Carnegie Mellon University
15-826 Multimedia Databases & Data Mining
Fall 2013 - C. Faloutsos

Project2 Description: Graph Mining using SQL

Contact TA: Alex Beutel , abeutel@cs.cmu.edu

1 Introduction - Problem description

Do we need an additional language, to do graph manipulations? Show that SQL (with a little bit of a host language, like Java), is enough, to answer all the questions we want. Given a graph of (source, destination) pairs on a disk, write the SQL queries to answer the questions of interest below, like 'which are the most important nodes', 'find the radius of each node' etc.

Assume that the input file is in edge-list form `(source-id, destination-id)`. Use *postgres*, since it has the best query optimization (mysql and sqlite3 may be used, for phase 1, only).

2 Data

There are some terrific repositories, with several overlapping datasets:

- `http://konect.uni-koblenz.de/networks/` from the KONECT project.
- `http://snap.stanford.edu/data/index.html` from the SNAP project.
- `http://www-personal.umich.edu/~mejn/netdata/` from Prof. Mark Newman.

Start with a few, small, datasets for phase 1, and we will create local mirrors here, to save you downloading effort and bandwidth. Some of the datasets are large (Gb) - please avoid them, until we set up the local mirror.

3 Introductory Papers

- The PEGASUS paper with GIM-V `http://www.cs.cmu.edu/~ukang/papers/PegasusKAIS.pdf` Discusses pagerank and connected components.
- the follow-up paper of GBASE `http://www.cs.cmu.edu/~ukang/papers/GbaseVLDBJ2012.pdf`

- 'HADI' [TKDD'11] http://www.cs.cmu.edu/~ukang/papers/HadiTKDD2010.pdf for diameter and radius estimation,
- 'HEIGEN' [PAKDD'11] for eigenvalues,
- For belief propagation, check [ICDE'11] http://www.cs.cmu.edu/~ukang/papers/HalfpICDE2011.pdf
- Also skim the rest of the papers on the pegasus project web site http://www.cs.cmu.edu/~pegasus/publications.htm

4 Tasks and Deliverables

Implement as many of the following as you can, and apply them on as many of the datasets from SNAP/KONECT/Newman as applicable. The number in **[brackets]** indicates the degree of difficulty in the scale 1 to 10 - lower is easier.

4.1 Task list

1. degree distribution (should be a few lines of SQL - good for warm-up) **[diff:1]**
2. PageRank: just a bit harder - needs several iterations (=joins) **[diff:2]**
3. (weakly) connected components **[diff:3]**
4. radius, for every node. Use the Martin-Flajolet method in 'HADI', or do sampling **[diff:5]**
5. eigenvalues/singular values: Use the method in HEIGEN. **[diff:8]**
6. Belief Propagation - use the binary, "Fast" belief propagation, from [Koutra+, PKDD'11] http://www.cs.cmu.edu/~dkoutra/papers/fabp_pkdd2011.pdf Comparable to PageRank **[diff:2]**
7. Count of Triangles **[diff:3]**. If you have successfully implemented the eigenvalues, use the "sum of cubes of eigenvalues" approach [Tsourakakis ICDM'08] http://www.math.cmu.edu/~ctsourak/tsourICDM08.pdf Otherwise, try the (award winning) sampling approach of [Seshadri, Pinar, Kolda, SDM'13] http://www.sandia.gov/~tgkolda/pubs/pubfiles/SePiKo13a-SDM13.pdf
8. Broad-spectrum graph mining (**[diff:4]**): apply your algorithms to as many of the graph datasets of SNAP and KONECT as possible; detect global patterns; spot strange-behaving datasets. For example, one global pattern could be *all datasets have radius about 7, except dataset-bla and dataset-blabla*; another one would be, e.g., that *most graphs have power-law (or lognormal, or log-logistic) degree distribution, with the same slope*.

4.2 Specific deliverables:

- Phase 1: Tasks 1, 2, 3 (in addition, of course, to the literature survey).
- Phase 2: about half of the remaining tasks; propose an additional task of your own, that you think it is worth doing (one such extra task per group member).
- Phase 3: all the tasks, including your innovative task; Especially interested in the 'broad-spectrum' mining task (Task 8).

4.3 Important: code packaging and ease-of-use

Being a mainly implementation project, your code should be

- neatly packaged,
- able to run on at least the linux-andrew machines,
- with unit-tests,
- with a (brief) user's manual
- and with an easy interface: e.g., from the unix prompt
  `graphmine my-edge-list.csv`
  should load the given input file into postgres, run all your queries, and generate all the plots for above tasks.

5 Room for innovation

There are numerous directions you could try:

- **[diff:2]** extend the above algorithms for weighted graphs; directed graphs; graphs with negative edge weights
- **[diff:8]** think, design and implement algorithms for time-evolving graphs (trying to spot, e.g., discontinuity points, and the culprits, that is, the nodes that caused the big change)
- **[diff:4]** Data layout: what is the best way to store the edges in a relational table? Try to use SQL commands only (e.g., create a clustering index, on the appropriate source-destination combination)

6 Logistics - reminders

- The detailed grading scheme is in the project guidelines `http://www.cs.cmu.edu/~christos/courses/826.F13/proj.html`
- As mentioned in the project guidelines, all projects will be done in **groups of two** (with exceptions only under special circumstances, after instructor's permission.)
- *Academic Attribution*: Whenever you use ideas, code, algorithms from someone else, *please cite this person, paper, or url*.