Fast Incremental and Personalized PageRank

Paper Presentation

Ioannis Chios, Georgios Kyziridis

November 30, 2018

Social Network Analysis for Computer Scientists

Leiden University

Table of Contents

Background

PageRank

- A variant of eigenvector centrality
- The key idea behind Google Search
- Measures relative importance of web pages
- Apply citation analysis to the web's hypertextual structure
- Importance of a web page based on the number of backlinks

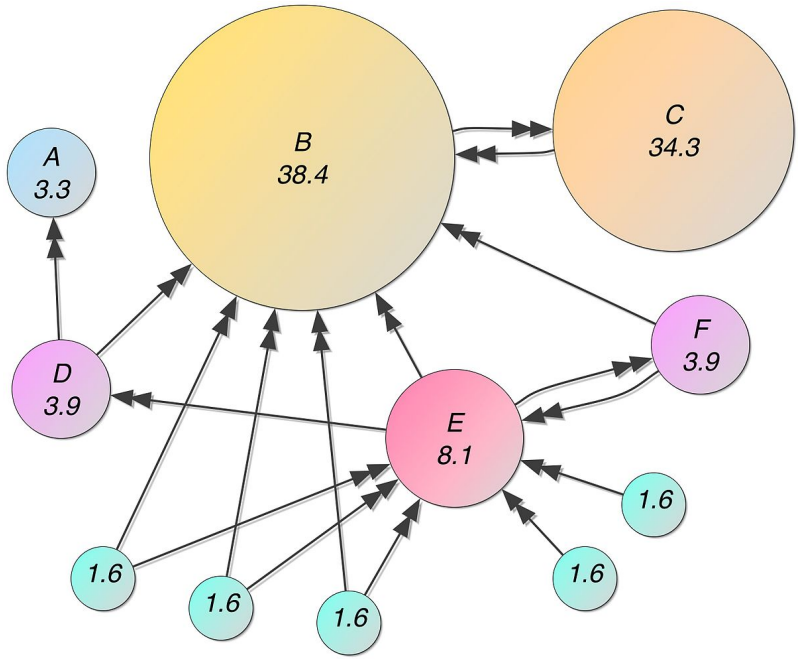


Image taken from Wikipedia

Personalized PageRank

- Apply PageRank to a specific node
- Different rankings for each node
- Applications
  - Personalized search results
  - Personalized Recommendation Systems

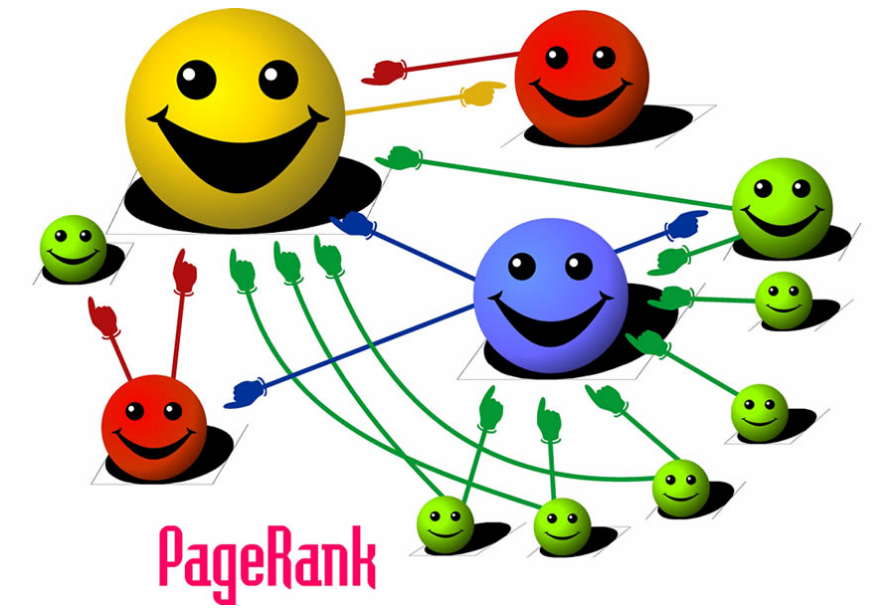


Image taken from Wikipedia

Twitter Network

- Very large dynamic social network
- Network topology changes over time
- Nodes and edges are leaving and coming
- High levels of uncertainty
- Uses personalized PageRank for WTF

Problem Statement

Problem Statement

- **Research Question:** Can we build a recommender system for Dynamic Social Networks?

- **Possible Solution:** Find a way to keep the personalized PageRank values updated all the time and use them for recommendation.

Algorithms

Approximating PageRank

Goal: Design an algorithm to update PageRank incrementally (i.e. upon an edge arrival). Reduce algorithm's complexity using approximations.

Idea: Instead of calculating the PageRank values exhaustively, use Monte Carlo method and approximate them.

Approximating PageRank

Approach: R random walks for each node in the network. Store all walks in a database. Then approximate the PageRank value of a node with:

$$\pi_v = \frac{X_v}{nR/\epsilon}$$

π_v: PageRank value for v node

X_v: Total number of node v occurrences in all walks

n: Total number of network nodes

ϵ: Transmission probability

Updating Approximations

- Network evolves through time
- PageRank values of the nodes have to be updated

Expected amount of work for keeping approximations updated through time over m edge arrivals:

$$\frac{nR}{\epsilon^2} \sum_{t=1}^{m} \frac{1}{t} = \frac{nR}{\epsilon^2} H_m \leq \frac{nR}{\epsilon^2} \ln m$$

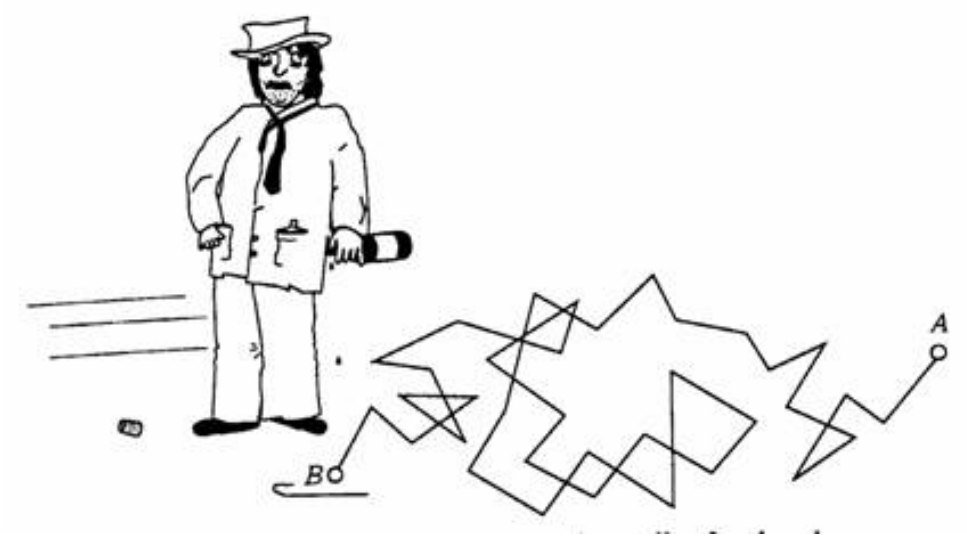
H_m: The m^{th} harmonic number

t: Time-stamp

Approximating Personalized PageRank

Idea: Recommendation system according to Personalized-PageRank.

Approach: Reuse the R stored random walks and perform the personalized PageRank random walk for node w and walk length L.



Approximating Personalized PageRank

Algorithm 1 Personalized PageRank Walk

Input: Source node w, required walk length L
Output: A personalized PageRank walk P_w
Start the walk at w : $P_w \leftarrow [w]$
while length$(P_w) < L$ **do**
    $u \leftarrow$ last node in P_w
    Generate a uniformly random number $\beta \in [0, 1]$
    if $\beta < \epsilon$ **then**
        Reset the walk to w : $P_w \leftarrow P_w \cdot \text{append}(w)$
    else
        if u has unused walk segment Q in memory **then**
            Add Q to the end of P_w : $P_w \leftarrow P_w \cdot \text{append}(Q)$
            Reset the walk to w : $P_w \leftarrow P_w \cdot \text{append}(w)$
        else
            if u was previously fetched **then**
                Take a random edge (u, v) out of u
                Add v to the end of P_w : $P_w \leftarrow P_w \cdot \text{append}(v)$
            else
                Do a fetch at u

Personalized PageRank Values

Let $\vec{\pi}$ to be the vector of Personalized PageRank's scores of interest.
Assume that $\vec{\pi}$ follows a power law model.
If π_j is the J^{th} largest element in $\vec{\pi}$ then for some $\alpha \in (0, 1)$ approximate the PageRank values:

$$\pi_j = \frac{(1-\alpha)j^{-\alpha}}{n^{1-\alpha}}$$

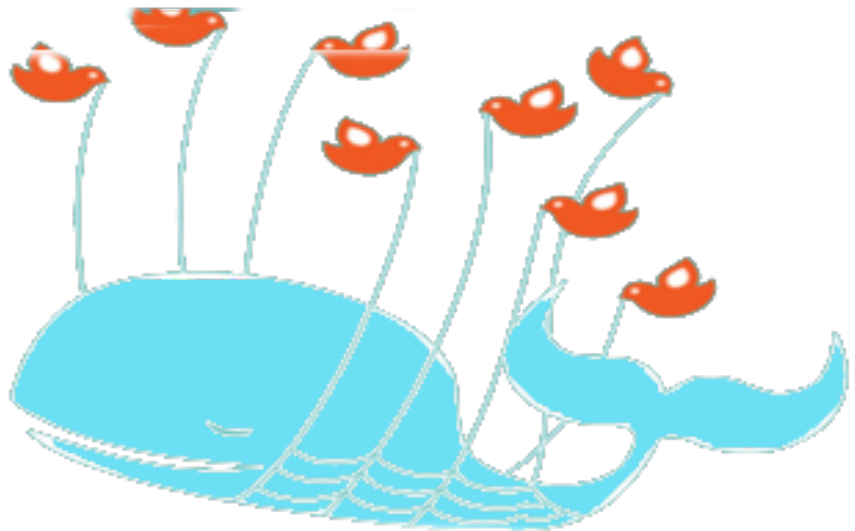
$$\pi_j \propto j^{-\alpha}$$

π_j is a proportion of the j^{th} largest element, $j^{-\alpha}$

Experiments

Experimental Setup

- **Twitter Network** consists of directed edges
- Data access through **Twitter's Social Store** called **FlockDB**
- **FlockDB** is stored in distributed memory
- Personalized experiments on 100 randon users
- Reasonable amount of user's friends e.g. 20-30

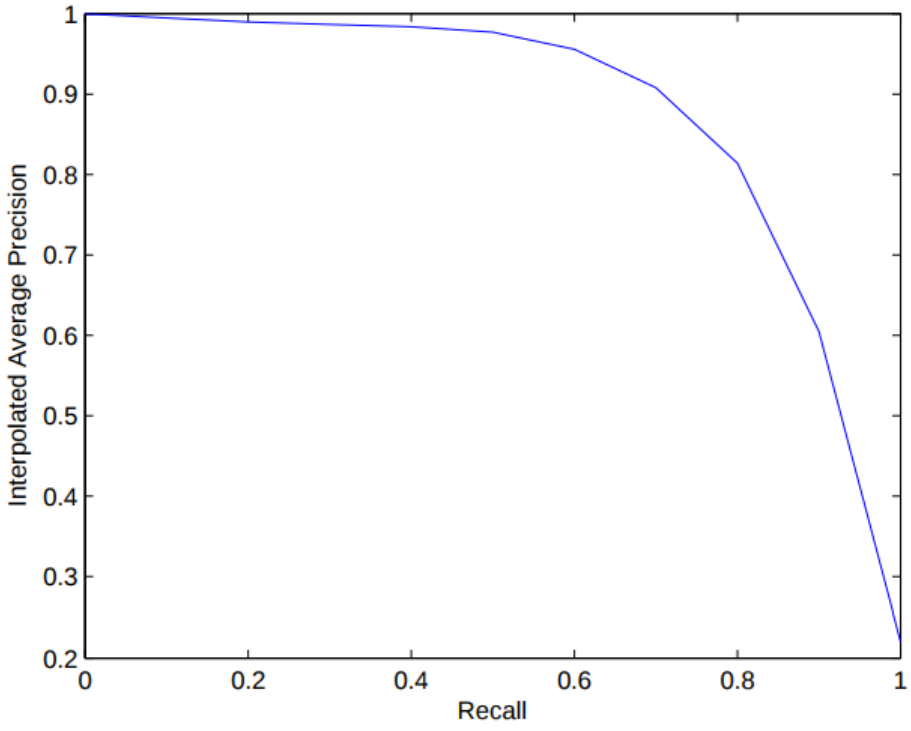


Experimental Setup (Cont.)

- **True**
  - An infinite random walk would accurately give the top **k** nodes
  - For each user do a personalized walk with 50.000 steps
  - Consider the top 100 most visited nodes as true
- **Retrieved**
  - For each user do a personalized walk with 5.000 steps
  - Retrieve the top 1000 most visited nodes
- **Metrics**
  - Interpolated average precision
  - Number of fetches

Results

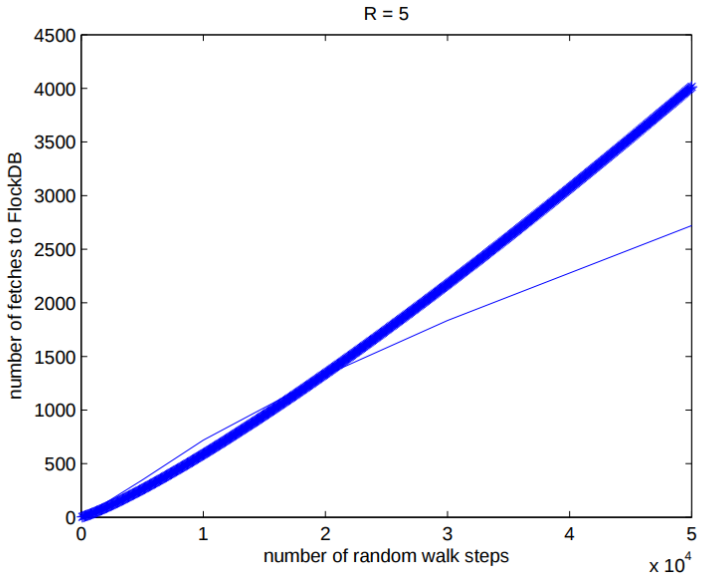
Interpolated Average Precision



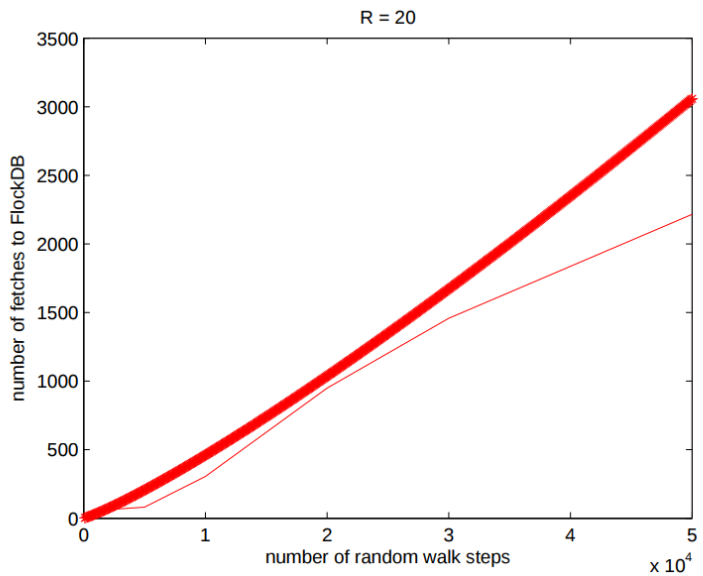
11 point interpolated average precision for top 1000 results

- Shows that even short random walks are enough to find the top nodes.

Number of Fetches



(a) R=5



(b) R=20

- Thin line shows the average number of fetches with R walk segments per node.
- Thick line shows the theoretical upperbound on the number of fetches.

Our Project

General Ideas

Goal: Test the above algorithms on different datasets. Evaluate the results according to time and performance.
Intuition: The approximated version of PageRank probably is not capable of handling sparse networks.

- Implement PageRank and approximated PageRank efficiently
- Test both on different kinds of datasets
- Compare the results
- Evaluate both algorithms regarding time and results outcome
- Explore the intuition on sparse networks

Possible Evaluation Metrics

True: Personalized PageRank output from `NetworkX`.
Predicted: Approximated personalized PageRank outcome.

Retrieval Metrics

- Accuracy
- Interpolated Precision
- $F1_{measure}$
- Jaccard Similarity

Continuous Metrics

- Euclidean Distance
- Mean Square Error (`MSE`)
- Mean Absolute Error (`MAE`)
- Root Mean Square Error (`RMSE`)

Thank you for your attention!

Questions?

Demo