TI2736-B: Assignment 7 (OPTIONAL)
Big Data Processing

Due date: 22.01.2017 (11.59pm)

Please submit your report and Scala code via Blackboard: the report should be in PDF format, the source code files should be submitted as separate files with proper code documentation. Please submit a properly formatted report: it should contain your name, student number, understandable answers (sentences, not notes or keyword lists) with a numbering that makes it clear which answer refers to which assignment question.
This assignment is optional. You can gain a maximum of 12 points across all assignments: if you successfully completed (i.e. gained 2 points each) all six assignments so far, you have already achieved the maximum possible score. If you have not achieved this, this assignment may be for you: you can gain an additional two points to make up for lost points in the earlier assignments.

In this assignment you will learn about `Spark`, how to use it and the tools and facilities it provides for big data processing. `Spark` is available in Cloudera's CDH and can be used out-of-the-box: open a terminal and type `/usr/bin/spark-shell` to start the Spark shell. Note though, that it may be slow to run (depending on your machine's performance); as an alternative (and in contrast to Hadoop) `Spark` can also be installed quite simply directly as discussed in the lecture.
Many Spark tutorials exist online. Apart from the one listed below, feel free to consult other tutorials.

1. Get acquainted with `Spark` and run some simple tasks. Visit `Spark`'s documentation[1] and watch **Screencasts 3 & 4** of the *Screencast Tutorial Videos* under the *Videos* section.
   Exercise through the assignments in the videos as they are described on `Spark`'s *Quick Start* page [2].
   – nothing to submit here –

2. In this exercise you explore the page views of Wikimedia projects.
   Download the page view statistics generated between 0-1am on Jan 1, 2016[3]. Each line, delimited by a space, contains the statistics for one Wikimedia page.
   The schema looks as follows:
   `<project_code> <page_title> <num_hits> <page_size>.`

[1] `http://spark.apache.org/documentation.html`
[2] `http://spark.apache.org/docs/latest/quick-start.html`
[3] `https://dumps.wikimedia.org/other/pagecounts-raw/2016/2016-01/pagecounts-20160101-000000.gz`

Launch the Spark shell and then create an RDD (Resilient Distributed Dataset) named `pagecounts` from the input file.
For each of the following tasks, write Scala code to solve it. Submit the code alongside your answers to the tasks.

(a) Retrieve the first k records and beautify: Use the `take()` operation of an RDD to get the first k records, with $k = 15$. The `take()` operation returns an array and Scala simply prints the array with each element separated by a comma. This is not easy to read. Make the output prettier by traversing the array to print each record on its own line.

(b) Determine the number of records the dataset has in total.

(c) Determine the record with the largest page size. If multiple records have the same size, list all of them.

(d) Determine the record with the longest page title. If multiple titles have the same length, list all of them.

(e) Pageviews per project: compute the total number of pageviews for each project (as the schema shows, the first field of each record contains the project code).

(f) Determine the number of page titles that start with the article "The". How many of those page titles are **not** part of the English project (Pages that are part of the English project have "en" as first field)?

(g) Determine the percentage of pages that have only received a single page view in this one hour of log data.

(h) Determine the number of *unique* terms appearing in the page titles. Note that in page titles, terms are delimited by "_" instead of a whitespace. You can use any number of normalization steps (e.g. lowercasing, removal of non-alphanumeric characters).

(i) Determine the most frequently occurring page title term in this dataset.