Lab Course: distributed data analytics
Exercise Sheet 7

Nghia Duong-Trung, Mohsan Jameel
Information Systems and Machine Learning Lab
University of Hildesheim

Submission deadline: Friday 23:59PM (on LearnWeb, course code: 3117)

Note: this exercise sheet is for the first term students only.

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit a zip or a tar file containing two things a) python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in form of graphs and tables.
3. The submission should be made before the deadline, only through learnweb.
4. If you are M.Sc. Data Analytics summer 2017 intake student, you should submit to "First term students" link on LearnWeb.
5. And if you are M.Sc. Data Analytics winter 2016 intake student, you should submit to "Second term students" link on LearnWeb.
6. If you are not M.Sc. Data Analytics student, you can submit to anyone of the links above.

Preparing your Apache Spark

All the codes in this exercise sheet are tested in pyspark version 2.1.0, pre-built for Apache Hadoop 2.7 and later, python 3.5.

Exercise 1: Basic Resilient Distributed Dataset (RDD) (3 points)

Let's have two lists of words as follows:

- `a = ["spark", "rdd", "python", "context", "create", "class"]`
- `b = ["operation", "apache", "scala", "lambda","parallel","partition"]`

Create two RDD objects of `a`, `b` and do the following tasks. Words should be remained in the results of join operations.

1. Perform `rightOuterJoin` and `fullOuterJoin` operations between `a` and `b`. Briefly explain your solution. (1 point)
2. Using `map` and `reduce` functions to count how many times the character `"s"` appears in all `a` and `b`. (1 point)
3. Using `aggregate` function to count how many times the character `"s"` appears in all `a` and `b`. (1 point)

Exercise 2: DataFrame API and Spark SQL (7 points)

Use dataset `students.json` for this exercise. First creating DataFrames from the dataset and do several tasks as follows:

1. Replace the `null` value(s) in column `points` by the mean of all points. (0.5 point)
2. Replace the `null` value(s) in column `dob` and column `last_name` by `"unknown"` and `"--"` respectively. (0.5 point)
3. In the `dob` column, there exist several formats of dates, e.g. `October 14, 1983` and `26 December 1989`. Let's convert all the dates into `DD-MM-YYYY` format where `DD`, `MM` and `YYYY` are two digits for day, two digits for months and four digits for year respectively. (2 points)
4. Insert a new column `age` and calculate the current age of all students. (1 point)
5. Let's consider granting some points for good performed students in the class. For each student, if his point is larger than 1 standard deviation of all points, then we update his current point to 20, which is the maximum. See Annex 1 for a tutorial on how to calculate standard deviation. (2 points)
6. Create a histogram on the new points created in the task 5. (1 point)

You should report a snapshot of the final dataset in your submitted .pdf file.

Exercise 3: Pipeline API and ML workflows (8 points)

Let's assume that we have a little tweets dataset crawled by using Twitter API (See Annex 2, 3 for further reading). For the purpose of this exercise, we sample randomly 20 tweets and make it the dataset `tweets.json`. You might already know about TF-IDF weights from the previous exercise sheet. Basically, we have a *pipeline* of data preprocessing and calculating TF-IDF weights: crawling the dataset from sources, cleaning and tokenizing, calculating TF and IDF, and finally feeding processed data for any machine learning models.

In this exercise, you are going to design a `pipeline` to accomplish a sequence of tasks discussed above using Pipeline API and Spark ML. You might also need to explain why you come up with your pipeline solution in the submitted .pdf file.

Exercise 4: Word2Vec (2 points)

Word2Vec computes a distributed vector representation of words. The Word2Vec takens a sequence of words in a document and produces a fixed-size vector that presents the document. The `tweets.json` dataset in the exercise 3 is re-used in the exercise 4. Let's compute the word2vec of the datasets after removing stop words. You might briefly explain your solution in the submitted .pdf file.

Bonus (10 points)

This is the first bonus for Lab Course. It is optional. You are encouraged to do the bonus that you can gain a better score at the end of the semester. However, if you want to skip it, you are free to do.

0.1 Setting up your MongoDB Environment

First of all, you need to install MongoDB and set up your MongoDB server. Depending on your OS, please look at how to install MongoDB via the following instructions:

- Install MongoDB Community Edition on Windows `https://docs.mongodb.com/getting-started/shell/tutorial/install-mongodb-on-windows/`
- Install MongoDB Community Edition on Linux `https://docs.mongodb.com/getting-started/shell/tutorial/install-on-linux/`

If you prefer to manipulate with MongoDB using a Graphical User Interface, you can install `robomongo` - Native and Cross-platform MongoDB Manager `https://robomongo.org/`.

0.2 MongoDB Tutorials

After successfully installing MongoDB, let's jump into some good lectures and tutorials proposed below. And of course, there are much more tutorials that you can search for yourself.

- Penn State University `http://www.cse.psu.edu/~yul189/cmpsc431w/lectures.html`
- `https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm`
- `http://www.w3resource.com/mongodb/introduction-mongodb.php`
- `http://www.guru99.com/mongodb-tutorials.html`
- `https://www.mongodb.com/presentations`

Notes

In order to accomplish exercises and tasks accordingly, you should provide:

1. your mongoDB codes,
2. and the equivalent screen shots.
3. If you want to explain your codes, feel free to do it.

Bonus 1: Querying and Updating Documents (6 Points)

Use dataset `students.txt` for this exercise. Complete several tasks using basic querying, updating, and inserting mongoDB statements.

1. Display only students' `last name` and students' `course_gpas`. For the `course_gpas`, only display 5 elements starting at the 5th index in the `course_gpas` array. (1 point)
2. Display only students' `last name` and students' `SAT test scores`. You need to accomplish three requirements in this task: (2 points)
   (a) use `$elemMatch`,
   (b) `SAT test scores` are greater than 700,
   (c) and the result is sorted by `SAT test scores` in descending order.
3. Add new empty list called `evaluations:[]` to all students using one mongoDB command. Then, add two following documents to that empty list. (2 points)
   - "eval_comment": "This student is very clever"
   - "eval_comment": "This student always submits exercises ontime"
4. Insert a new `student document` into `student collection` using journaled write concern. You can manually create the content of that `student document`. (1 point)

Bonus 2: Regular Expressions (4 Points)

Use dataset `studentsPoint.txt` for this exercise. Complete the following tasks using regular expressions.

1. Find all students with conditions: first name begins with `K` or `C`; first course is `Python`, case sensitive. (1 point)
2. Find a list of all students who have at least one of three marks less than 10 points. However, we just want to show only 2 students at the bottom of the list. (1 point)
3. Find any student(s) with conditions: both first name and last name contain `ri` somewhere in the middle, and all marks are maximum, e.g. equal to 20 points. (2 points)

Annex

1. Calculating standard deviation step by step `https://www.khanacademy.org/math/probability/data-distributions-a1/summarizing-spread-distributions/a/calculating-standard-deviation-step-by`
2. Twitter API tutorial `http://socialmedia-class.org/twittertutorial.html`
3. tweetf0rm `https://github.com/bianjiang/tweetf0rm`