A7 - Clustering

“Gallia est omnis divisa in partes tres (...)”

— Julius Caesar, Commentaries on the Gallic War

Objectives

Performing iterative computations, computations on a graph, learning basic clustering algorithms.

Dataset

We are using a dataset based on the metadata in the Million Song Dataset (https://labrosa.ee.columbia.edu/millionsong/). The data can be found here:

- Million Songs Subset Metadata (CSV (http://violet.ele.fit.cvut.cz/~kondziu/pdpmr/MillionSongSubset.tar.gz))
- Million Songs Full Dataset Metadata (CSV (http://violet.ele.fit.cvut.cz/~kondziu/pdpmr/MillionSongDataset.tar.gz))

Use the subset for development and testing. Use the full dataset for your final submission.

Description of the dataset can be found here (https://labrosa.ee.columbia.edu/millionsong/pages/example-track-description). Watch out for dirty data!

Functional requirements

Subproblem 1: Clustering

Using Spark, perform hierarchical agglomerative clustering and k-means clustering as follows:

- **Fuzzy loudness**: cluster songs into **quiet**, **medium**, and **loud**
- **Fuzzy length**: cluster songs into **short**, **medium**, and **long**
- **Fuzzy tempo**: cluster songs into **slow**, **medium**, and **fast**
- **Fuzzy hotness**: cluster songs into **cool**, **mild**, and **hot** based on song hotness
- **Combined hotness**: cluster songs into **cool**, **mild**, and **hot** based on two dimensions: artist hotness, and song hotness

Use each method to perform each clustering task.

Compare the results, as well as the performance of the solutions.

Observe whether:

- A song's loudness, length, or tempo predict its hotness
- A song's loudness, length, tempo, or hotness predict its combined hotness

Subproblem 2: Graphs

Let's define **popularity** as the product of:

- The artist's familiarity
- The number of songs created by the artist
- The number of similar artists

Let's define **trendsetters** as the top 30 most popular artists.

Let's define **commonality** between two artists as the number of terms shared between the artists.

Let's define a **commonality graph** as an undirected graph. The nodes are artists. There is an edge between artists if they are similar. The weight on the edge is the commonality between the artists.

Cluster all artists using k-means in Spark, using the trendsetters as initial centroids and some definition of distance in the commonality graph as distance measure.

Discuss the results of the clustering and performance.

Non-functional requirements

Assignment is performed in groups of 2 (assigned by TA). Authors are clearly marked on all deliverables.

Required files:

- `src/` (directory containing the sources of the implementation)
- `README.md` (Markdown file containing the description of the implementations)
- `Makefile`
- `report.Rmd` (The report as described in the previous section)
- `report.pdf` (A PDF rendering of `report.Rmd`)
- `input/` (A directory containing sample input files; please `gzip` all text files to save space)

Provide a Makefile with the following rules:

- `build` builds all the implementations
- `run` runs the variants
- `report` generates the reports

- `all` build, run, and report

Prepare the report in R Markdown and generate a PDF.

Evaluate your solution using your a local machine and AWS.

Extra credit

It is possible to get extra credit for the assignment by:

- providing a Hadoop implementation of the Clustering subproblem and comparing the performance of the Hadoop and Spark solutions.
- providing a Hadoop implementation of the Graphs subproblem and comparing the performance of the Hadoop and Spark solutions.