Recommender System Notes

Tiangang Chen, Mengqiao Zhang

April 21, 2017

1 Intro

In our project, we implemented a naive version of Twitter's preliminary recommender system [Gup+13] for recommending who to follow. We also implemented a very simple cosine-distance based content-filtering algorithm as a complement.

These two algorithms are written in Python scripts, intended to be executed by the OS scheduler on fixed time interval. So the recommendation is not reactive immediately to user input.

This document serves to record the details of our understanding and implementation, primarily as our self-reference.

2 Content Filtering

We collect the following raw personal information: gender, major, and tags. The tags are predefined predicates of a person's hobbies/traits that the user may select as he/she sees fit.

We model the tags as a boolean vector with dimension of number of tags available.

The rank/similarity score between two users u and v is calculated as follows:

$$\frac{u \cdot v}{\|u\|_2 \|v\|_2}$$

Which is readily available via `1 - scipy.spatial.distance.cosine(u,v)`.

3 Collaborative Filtering

The following illustrates an overview of the algorithm described in [Gup+13].

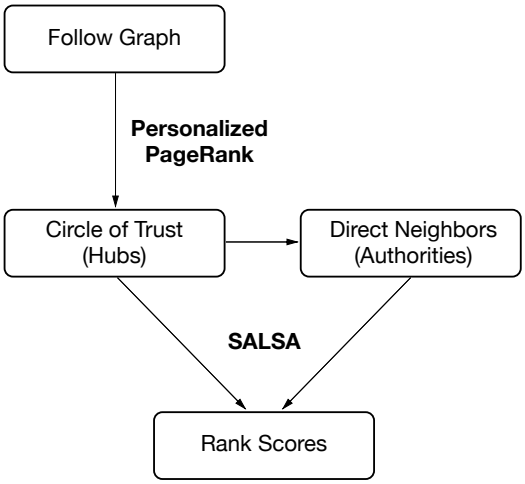

Figure 1: overview of the algorithm

3.1 PageRank

The setup for PageRank [Pag+99] is as follows: let u be a web page, F_u be the set of pages that u points to, and B_u be the set of pages that points to u. Let $N_u = |F_u|$, which is the number of links originating from u. c is the factor for normalization.

The simplified PageRank:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}$$

$$R = cAR$$

Where A is the matrix representation of the graph. $A_{u,v} = \frac{1}{N_u}$ if there is an edge from u to v, otherwise 0.

Simplified PageRank suffers from "rank sink" when we encounter situation similar to: two pages pointing only to each other, then some other page

points to one of them. Then the rank will accumulate in the two pages but never distribute.

The improved PageRank includes a source of rank $E(u)$:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} + cE(u)$$

$$R = c(AR + E)$$

We aim to maximize c and keep $\|R\|_1 = 1$. Because $\|R\|_1 = 1$, we can rewrite $R = c(A + E \cdot 1)R$ where 1 is a vector of all ones. The introduction of vector E remedies the rank sink. Generally, we can populate E with uniform value. By giving E some bias we can obtain a "personalized" ranking focusing on a particular webpage.

We simply implement PageRank as follows, identical to the original [Pag+99] algorithm:

Data: R the rank vector;
A matrix representation of the graph;
E the initialization vector;
ϵ convergence limit;
initialize: $R_0 \leftarrow E$;
while $\delta > \epsilon$ **do**
    $R_{i+1} \leftarrow AR_i$;
    $d \leftarrow \|R_i\|_1 - \|R_{i+1}\|_1$;
    $R_{i+1} \leftarrow R_{i+1} + dE$;
    $\delta \leftarrow \|R_{i+1} - R_i\|_1$;
end

Algorithm 1: PageRank

3.2 SALSA

SALSA [LM01] operates on a bipartite graph, traversing while assigning scores to nodes on both sides. The two sets of nodes are called hubs and authorities. In our case, the set of hub nodes consist of top results from

personalized PageRank for a user, called the "circle of trust", and the set of authority nodes consist of direct (one degree) neighbors of those hub nodes.

We set up SALSA as follows: initially we have a collection (set) of pages, denoted by $\mathcal{C}$, edges are directed in this collection. Now we convert $\mathcal{C}$ to an undirected bipartite $G = (V_h, V_a, E)$, where

$$V_h = \{s_h | s \in \mathcal{C},\ \text{outDegree(s)} > 0\}$$
$$V_a = \{s_a | s \in \mathcal{C},\ \text{inDegree(s)} > 0\}$$
$$E = \{(s_h, r_a) | s \to r\ in\ \mathcal{C}\}$$

So a page with both nonzero in-degree and out-degree will appear on both sides of the bipartite. Our $\mathcal{C}$ consists of the previously mentioned "circle of trust" and the nodes that they point to, which is naturally a bipartite.

Now we perform two distinct random walks on G. The two walks start from opposite side of the bipartite, traversing two edges in each iteration. So they will end up at the same side where they started. This process is represented as two Markov chains. The transition matrices are defined as follows:

$$h_{i,j} = \sum_{\{k|(i_h,k_a),(j_h,k_a)\in G\}} \frac{1}{deg(i_h)} \cdot \frac{1}{deg(k_a)}$$
$$a_{i,j} = \sum_{\{k|(k_h,i_a),(k_h,j_a)\in G\}} \frac{1}{deg(i_a)} \cdot \frac{1}{deg(k_h)}$$

Our goal is to find the pricipal eigenvectors of the hub and autority matrix, which are eigenvectors with eigenvalues of the largest magnitude. It has been argued in [LM01] that such eigenvalue/eigenvector exists and is unique (has multiplicity of one). The values within are the assigned scores for the hub and authority nodes. Higher values stand for higher probability of the random walk ends up at the nodes.

References

[Gup+13] Pankaj Gupta et al. "WTF: The Who to Follow Service at Twitter". In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: ACM, 2013, pp. 505–514. ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488433. URL: http://doi.acm.org/10.1145/2488388.2488433.

[LM01] R. Lempel and S. Moran. "SALSA: The Stochastic Approach for Link-structure Analysis". In: *ACM Trans. Inf. Syst.* 19.2 (Apr. 2001), pp. 131–160. ISSN: 1046-8188. DOI: 10.1145/382979.383041. URL: http://doi.acm.org/10.1145/382979.383041.

[Pag+99] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, Nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/.