Web Page Quality Ranking

Kai Shen

PageRank

- PageRank ...
  - “works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.”

- Count of in-links is important; the quality of the sources of in-links are also important
  - http://upload.wikimedia.org/wikipedia/commons/6/69/PageRank-hi-res.png

- Originated at Google, though may not be heavily relied upon by Google now

Web Page Quality Ranking

- Inverted web indexes help locate matching pages of search words
  - But there are too many matches and humans can’t read all

- Both relevance and quality are important in web search

- What is a high-quality web page?

- How to identify a high-quality web page?
  - Hard to spam

- Related to identifying high-quality scientific publications
  - But much bigger dataset

PageRank

- PageRank scores each page
  - the score is the converged probability for a random web surfer to arrive at the page

- Scores transfer through links (random click by the surfer):
  - If pages B, C are the pages that link to A, then
    \[ PR(A) = PR(B)/L(B) + PR(C)/L(C) \]

- Damping factor (for stability):
  - The surfer has a tendency to stay at where he/she is
    \[ PR(A) = \frac{(1-d)}{N} + d(PR(B)/L(B) + PR(C)/L(C)) \]

- A linear system of equations (N variables, N linear equations)
How to compute it?

- How to solve a linear system of equations?
- Big data \( \Rightarrow \) large matrix (50 billion \( \times \) 50 billion)
  - Parallelizing it is essential

Gaussian Elimination

- Simplification – ignore final solving step and pivoting
- Reduce an equation matrix into an equivalent upper-diagonal

\[
\begin{align*}
A & \xrightarrow{pA_1 + A_2} X \\
pA_1 + A_2 & \quad r_1 + p^*r_1 \\
& \quad X = R
\end{align*}
\]

for \( c=1 \) to \( N \)
for \( r=c+1 \) to \( N \)
zero out \( A_{c,r} \) by adding \( p^*A_{c,r} \) to \( A_{c,r} \)

Parallelizing Gaussian Elimination

- How to parallelize? Dependencies:
  - Outer loop instances (e.g., \( c=2 \) depends on \( c=1 \))?
  - Inner loop instances (e.g., \( r=3 \) depends on \( r=2 \))?
  - MapReduce? Does it scale?
  - Traditional parallel programming is fine, but hard to program correctly, robustly with high performance and scalability

Iterative Solver

- \( PR(A) = (1-d)/N + d(PR(B)/L(B) + PR(C)/L(C)) \)

- Iterative solver:
  - Start from an initial PR vector (1/N for each page)
  - Compute a new PR vector by the above linear transformation
  - Keep repeating the linear transformation until the PR converges (very small change after further linear transformation)
    - Even with the large, complex web graph, often converging after dozens of iterations
  - Each iteration is a matrix-vector multiplication, so the whole computation is a series of matrix-vector multiplication
Parallelizing the Iterative Solver

- The whole computation is a series of matrix-vector multiplication
- How to parallelize?
  - MapReduce?
  - Does it scale?

Load Distribution

- One row (or a few rows) per task:

```
  A  B  C
  X  =  X
```
- Or one block per task:

```
  A  B  C
  X  =  X
```

Load Balancing

- Matrix vector multiplication
  - Load balancing by distributing equal-size blocks of A to each map task
  - But matrix is extremely sparse \( \Rightarrow \) equal-size blocks doesn't mean equal amount of work
  - More map tasks than the number of nodes; rely on dynamic load balancing in MapReduce

Topic-Sensitive PageRank

- Web page quality ranking is sometimes affected by interest on a particular topic
- Original PageRank score
  - Start from a PR vector (1/N for each page)
  - Compute a new PR vector by the link-based linear transformation
    - \( PR(A) = (1-d)/N + d(PR(B)/L(B) + PR(C)/L(C)) \)
  - Keep repeating the above until the PR converges
- Topic-sensitive PageRank:
  - Start from a PR vector (1/n for each topic-matching page, 0 otherwise), n is the number of topic-matching page
  - In iterative computation, \( PR(A) \)'s damping factor is (1-d)/n if it is a topic-matching page, 0 otherwise
  - Effect: the random surfer always starts from a topic-matching page, and has an extra staying probability if it is on such a page
**Topic-Sensitive PageRank**

- A great way to return searches online searches
- Can it be done that quickly?

**Hubs and Authorities**

- Fundamentally wrong to think that web pages are of one kind
  - Existence of hubs, pages with many links to good pages (even though they are not linked by others very well)
  - In original PageRank computation, these pages have very low scores and their out-links are even less effective (since they have many links)
- HITS [Kleinberg 1999]
  - Each page has two scores—authority score is like the traditional quality score; hub score indicates the strength of the page linking to high-quality page
  - \[ a = Lh \]
  - \[ h = L^T a \]
  - Solved iteratively, can use MapReduce

**Disclaimer**

- Preparation of this class was helped by materials in the online book “Mining of Massive Datasets” by Rajaraman, Leskovec, and Ullman.