Project Two: Finding Influential Users of Online Community with PageRank

Yu-Qiao Xian
Student ID: 15349112
School of Electronics and Information Technology
Sun Yat-Sen University
Guangzhou, China
ericxian1997@gmail.com

Abstract

This is a project report for assignments in Data Mining course instructed by Prof. Wei-Shi Zheng. PageRank has been widely applied to measure the importance of website pages by web search engine (e.g. Google). Users interactions in online communities such as Bulletin Board System (BBS), open source community (e.g. GitHub) and social network (e.g. Weibo) have similar structure with links of websites on Internet. In this project we model interactions between users of online community with directed unweighted graph and use a modified algorithm from PageRank, VoteRank, to find influential users in the community. We deal with the practical problem of ranking users of Wikipedia according to the voting data. Our improved method can prevent users from deliberately raising their ranking by voting each other.

1 Introduction

An online community or also called an internet community is a virtual community whose members interact with each other primarily via the Internet. There exist many forms of online communities, such as Bulletin Board System (e.g. Tianya-China), open source community (e.g. GitHub) and social network (e.g. Weibo). The interactions between users of online communities are also various, such as comment / forward on Weibo, fork / star on GitHub, share / subscribe on YouTube and so on. In our project, we analyze the data of Wikipedia, one of the most famous and useful online communities on the Internet.

Wikipedia [1] is a multilingual, web-based, free-content encyclopedia project supported by the Wikimedia Foundation and based on a model of openly editable content. Wikipedia is the largest and most popular general reference work on the Internet and is named as one of the most popular websites. The project is owned by the Wikimedia Foundation, a non-profit organization. Wikipedia is a free encyclopedia written collaboratively by volunteers around the world. A small part of Wikipedia contributors are administrators, who are users with access to additional technical features that aid in maintenance. In order for a user to become an administrator a Request for administrator is issued and the Wikipedia community via a public discussion or a vote decides who to promote to administrator.

In the graph model of online community, nodes stand for users, while edges stand for their interactions. Despite the diversity of interactions, we can model online communities with directed unweighted graph. This is mainly due to two common points of different kinds of interactions: 1) Users in interaction can be divided into active side(s) and passive side(s), which is the reason why we use directed edge. 2) Interactions are difficult to be quantified,

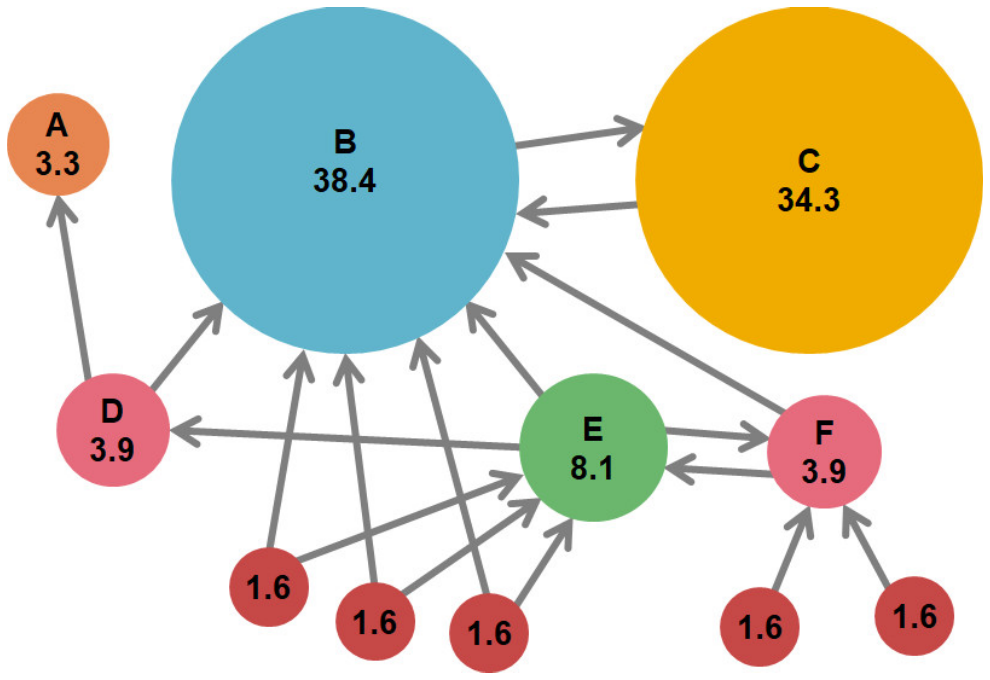

Figure 1: Sketch of PageRank example

which is the reason why we use unweighted edge. For example, on Wikipedia, the requestors for administrator can be regarded as passive sides in the vote, while those who vote for the candidates can be regarded as active sides.

PageRank [2] is an algorithm used by Google Search to rank websites in their search engine results. It was firstly proposed by co-founders of Google Inc., Larry Page and Sergey Brin in 1998. PageRank algorithm assigns a rank to each website in the database, which measures the importance of this website. Usually, an important website should be cited by many other websites, which can be discovered by analyze the links in the HTML files. If another important website has links to this website, then the link from this important website should contribute more to the rank of website than links from other trivial websites. That is the main idea of PageRank. PageRank also use random teleports to deal with "leaking out importance" problem caused by spider traps and dead-ends [3].

Our method is adapted from PageRank and deals with the problem of measuring the influence of users in online community. Similar to measuring the importance of website, an influential user in online community should has many interactions with other users, just like links on the Internet. However, a user can pretend to be influential by deliberately having interactions with some prearranged users. For example, in our task of measuring the influence of users on Wikipedia with the voting data, some users may agree on voting each other regardless of whether they really have enough contributions to the communities. Although such behavior is allowed and reasonable to some degree, it will bring our measurement more unfairness. In order to regularizing this problem, we propose a new method, VoteRank, to look down upon mutual voting by weighted tax rate. The more a user's tickets are from another user he also votes for, the more punishment on his rank he will receive. We will describe our method in detail in section 3.

In our experiments we implement both original PageRank and our adapted method VoteRank, our code is public on GitHub: https://github.com/ericxian1997/VoteRank

In the rest of this report, we present a brief review of PageRank algorithm in section 2, a description of our adapted method, VoteRank, in section 3, our implementation detail and experimental results in section 4 and finally our conclusions in section 5.

2 PageRank

2.1 Formulation

PageRank is a function that assigns a real number to each page in the Web (or at least to that portion of the Web that has been crawled and its links discovered). The intent is that the higher the PageRank of a page, the more "important" it is. In PageRank, each website is represented by a node in a directed graph, and links between websites are represented as directed edges. PageRank assume that users behave like a random surfer on any websites. At each step, random surfer has two options: following a link at random with probability β, or jumping to some random page with probability $1-\beta$.

The rank of node j can be computed by the following PageRank equation [2] (see Eq. (1)):

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N} \qquad (1)$$

In Eq. (1), constant β is called as the tax rate, which describe the probability for a user to go to another website following a link on one website. Usually β is equal to 0.8 ~ 0.9. N is the total number of websites in database, and d_i is the outdegree of node i.

The Google Matrix $\boldsymbol{A}$ is defined by the following equation (see Eq. (2)):

$$\boldsymbol{A} = \beta \boldsymbol{M} + (1-\beta)\left[\frac{1}{N}\right]_{N\times N} \qquad (2)$$

The rank vector $\boldsymbol{r}$ can be solved by the following iterative update equation (see Eq. (3)) with random initialization:

$$\boldsymbol{r}_{new} = \beta \boldsymbol{M} \cdot \boldsymbol{r}_{old} + \left[\frac{1-\beta}{N}\right]_N \qquad (3)$$

2.2 Efficient Implementation

As the number of websites on the Internet is massive and only few of them are linked to each other, the transition matrix $\boldsymbol{M}$ is a high-dimensional sparse matrix. Representing this matrix in regular way need huge storage space. As there are billions of websites on the Internet, storing the transition matrix in a disk is impossible. In practice, we usually represent this matrix with adjacency table. Adjacency table only lists its nonzero entries and record their source nodes and destination nodes. When computing rank vector $\boldsymbol{r}$ with Eq. (1), there is no longer to traverse the whole column of matrix $\boldsymbol{M}$ to look up non-zero entries. We can directly find links connected to the node according to the adjacency table. Therefore, representing matrix $\boldsymbol{M}$ with adjacency table not only save storage space, but also reduce computing time of PageRank algorithm.

Figure 2 shows examples of two different ways to represent the following matrix with adjacency table. Both methods can be used in the implementation of PageRank.

$$\boldsymbol{M} = \begin{bmatrix} 0 & 1/2 & 1 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

Source	Degree	Destinations
A	*3*	*B, C, D*
B	*2*	*A, D*
C	*1*	*A*
D	*2*	*B, C*

Destination	Degree	Sources
A	*2*	*B, C*
B	*2*	*A, D*
C	*2*	*A, D*
D	*1*	*A*

Figure 2: Examples of sparse representation of matrix: out-degree adjacency table (left) and in-degree adjacency table (right).

3 VoteRank

Our adapted algorithm, VoteRank, introduce a penalty factor to punish the behavior of mutual voting in Wikipedia voting, which presents as bidirectional edges in the graph. Regularized by the penalty factor, the tax rates of different notes are diverse in VoteRank. Nodes with higher ratio of bidirectional edges will have higher penalty factor, which results in lower tax rate and lower contributions of edges to their ranks. Our method is easy to implement and has same magnitude of computational cost with PageRank. For convenience of computation, our method treats two kinds of votes (see Figure 3) equally but treat users with different penalties.

3.1 Formulation

Given an original tax rate constant β (usually between 0.8 and 0.9), we define a penalty factor for i^{th} node in the graph by α_i, and the regularized tax rate for each node by β_i, which is computed by the following equations (Eq. (4) and Eq. (5)):

$$\alpha_i = \frac{number\ \ of\ \ bidirectional\ \ edges\ \ on\ \ node\ \ i}{indegree\ \ of\ \ \ node\ \ i} \qquad (4)$$

$$\beta_i = \beta\ \ (1 - \alpha_i) \qquad (5)$$

Then the PageRank equation (Eq. (1)) becomes:

$$r_j = \sum_{i \to j} \beta_j \frac{r_i}{d_i} + (1 - \beta_j)\frac{1}{N} \qquad (7)$$

Now, for different nodes in the graph, the tax rates are different. Therefore, the regularized tax rate of the graph become a vector $\boldsymbol{\beta}$ in Eq. (8) (9), rather than a constant β in Eq. (1) (2) (3). The Google Matrix in Eq. (2) and the iterative update equation in Eq. (3) become the following:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{\beta}^T \\ \boldsymbol{\beta}^T \\ \vdots \\ \boldsymbol{\beta}^T \end{bmatrix}_{N \times N} \circ \boldsymbol{M} + (\boldsymbol{1} - \boldsymbol{\beta}) \circ \boldsymbol{r} \cdot \left[\frac{1}{N}\right]_{1 \times N} \qquad (8)$$

$$\boldsymbol{r}_{new} = \boldsymbol{\beta} \circ (\boldsymbol{M} \cdot \boldsymbol{r}_{old}) + \frac{(\boldsymbol{1} - \boldsymbol{\beta})^T \cdot \boldsymbol{r}}{N} \qquad (9)$$

We use $\circ$ to represent element-wise multiplication. Our method regularizes the rank of nodes with bidirectional edges, a part of rank is extracted from the first term of Eq. (7) and joined into the second term by the penalty factor α. Nodes with more bidirectional edges benefit from their edges with smaller percentage, which is our way to punish the behavior of exchanging votes.

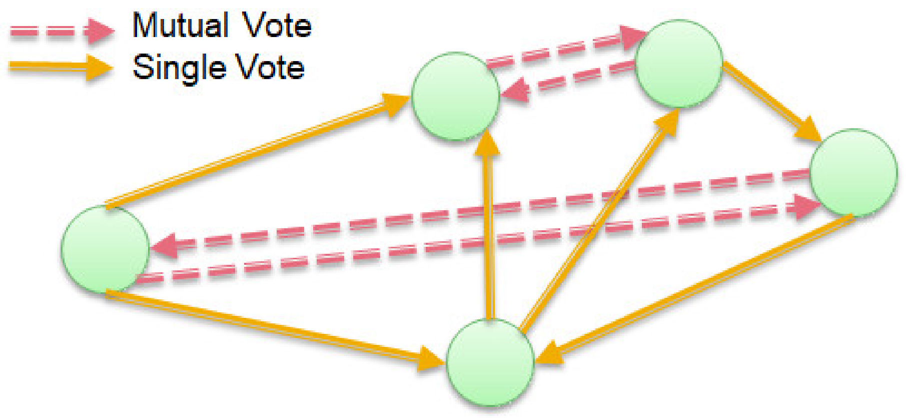

Figure 3: Two kinds of vote on Wikipedia

3.2 Complexity Analysis

Compared to PageRank, VoteRank need to precompute the penalty factor of each node and their regularized tax rate. During the iterations, the regularized tax rate vector remains constant, which is the same with PageRank. As most of time consumption is in the step of solving rank vector $\boldsymbol{r}$ by iterating, the time of precomputing the penalty can be neglected. Therefore, VoteRank has almost the same time complexity with PageRank.

In PageRank, we only need to use one form of adjacency table to represent the transition matrix $\boldsymbol{M}$. However, in VoteRank, for the convenience of computing the penalty factor, we use both in-degree and out-degree adjacency table in our implementation, which is twice of PageRank. In iterating step, one of these adjacency table can be deleted and more space can be released.

4 Implementation and Experimental Results

We implement original version of PageRank and our adapted version, VoteRank, with Python 3.6 and test them on data set **Wiki-Vote**.

4.1 Dataset Source

Our experiment uses data download from the website of Stanford Network Analysis Project (SNAP) [6]. It is a general-purpose network analysis and graph mining library. It is written in C++ and easily scales to massive networks with hundreds of millions of nodes, and billions of edges. It efficiently manipulates large graphs, calculates structural properties, generates regular and random graphs, and supports attributes on nodes and edges. We use the data **Wiki-Vote** [4] [5], which using the latest complete dump of Wikipedia page edit history (from January 3 2008) and extracted from all administrator elections and vote history data. This gave us 2,794 elections with 103,663 total votes and 7,066 users participating in the elections (either casting a vote or being voted on). Out of these 1,235 elections resulted in a successful promotion, while 1,559 elections did not result in the promotion. About half of the votes in the dataset are by existing admins, while the other half comes from ordinary Wikipedia users.

Nodes	Edges	Avg. Clustering coefficient	Num. Triangles	Diameter
7115	103689	0.1409	608389	7

Table 1: Dataset statistics of wiki-Vote

4.2 Implementation Details

We use Python 3.6 to implement PageRank and VoteRank. As the data is extracted from the real database of Wikipedia, users are anonymous in the data and represented by user IDs. The IDs are not continuous (range from 3 to 8276), so we use a hash table to map these IDs to serial numbers (from 1 to 7115).

For PageRank, we use in-degree adjacency list to represent the sparse transition matrix, which is more convenient for computing the rank of each node during iterations. For VoteRank, we use both in-degree and out degree adjacency list because it need to precompute penalty factor using in-degrees and numbers of bidirectional edges.

Before iteration, we set a maximum error constant ε, when the differences between each value in the current rank vector and the previous rank vector are all not greater than ε, the iteration will stop. Then we will finish the algorithm and use the current rank vector as our final result. Typical values of ε are 1e-6, 1e-8 and 1e-10.

The original tax rate is set to be 0.8 in both PageRank and VoteRank implementation.

4.3 Results

We sort the users by their ranks computes by PageRank and VoteRank algorithms. Table 2 shows the most 15 influential users on Wikipedia using ranks computed by PageRank. It indicates that users with high ranks usually have large indegrees, which means the receive many votes from other users. Their own votes do not contribute to their ranks, as outdegrees of many top-ranked users are zero but the they still obtain high ranking. However, we can not just judge the ranking of a user only by indegree of its node. For example, user 2470 (rank 5) have fewer votes than user 2398 (rank 7), but he still rank higher than user 2398. This is probably because voters for user 2398 have high ranks and they contribute more to the rank of user 2398. This is exactly the idea of PageRank, which can avoid user raising their ranking by voting himself using multiple accounts. Because other accounts registered by the same users may not have high ranks and their votes do not matter much. Only those who are active in the online community have significant effects on voting others.

Ranking	**User ID**	**Rank**	**Outdegree**	**Indegree**
1	4037	4.52E-03	15	457
2	15	3.54E-03	50	361
3	6634	3.26E-03	3	203
4	2625	3.11E-03	0	331
5	2470	2.53E-03	0	149
6	2237	2.48E-03	241	181
7	2398	2.45E-03	62	340
8	4191	2.17E-03	20	259
9	5254	2.07E-03	33	265
10	7553	2.05E-03	0	190
11	1186	2.03E-03	0	193
12	2328	1.95E-03	215	266
13	7620	1.84E-03	0	208
14	1297	1.84E-03	76	309
15	4335	1.82E-03	32	228

Table 2: Top-15 most influential users on Wikipedia with PageRank

Table 3: Top-15 most influential users on Wikipedia with VoteRank

Ranking	User ID	Rank	Outdegree	Indegree	Punishment	Delta Ranking
1	4037	4.66E-03	15	457	0.01	0
2	15	3.55E-03	50	361	0.02	0
3	6634	3.27E-03	3	203	0.01	0
4	2625	3.12E-03	0	331	0	0
5	2470	2.70E-03	0	149	0	0
6	2398	2.31E-03	62	340	0.05	↑ 1
7	4191	2.14E-03	20	259	0.02	↑ 1
8	1186	2.14E-03	0	193	0	↑ 3
9	2237	2.11E-03	241	181	0.19	↓ 3
10	7553	2.08E-03	0	190	0	0
11	5254	2.05E-03	33	265	0.03	↓ 2
12	7620	1.87E-03	0	208	0	↑ 1
13	4875	1.85E-03	0	181	0	↑ 3
14	8293	1.77E-03	0	164	0	↑ 8
15	7632	1.77E-03	0	178	0	↑ 2

Table 3 shows the ranking according to the VoteRank scores of users. We can see variations between table 2 and table 3. “Punishment” in table 3 stands for the penalty factor of each users, which is the proportion of mutual votes in their total votes. We can see the ranking drops of users with punishment (e.g. user 2237 and user 5254), compared to their ranking in table 2. Those users who have no mutual votes obtain ranking raises (e.g. users ranked 12-15). According to the statistics in table 3, we can find some interesting behavior patterns of users in Wikipedia. For example, we can see user 2237, who has a large out-degree (241), and also receives harsh punishment. We can guess that he votes a lot of other users despite their contributions in order to acquire more votes for himself. For the sake of fairness, we do not encourage such a behavior, so VoteRank is effective to strike users with such behavior.

We can also see the obvious effect of VoteRank in table 4. It shows top 10 users with highest rate of mutual votes. These users receive more than 38 percent of their votes from users they also vote. We regard such behavior as cheating so that these users suffer cruel penalty by punishment mechanism of VoteRank. We can see that their delta rankings (compared to their PageRank ranking) reach hundreds, which are much different from their rankings or PageRank. Similarly, we can find several users with large out-degree in table 4, such as user 1487, 166 and 693. These guys are probably deliberately votes others for the sake of themselves. Some other users (e.g. user 6 and user 7) have non-zero penalty but still have slightly raised their rankings, we consider that this is caused by calculation errors during iterations, which also happened in PageRank implementation.

Comparatively speaking, PageRank algorithm cannot discover and regularize the behavior of mutual voting, it just simply computes the ranking by the idea of random suffer. VoteRank measures a penalty by computing the rate of mutual votes using in-degree and out-degree, as well as the number of bidirectional edges. In practical, we can use the penalty factor defined in Eq. (4) as an extra index besides the VoteRank and consider additional punishments on those users according to our realistic demand. From this aspect, VoteRank also provides us another perspective to describe the patterns of users.

Table 4: Top-10 ballot riggers detected by VoteRank

Ranking	User ID	Rank	Outdegree	Indegree	Punishment	Delta Ranking
1785	1029	1.17E-04	135	12	0.5	↓ 756
2138	8227	8.17E-05	1	2	0.5	↓ 219
1609	311	1.41E-04	321	47	0.47	↓ 400
2026	707	9.20E-05	123	28	0.43	↓ 241
1741	1236	1.22E-04	103	34	0.41	↓ 307
1201	6	2.12E-04	302	20	0.4	↓ 370
1487	5802	1.57E-04	385	41	0.39	↓ 298
2019	4355	9.24E-05	50	18	0.39	↓ 227
166	1549	7.00E-04	587	245	0.38	↓ 109
693	322	3.38E-04	599	144	0.38	↓ 371

4.4 Time Complexity

Our Python code for PageRank and VoteRank are implemented under the same settings. We compare the time complexity if these two methods on wiki-Vote under same situations. The result in table 5 shows that time cost of VoteRank is very close to PageRank, but just a little slower. This is because VoteRank needs extra time to compute the penalty factor and the regularized tax rate.

The experiments are conducted on a desktop PC with an Intel Xeon E3-1231v3 Octa-core 3.4GHz CPU and 16G RAM. The unit of run time is seconds.

Method	Maximum Error	Iteration Time	Run Time
PageRank	1e-6	12	4.96
	1e-8	18	7.65
	1e-10	24	9.99
VoteRank	1e-6	12	5.10
	1e-8	18	7.72
	1e-10	24	10.32

Table 5: Time cost of PageRank and VoteRank

5 Conclusions

In this project we propose a novel method, VoteRank, for measuring the influences of users in online community, which is basically adapted from PageRank. We use Python to implement and test both methods on dataset about votes on Wikipedia and show that VoteRank is effective on regularizing the behavior of voting each other with its penalty mechanism. Our method has the same order of magnitude in time complexity and space complexity with PageRank and can be easily and efficiently implemented. Our method also has potential to be applied in other related fields.

The experiments in this course project and this report are finished by myself in Apr 2018.

References

[1] Wikipedia: https://www.wikipedia.org/

[2] S. Brin and L. Page, “Anatomy of a large-scale hypertextual web search engine,” *Proc. 7th Intl. World-Wide-Web Conference*, pp. 107–117, 1998.

[3] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Weiner, “Graph structure in the web,” *Computer Networks* 33:1–6, pp. 309–320, 2000.

[4] J. Leskovec, D. Huttenlocher, J. Kleinberg, “Signed Networks in Social Media,” *ACM Conference on Human Factors in Computing Systems*, 2010.

[5] J. Leskovec, D. Huttenlocher, J. Kleinberg. Predicting Positive and Negative Links in Online Social Networks. *19th Intl. World-Wide-Web Conference*, 2010.

[6] Stanford Network Analysis Project (SNAP): http://snap.stanford.edu/