and `/amazon/ratingsOnly`.

a) Average ratings

The goal in this part of the exercise is to implement a MapReduce program that calculates the average review score for all products in a given product category (i.e., a particular `ratings_<category>.csv` file).

To implement your program, you'll need (at the very least):

- a `ReviewScoreMapper` class that for each line extracts the relevant parts (i.e., the item and the rating contained in the second and third fields, respectively, of the csv files) and emits the respective product ratings (i.e., item-rating pairs).

  Example:
  `AGZ8SM1BGK3CK,B000GFDAUG,5.0,1198195200 → <B000GFDAUG,5.0>`

- an `AvgReducer` class that calculates the average rating for each item

- a `ReviewScoreAvgDriver` as your Main class

Once you have implemented and tested your program, run it for each of the three chosen product categories and include the first 15 results in your solution document.

Assignment 3: Big Data Technologies

Hints:

- This exercise can be solved in a structurally similar manner to prototypical word count MapReduce programming examples, except that the mapper(s) will emit product ids as keys (rather than words) and review ratings as values (rather than "1") whereas the reducer(s) will calculate averages (rather than summing up the counts for each word).
- Maven is recommended for build management. If you are using Cloudera Quickstart VM, you can copy the Stub class files provided in the example Eclipse project into a newly created Maven project. As a bare minimum, you'll need to include the following dependencies:

```
<dependencies>
 <dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-client</artifactId>
  <version>2.7.3</version>
 </dependency>
 <dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-mapreduce-client-core</artifactId>
  <version>2.7.3</version>
 </dependency>
</dependencies>
```

Deliverables:

1. `1a_AvgReviewScoreAmazon/source`
   Maven project with complete source code.

2. `1a_AvgReviewScoreAmazon/AvgReviewScoreAmazon.jar`
   Compiled jar file that can be run as a MapReduce job.

3. Your solution document should include:
   - A brief documentation of your implementation and a discussion of its scaling characteristics.
   - The average ratings of the first 15 products in your result for each category.

b) Sentiment analysis

In this exercise, you will conduct a very basic sentiment analysis by counting up the positive and negative words in all the review texts for each product and calculating the overall share of positive words. Hence, the overall sentiment score you should calculate for each product is defined as

```
sentiment = (positive - negative) / (positive + negative)
```

You can download the positive (pos-words.txt) and negative (neg-words.txt) word lists from TUWEL.

Your Map Reduce job should calculate sentiment scores for all products in a given `<category>_5.json` file. Once you have implemented and tested your program, calculate the sentiment value for each product in the 5-core data sets of the three categories you have chosen. Include the sentiment scores of the first 15 products in your result in your solution document.

Assignment 3: Big Data Technologies

Hints:

- You can find a tutorial for sentiment analysis using MapReduce at [1].

Deliverables:

1. `1b_SentimentAmazon/source`
   Maven project with complete source code.

2. `1b_SentimentAmazon/AvgReviewScore.jar`
   Compiled jar file that can be run as a MapReduce job.

3. Your solution document should include:
   - A brief documentation of your implementation and a discussion of its scaling characteristics. In particular, discuss questions such as:
     - How many invocations of the Map and Reduce methods are there?
     - In which phase of the process is most of the runtime being spent?
     - What speedup can you expect from distributing the job among more machines?
   - The sentiment scores of the first 10 products in each of your chosen categories.

Resources

[1] *Example: Sentiment Analysis Using MapReduce Custom Counters*
https://www.cloudera.com/documentation/other/tutorial/CDH5/topics/ht_example_4_sentiment_analysis.html (accessed December 18, 2016)

[2] *MapReduce Tutorial*:
https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (accessed December 18, 2016)

Dataset citation

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

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

Assignment 3: Big Data Technologies

II. ⏎ MovieLens dataset analysis with Hive

The goal of this exercise is to query a moderately large movie data set using Hive. This data set includes 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,00 users and includes a dense matrix of calculated relevance scores ("genome tags"). For details about the data set, cf. [1].

Prerequisites

1) Get the MovieLens 20M Dataset (190 MB compressed) from

   `http://grouplens.org/datasets/movielens/20m/`

   and extract it into your Hadoop environment.[2]

2) Strip the header line from each file[3]

```
sed -i 1d <files>
```

3) Copy the files into hdfs

```
hadoop fs -mkdir /movieLens
```

4) Create a new database in hive (either by invoking `hive` on the shell or using the web-based Hive query editor[4]):

```
CREATE DATABASE movieLens;
USE movieLens;
```

5) Create the tables:

   - movies (for movies.csv)
     Schema: movieId,title,genres
   - tags (for tags.csv)
     Schema: userId,movieId,tag,timestamp
   - ratings (for ratings.csv)
     Schema: userId,movieId,rating,timestamp
   - genome_scores (for genome-scores.csv)
     Schema: movieId,tagId,relevance

---

[2] You can also download and use the small subset to develop and test your queries. However, make sure to include your final results on the 20M data set in your solution document.
[3] Alternatively, you can skip the line when creating your hive tables using `TBLPROPERTIES ("skip.header.line.count"="1")`
[4] `http://quickstart.cloudera:8888/notebook/editor?type=hive` on Cloudera's Quickstart VM.

Assignment 3: Big Data Technologies

- genome_tags (for genome-tags.csv)
  Schema: tagId,tag

Your table declarations should:

- use appropriate data types
- specify that the actual data will be stored as a text file
- specify that the data is in row format and comma separated

You can test your code in interactive mode, but when you hand in the results, your solution folder should contain your HiveQL `CREATE TABLE` statements in a hql file that can be passed to Hive in batch mode (`createMovieLensTables.hql`).

6) Load data from the text files to populate the respective tables you just created in the previous step. Place your HiveQL `LOAD DATA` statements in a hql file named `loadMovieLens.hql`. In your solution document, also describe your understanding of what happens behind the scenes here.

Queries

Formulate the following Hive queries:

1) How many movies are there in total in the dataset?

2) How many movies in the dataset belong to the "Film-Noir" genre?

3) Which are the 10 most frequently assigned tags (by users, i.e., from the tags table)?

4) Which 10 movies were the most controversial in 2015 (i.e., had the highest *variance* in ratings between 2015/01/01 and 2015/12/31)?

5) Which movies (titles) are the 10 most frequently tagged and how often have they been tagged?

6) Which 15 movies (titles) have been most frequently tagged with the label "mars"?

7) Which are the 10 best-rated movies (on average; list titles) with more than 1000 ratings?

8) Which are the highest-rated "Film-Noir" movies with more than 10 ratings (average rating; movies with genre "Film-Noir", max. 10)?

9) What are the 15 most relevant genome tags for the movie "Toy Story (1995)" (movieId=1)?

10) Which are the 10 most relevant movies for Vienna (i.e., with the highest genome tag relevance rating for the tag "vienna")?

Deliverables

Place your HiveQL `SELECT` statements in hql files named `hive-query_<no>.hql`. Provide the answers to the questions in the solution document. For each query, also discuss how many MR jobs your query is translated into and why.

Assignment 3: Big Data Technologies

- `2_movieLensHive/createMovieLensTables.hql`
- `2_movieLensHive/loadMovieLens.hql`
- `2_movieLensHive/hive-query_<queryNo>.hql`

- Solution document:
  - Results for all queries
  - Description of your understanding of what happens behind the scenes, including a discuss on how many MR jobs your queries are translated into and why. Finally, also comment on what scale-up you would expect when running your queries on a real cluster in parallel.

Resources

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), http://dx.doi.org/10.1145/2827872

[2] Hive Wiki:
https://cwiki.apache.org/confluence/display/Hive/Home

[3] Hive Tutorial:
http://thinkbig-academy.s3.amazonaws.com/Strata2013/HiveTutorial/index.html

[4] Hive Language Manual: https://cwiki.apache.org/confluence/display/Hive/LanguageManual