COMPSCI 753

Algorithms for Massive Data

Assignment 3 / Semester 2, 2022

Graph Mining

Kaiqi Zhao

Submission:

Please submit (1) an answer.pdf file reports the requested values and explanation of each task, and (2) a source code file contains detailed comments on CANVAS by 23:59 NZST, Sunday 25 September. The answer file must contain your student ID and name.

Penalty Dates:

The assignment will not be accepted after the last penalty date unless there are special circumstances (e.g., sickness with certificate). Penalties will be calculated as follows as a percentage of the mark for the assignment.

- **23:59 NZST, Sunday 25 September – No penalty**
- **23:59 NZST, Monday 26 September – 25% penalty**
- **23:59 NZST, Tuesday 27 September – 50% penalty**

1 Assignment problem (100 pts)

This assignment aims at exploring the PageRank algorithm on big real-world network data. By working on this assignment, you will learn how to implement the PageRank algorithms that we have learned in the class. Download the dataset[1] 'Google-web.txt' from the assignment page on Canvas. Each line of the file represents a directed edge from a source node to a destination node. Nodes are represented by IDs ranging from 0 to 875712.
The assignment has the following tasks:

1. **[40 pts] Task 1: Implementation of Power Iteration Algorithm.**

   (a) Implement the power iteration algorithm in matrix form **without teleport** (on slides page 20): Let stop criteria be $||\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}||_1 < 0.02$. Spider traps and dead ends are not considered in this task [30 pts].

   (b) Calculate the rank score for all the nodes and report: (1) The running time of your power iteration algorithm; (2) the number of iterations needed to stop; (3) the IDs and scores of the top-10 ranked nodes [10 pts].

   Hint: The adjacency matrix is very sparse. Consider using sparse matrix (e.g., use scipy.sparse in Python[2]). If you implement the PageRank algorithm efficiently (slides page 32), your PageRank implementation should stop within 10 seconds even with an Intel i5. If your algorithm can't stop within several minutes, you may want to check your implementation.

2. **[10 pts] Task 2: Exploiting dead-ends:** Before extending your codes to support dead ends, let's first do some analysis on your current implementation in Task 1. **Report** the leaked PageRank score in each iteration of task 1 and **explain** the phenomenon you observe from the above experiments.

3. **[50 pts] Task 3: Implementation of Teleport.**

   (a) Extend your PageRank code to handle both spider traps and dead ends using the idea of teleport. Let $\beta = 0.9$ by default and the stop criteria be $||\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}||_1 < 0.02$. [30 pts]

   (b) Run your code on the Google web data and report: (1) the running time; (2) the number of iterations needed to stop; (3) the IDs and scores of the top-10 ranked nodes [10 pts].

[1] The dataset is adapted from SNAP https://snap.stanford.edu/
[2] `https://docs.scipy.org/doc/scipy/reference/sparse.html`

(c) By varying the teleport probability β in $[1, 0.9, 0.8, 0.7, 0.6, 0.5]$, report the number of iterations needed to stop for each β. Explain your findings from this experiment [10 pts].