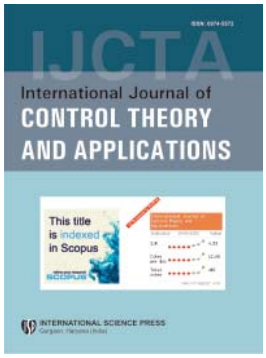


International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 40 • 2016

Enhanced user-Driven Reputation System using Splay Tree

R. Jayashree[a] A. Christy[a] S. Venkatesh[a] and K.Kanmani[a]

E-Mail: jayashreeram77@gmail.com

Abstract: An efficient and effective way to construct trust relationship among peer users in e-learning environment is ranking. User-driven reputation systems are based only on the feedback or ratings provided by the users. Users with higher points obtain high rating compared to less scored users. Thus, by Zipf's law, alleged low users are pushed to the bottom of the ranking list. This condition is avoided by encouraging less reputed users and preventing them from moving further down in ranking level. Thus, low ranked users are provided with few more chances to participate actively in the e-learning environment. A splay tree is a Binary Search Tree with self-balancing skill. The splay tree brings the recently accessed item to the top of the tree. A splay tree is used to represent user's ranks, and to semi-splay low ranked users again in the tree thus preventing them from further drowning in the ranking list for the 'n' number of times. In this paper, we presented re-ranking algorithm to enhance average scored users, and our algorithm is compared with a ranking algorithm in the existing Question-Answering websites. Bayesian approximation is to predict the ranking vector of a user before re-ranking.

Keywords: *User-driven reputation system, Splay tree in ranking, Re-Ranking algorithm, Zipf's law, Collaborative learning, Bayesian approximation method.*

1. INTRODUCTION

Privacy and protection rights are the key challenges that are needed to tackle when capturing and using contextual data. Content driven reputation systems are based on the feedback provided by analysis of all interactions whereas user-driven reputation systems are based only on the feedback or ratings provided by the users. In e-learning environments, co-learners trust relationships play a vital role to establish collaborative activities. For collaboration activities in e-learning environments, the trust relationships among co-learners are imperative. An individual's privacy diminishes by expectations of trust [27]. Reputation is an effectual source for measuring trust, and it is obtained through rating or ranking. Reputation is a contextual evaluation of a person's actions [27]. The splay tree brings the newly accessed items closer to the top of the tree, thus the recently searched items to be accessible in O(1) time if accessed again. The locality of reference states that 80% of the accesses are to 20% of the items. A splay tree search operation does the same standard Binary Search Tree (BST) searching, additionally, it also moves searched user to the top of the tree. If the search is successful, then that user is splayed and becomes the new highest rated user. Else the last user accessed before reaching the NULL is splayed and becomes the new highest rated user. The splay tree allows searching and insertion operations to

balance the tree so that future operations may run faster. Based on the heuristic, if user X is accessed once, then same user X is likely to be accessed again. After locating user X, perform "splaying" operations to bring up X to the top of the tree. Do this in a way that leaves the tree more or less balanced as a whole. The active (recently accessed) user will move towards the root, and inactive users will slowly move far-off from the root. Let user X is not a maximum_Ranked _User that is, X has, at least, one ancestor, and then "Zig" and "Zag" are just a single rotation, as in an AVL tree right and left rotation respectively [24]. "Zig-Zig"/ "Zag-Zag" consists of two single rotations of the same type, and "Zig-Zag"/ "Zag-Zig" consists of two rotations of the opposite type, similar to an LR imbalance correction [24]. This guarantees that even if the depths of some nodes get huge, a long sequence of O(N) searches does not occur because each search operation causes a rebalance. Stackoverflow is a well known Question-Answering website that allows registered users to post their questions and to post answers to others' questions [23]. In general, users with good answers are ranked high. In all Question-Answering websites users with the highest reputation scores are marked as highly reputed users. Overall rating depends on the ratings of users with low reputation. The higher the vote weight of a user with great reputation compared to a vote of a low reputation user, the less the overall reputation will change due to the low reputation votes. In this paper, users with the fewer score are shuffled for a limited number of times so that they do not get ignored in the top reputation list. In other words, moderately reputed or less active users are given few more chances to participate actively and thus, postponing them from getting eliminated from top scored list.

In this paper, reputation-based ranking methods using splay trees are discussed as a significant concept. The re-ranking algorithm is developed to prevent weak users from downfall and encourage them by boosting their ranks. The paper is organized as follows: Section 2 describes motivation and background, Section 3 describes searching time of a user and Re-Ranking algorithm. This section also describes how to break Zipf's law by increasing user level in the splay tree using zig or zag rotation. In section 4, we present reputation ranking method using Bayesian approximation, Section 5 describes and discusses experimental results and finally, section 6 concludes and describes future work.

2. MOTIVATION AND BACKGROUND

The quality and quantity of user's contributions compute their reputations. The good quality contribution preserves the introduced changes in subsequent revisions [30, 31, 32]. User status is evaluated to predict the quality of future user contributions [30]. The predictive ability of the content reputation system is used to measure its performance [27]. The design space characteristics influence the structure of a reputation system. D Movshovitz described the different experts versus non-experts activity patterns and highlighted the importance of detecting anomalous users in his work. The potential expert users are identified based on their business in the first few months of activity on the site. An Initial activity of a user when joining the site is indicative of his/her long-term contribution [29]. Enhancing or reducing the influences of the large-degree users could produce accurate reputation ranking lists [28]. SChord [1] is based on splay tree and implements Chord finger table with improved resource locating efficiency. Nodes hierarchy is related to the access frequency. Routing and caching are the two operations in Schord ring[1] where the routing process is to look up the closest preceding node and caching is searching and inserting (key, node) to its splay tree. Each node contains a splay tree. Insert node n with s successors $(n.id + 2i)$, $0 <= i <= s-1$ into splay tree[1]. Stefan et al. [2] explored fully decentralized and self-adjusting network that minimizes the routing cost between arbitrary communication pairs. They proved by the empirical entropies of the sources and destinations that the overall cost is upper bounded. A new content authentication scheme proposed by Liangbin et. al, in which Merkle hash tree (MHT) is constructed based on an OBST [3]. The basic idea in MHT is to produce a short cryptographic description of a large data set. Parent node stores the concatenated children node values. An element is verified using node's siblings in the path from the associated node to the root. An element's authentication cost depends on the computation time which is a linear to node's depth in MHT. In [4], Randomized splay tree version is presented with chain splay technique for compressing data. An adaptive data compression algorithm called as the splay-prefix algorithm on the prefix code, where the code tree is restructured using semi-splaying. In semi-splaying technique leaf corresponding to

the transmitted symbol is splayed so that it moves halfway to the root, thus moving other symbols automatically to the bottom of the tree. Comparing randomized with non-randomized versions based on rotations and time proves that randomized algorithm is much smaller than the deterministic text of the algorithm. Randomized version [4] achieves up to 3% reduction in the rotations and is preferable for the application of relatively small sequences of accesses on a large amount of data. The splay tree is very suitable for caching the recently accessed content to provide quick access again. The splay tree has good performance [14] since it is self-optimizing. For quick access, move frequently accessed nodes closer to the root. Packets sorted as binary search tree and then balanced tree [12] and self-adjusting tree [13, 21] ideas are implemented to design a cache management for Content-Centric Networking (CCN) [5]. The download time is related to the class popularity that is, the download time is very short for the content with high probability to access. The frequency of visits and the recent visit are considered to evaluate the content popularity [5]. Overall packets matching time reduces with Splay tree by rejecting unwanted traffic in early stages and by accepting repeated packets with fewer memory accesses [6]. Splay tree changes dynamically according to the flow of traffic and is used to the store length of the prefixes. The level of access determines binary search on prefix lengths [15]. Statistical Splay Tree Policy Filters (SSF-BSPL) [6] optimize the early rejection of unwanted flows, and the acceptance of repeated wanted traffic through splaying properties. Filtering processing time for the unwanted packets reduces by arranging policy fields in descending order starting from the area with the highest rejection statistics. In Splay Tree Packet Classification Technique (ST-PC) [7] integer values with their matching rules are stored in splay trees. Whereas in Self-Adjusting Binary Search on Prefix Length (SA-BSPL) [22], the prefix lengths and their corresponding hash tables with matching rules are stored which gives better-amortized analysis. System performance is affected significantly by default-deny rule [7] which increases filtering processing time. Early packet rejection techniques reject the maximum number of packets as soon as possible, thereby filtering processing time is reduced. Key Insertion and Splay Tree encryption (KIST) [8] algorithms use the splay tree for encryption. Key injection algorithm is used to compress the cipher text that moves inner nodes which are higher than specified layer. In cloud environment key insertion and splay tree-based outsourcing key management [8] provides an approach that is highly secure and flexible. SplayNet [9] is a distributed generalization of the splay tree where frequently communicating nodes are moved closer together. Sleator and Trajan [21] proposed splay tree as optimized binary search tree which reduces average access time by moving more popular nodes closer to the root. Harper [17] introduced Minimum Linear Arrangement (MLA) problem [16] to design error-correcting codes with minimum average absolute errors. The domains such as job scheduling [20] and nervous activity in the cortex [19] use MLA concept. Leitao et al. [18] study self-optimizing overlay networks with dynamic topology. Chen Avin et al. [9] designed a double splay algorithm to perform splaying in subtrees, Zouheir Trabelsi and Safaa Zeidan [10] proposed a mechanism based on multilevel filtering modules using the splay tree, to optimize filtering fields order according to traffic statistics. In this scheme, unwanted traffics are rejected in the early stages and thus decrease overall packets matching time. Statistical Splay Tree Policy Filters (SSF-BSPL) [10] system uses a mathematical model to decide statistical policy fields order for the next packet segment. Learning objects in the repository are ranked based on the citation numbers similar to Google page rank [11]. In Slivkins et al.'s [25] work, relevant documents are selected so as to obey the expected relevance rate $\mu(x)$, distributed according to a power-law, for each document x.

3. SEARCHING TIME OF A USER IN REPUTATION BASED TREE AND USER ACTIVITY BASED TREE

Splay trees are the self-adjusting tree with amortized time bounds. In this tree frequently accessed users are moved towards highly reputed users, Max_Reputed_User. In this rotate-to-Max_Reputed_User strategy, the more active users remain close to the Max_Reputed_User and thus can be quickly found. The average cost is O(log n). In this paper, two trees are built. One tree t1 is according to user reputation, where highly reputed users are close to the Max_Reputed_User. Second tree t2 is according to the active user; thus, the lastly accessed user is at the Max_Reputed_User. Searching a user in the splay tree is similar to Binary Search Tree searching.

Let UserN(Y) be the number of users ranked below the user Y then,

$$\text{rank(Y)} = \log(\text{userN(Y)})$$

Let rank`(Y) be the user Y's rank before splaying. The time taken for searching a user is proportional to the depth of the user Y in $t1$ and $t2$ before splaying, that is, the number of links L from Max_Ranked_User to user Y. For "m" number of users, searching operations runs in $\theta(\log m)$ worst-case time.

3.1. Splay tree in user Ranking

Searching time of user x in $t1$ is directly proportional to searching time of user x in $t2$. If the user x is both active and highly reputed, then user x is located approximately at the same level or depth, in both the trees. But in some cases, users are more active at the beginning with the highest reputation and then they may become inactive. In other cases, though users are active they score destitute status. Calculate the reputation of users with their positive or negative votes. As stated in Zipf's distribution, inactive users may be pushed to more inactive/dead state and poorly reputed users to poorest/eliminated state. Avoid this situation by improving the level of users in the splay tree, thus proving chances for weak users to become active and to gain more reputation. Algorithm.1 finds the user x in tree $t2$ and splays the tree using zig and zag rotations.

Algorithm 1. Searching Active User X in A Splay Tree

/* If the Max_Reputed_User is x, then there is no need of splaying. If x exists, where x is not a Max_Reputed_User, then x is splayed to the Max_Reputed_User of the tree. If x does not exist then the last node along the search path for the user x is splayed to the Max_Reputed_User. */

SplayLookup_Active_users(User *x*)

if x is Max_Reputed_User then return /* do nothing since the accessed node is already Max_Reputed_User*/

if x = Max_Reputed_User_Left | Max_Reputed_User_Right then ZIG rotation

$p = x$_ predecessor

if p_left = x & p_predecessor_left = p | p_right = x & p_predecessor_right = p then

Zig-Zig | Zag-Zag rotation

if p_left = x & p_predecessor_right = p | p_right = x & p_predecessor_left = p then

Zig-Zag | Zag-Zig rotation

3.2. Breaking Zip's law

Zipf's law states that very few users are ranked high, and a large number of users listed at the middle level and very huge at the low level. Zipf's distribution shows that very low scored elements are massive in numbers, in other words, small occurrences are extremely common, whereas significant instances are extremely rare. Predict ranking vector of low ranked users using Bayesian approximation before re-ranking that user. In other words, Bayesian approximation method is to find out low ranked users with good ranking vector. Thus, re-ranking these users improve their future ranking. To boost up low ranked users, rotate ranks so that those users' are taken up two steps forward in ranking. Below algorithm improves the weak user position in active_user tree by rotating the tree assuming that user's parent as the root. Let $t1$ and $t2$ be the splayed tree, where keys are ordered according to the user activity value and user reputation value respectively. The grace_upper and grace_lower are the upper and lower levels in the ranking tree with constant values. The grace_depth is the range between grace_upper and grace_lower. The tolerate_factor represent the number of times to do ZIG operation to improve a weak user's rank. The tolerate_factor depends on the rank of the user in $t1$ (pos1) and $t2$ (pos2).

1. If pos1 is in between grace_upper and grace_lower values then calculate 'mean' value of grace_ upper and grace_lower.
2. If pos1 is lesser than mean value and pos2 is lesser than grace_lower then the tolerate_factor is the max_tolerate_factor.
3. If pos1 is lesser than mean value and pos2 is greater than grace_lower then tolerate_factor is mid_ tolerate_factor.
4. Otherwise, tolerate_factor is min_tolerate_factor.

Let us consider max_tolerate_factor = 3, mid_tolerate_factor = 2 and min_tolerate_factor = 1in our Improving Weak User Ranking algorithm. Increase the ranking of a user by doing an additional splay operation, as shown in the Fig.1, considering user x's parent as a maximum ranked user with subtree.

Algorithm 2. Improving Weak User Ranking

Rank_Improvment(user *x*)

//tolerate factor is the number of times to rotate the *x*. To calculate the tolerate factor for the *x* using grace_upper and grace_lower values.

pos1 = Searching In Splay(*t*1, *x*);

pos2 = Searching In BST(*t*2, *x*);

int tolarate_factor = 0, maxtf = 3,

midtf = 2,

mintf = 1;

//grace_upper is the upper limit and grace_lower is the lower limit of the ranks

int grace_upper, grace_lower;

if(pos1 > = grace_lower && pos1 <= grace_upper)

{ int mean = (grace_upper + grace_lower)/2;

if(pos1<=mean && p2 <= grace_lower)

tolarate_factor=maxtf;

if(pos1<=mean && p2 > grace_lower)

tolarate_factor=midtf;

if(pos1>mean)

tolarate_factor=mintf;

}

a[*i*] = *x*;// store *x* in the weak user list

i ++;

for(int *j* = 0; *j* < *i* ; *j* ++)

{if(a[*j*] == *x*)

countme++; //countme counts the number of times the element appears in weak user list

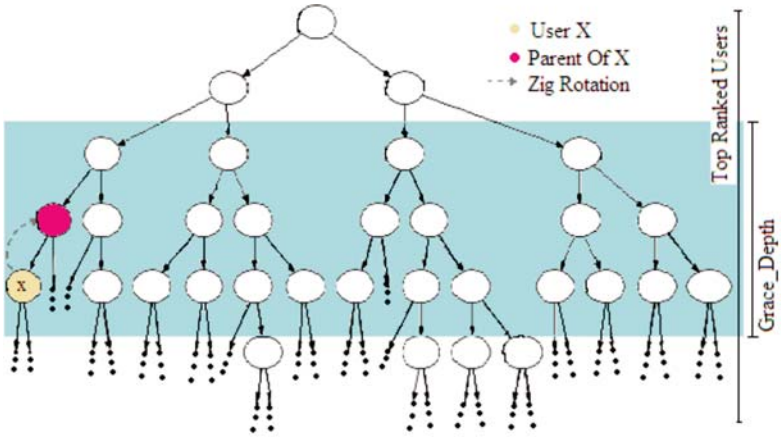
if(countme>tolerate_factor)

println("The element cannot be rotated anymore");

else

{// Move parent below child, and one of child's children below parent.

```
//zig(t1, parent_of_x, x)
if (parent_of_x.left() == x) {
parent_of_x.setLeft(x.right());
x.setRight(p);}
else { parent_of_x.setRight(x.left());
x.setLeft(p);}
}
```



Figute 1: Zig rotation to increase the ranking of user *x*

The splay time at a node '*n*' is proportional to the time to access an item in node '*n*'. Though the size of the tree grows as the number of active users' increases, the depth of the tree is based on the grace_depth value. In other words, only when the searching user's position is within the specified grace_depth, ZIG operation is carried out. Search user '*x*' in *t*1within grace_depth in a top-down approach and in *t*2 through inorder traversal. The grace_upper represents the maximum level in the splay tree, *t*1, to consider for searching '*x*'. The maximum number of nodes in a binary tree of depth grace_upper is 2 grace_upper−1 where grace_upper ≥ 1. The binary search tree with 2 grace_upper−1 nodes takes at least O(log (2grace_upper−1)) comparisons to find a particular node and the total amortized time for a sequence of m operations is O(*m* log (2grace_upper−1)).

4. A BAYESIAN APPROXIMATION METHOD FOR REPUTATION/RANKING

According to Bayesian method the observed data (the reputation) and the model parameters are random quantities. Let 'ObsData' denote the observed data, and 'unknown' the unknown quantities of interest. The prior distribution P(unknown) and the likelihood P(ObsData \ unknown) determine the joint distribution of 'ObsData' and 'unknown'. That is,

$$P(\text{ObsData, unknown}) = P(\text{ObsData} \setminus \text{unknown})P(\text{unknown})$$

Bayes theorem gives the distribution of 'unknown' conditional on 'obsdata' as:

$$P(\text{unknown} \setminus \text{ObsData}) = P(\text{unknown, ObsData})/P(\text{ObsData})$$
$$= P(\text{unknown, ObsData})/\int P(\text{unknown, ObsData})\, n(\text{unknown})$$

This is the posterior distribution of 'unknown', which is useful for estimation. Posterior expectations of some functions, f(unknown), express quantities about the posterior distribution. That is,

$$\text{Post\_exp}[f(\text{unknown}) \setminus \text{ObsData}] = \int f(\text{unknown})P(\text{unknown, ObsData})\, n(\text{unknown}) /$$
$$P(\text{unknown, ObsData})\, n(\text{unknown})$$

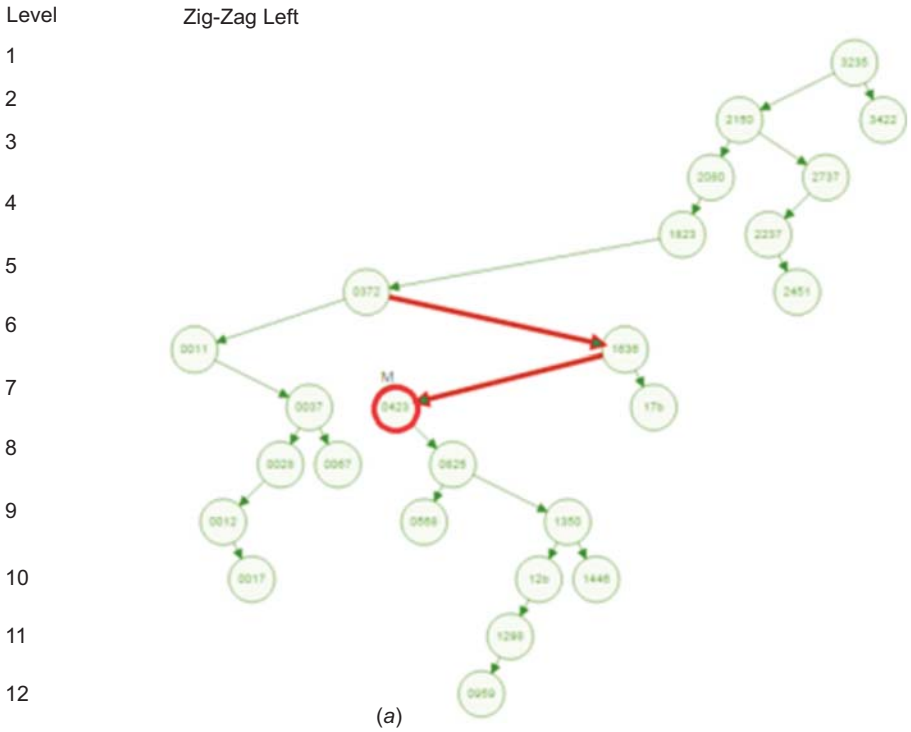
The probability P(ObsData) is useful for model selection, which is called as evidence or marginal likelihood of the data. The major objects of Bayesian inference are both P(unknown \ ObsData) and P(ObsData). In Bayesian re-ranking systems, let the rank of user *x* be represented as R*x*, which is to be estimated. Assuming R*x* has a prior distribution. At the end of the overall re- ranking calculation, update the ranks by either analytical or numerical approximations of the posterior mean and variance of R*x*. Next, reputation value is estimated with the prior information of these revised mean and variance, and the updating procedure is iterative. The posterior mean and variance of $R = [R_1,\ldots, R_n]^T$ characterise the re-ranking calculation. Let the result of a re-ranking calculation and standardized quantity of R*x* are denoted as Re_Rank and $S = [S_1,\ldots,S_m]^T$, where *m* is the number of users. The posterior density of S given the re-ranking outcome Re_Rank is

$$P(s\backslash \text{Re\_Rank}) = \text{(cumulative distribution function of a } m\text{-variance standard normal distribution)}$$

(Probability of reputation-ranking outcome *i.e.*, P(Re_Rank \s)).

5. EXPERIMENTAL RESULTS AND DISCUSSIONS

Reputation is a user's identity that reflects user's familiarity with the website, the amount of users' subject knowledge and the level of respect peers have on the user. Sometimes reputation also determines a user's privileges within the system. Gaining more reputation and trust can make a user access bestow new functionality. As user gain reputation, they gain abilities and responsibilities. The primary factors that determine reputation is users' voting. Posts which are voted up increases users' status, the reverse is true for posts which are voted down. Users' up-votes are more heavily weighted than down-votes. Reputation lost from the reputation cap is not awarded on the following days in the Question-Answering websites such as StackOverflow. Reputation cap is to prevent users from gaining privileges and trust too quickly. StackOverflow badges are similar to ranking, awarded to users for achieving an individual score in a specific tag. Tag score is the combined total number of upvotes and downvotes accumulated on answers under that particular tag. Tumbleweed badge in StackOverflow is to bring attention to a neglected user and thus encourages people to stay on the website. Our Re-Ranking



(*a*)

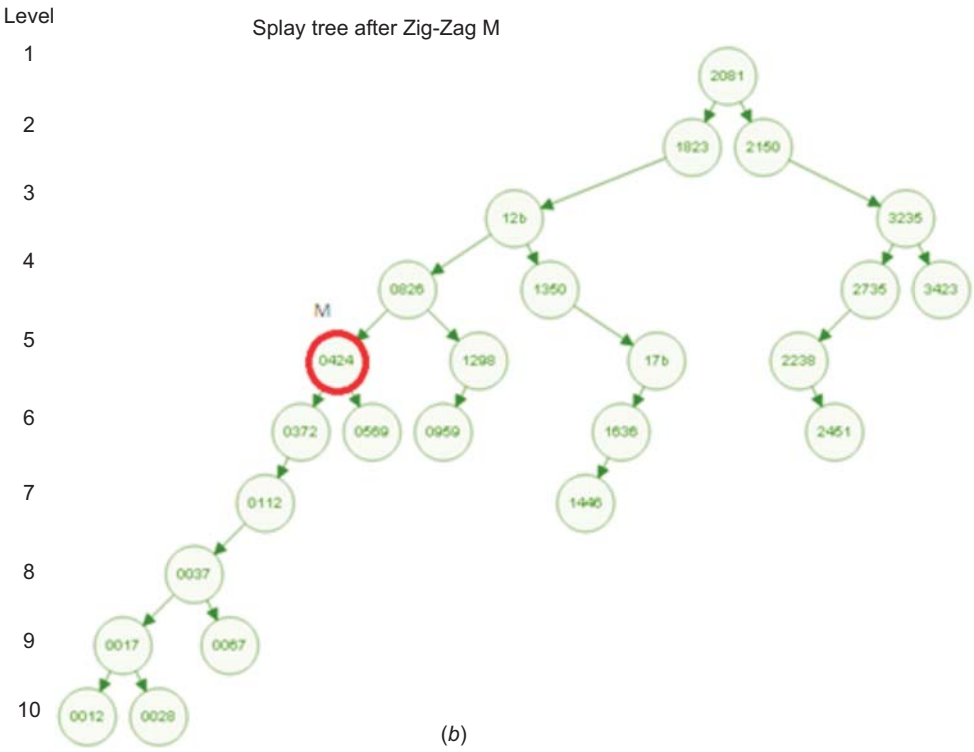


Figure 2: Splay tree of StackOverflow users' reputation values (*a*) Zig rotation of user M (*b*) After Zig rotaion of user M

algorithm is much similar to the tumbleweed badge, but neglected users' ranking is increased unknowingly to other users/voters. Data are collected from the StackOverflow website as shown in the below table with User Id, Reputations, Latest answered/questioned time and level. Bayesian Approximation method predicts user M's ranking vector is in upward. Thus, User M is rotated in Figure.2 (*a*) and pushed forward to level 7 from 9. In Figure.2 (*b*), after rotation, M is at level 5. This concludes that a participant becomes more active after increasing her ranking status.

Table 1
Data collected from StackOverflow website

User ID	*Reputation 1*	*Level in tree 1*	*Reputation 2*	*Level in tree 2*	*Reputation 3*	*Level in tree 3*	*Reputation 4*	*Level in tree 4*	*Reputation after Zig rotation*	*Level in tree after Zig rotation*
A	111	4	111	1	112	3	11	6	112	7
B	28	5	28	4	28	6	28	8	28	10
C	3232	6	323,2	2	3234	2	3235	1	3235	3
D	223,6	12	2236	9	2237	6	2237	4	2238	5
E	82,5	9	825	8	825	7	825	8	826	4
F	11	2	11	4	12	5	12A	9	12	10
G	12A	5	12A	5	12A	6	12	9	12	3
H	1635	8	1635	7	1636	4	1636	6	1636	6
I	2450	11	2450	7	2451	7	2451	5	2451	6

User ID	*Reputation 1*	*Level in tree 1*	*Reputation 2*	*Level in tree 2*	*Reputation 3*	*Level in tree 3*	*Reputation 4*	*Level in tree 4*	*Reputation after Zig rotation*	*Level in tree after Zig rotation*
J	1822	10	1822	8	1823	1	1823	4	1823	2
K	3421	5	3422	4	3422	3	3422	2	3423	4
L	2735	12	2735	8	2735	3	2737	3	2735	4
M	420	3	421	4	422	4	423	9	424	5
N	17	4	17	3	17	5	17	10	17	9
O	1297	5	1297	6	1298	8	1298	10	1298	5
P	567	8	568	9	568	6	568	10	569	6
Q	17	6	17	6	17	5	17	7	17	5
R	959	7	959	9	959	9	959	11	959	6
S	1348	4	1349	5	1350	3	1350	7	1350	4
T	2079	11	2079	9	2080	4	2080	3	2081	1
U	2149	10	2149	8	2150	5	2150	2	2150	2
V	36	2	37	2	37	4	37	7	37	8
W	65	5	66	3	67	5	67	8	67	9
X	371	4	371	5	372	2	372	5	372	6
Y	1445	7	1445	8	1446	5	1446	8	1446	7

In our work, the Re-Ranking algorithm is implemented in Java using NetBeans and it is compared with the existing Question-Answering ranking algorithm. Find weak users and boost up their levels. The implementation results show that the poor users are saved from further drowning to the lower level of the splay trees and their weakness, such as less active or low score, is identified. Boosting up of participants' degree in the splay tree yields them the chances to get active. Thus, these users may become active again. In the traditional ranking methods, the weak participants are not in the limelight, and results in high ranked members alone to grab voter's attention, and thus, only those groups of participants alone remain in the top level of the ranking lists, that is, in the ranking cap. Re-Ranking algorithm overcomes this and breaks zipf's or power law. The studied statistical data of StackOverflow from the tail of top 50 weekly ranked users' list conclude that nearly 75% of users' weekly ranks drop down to the lower levels. Users ranked 35, and above are in the critical place of dropping steep into the ranking list. Figure.3(*a*) show StackOverflow users' weekly rank report for the months January and February 2016.

The above figure clearly proves Zipf's law, that is, if users obtain low reputation or ranking they fall steep into the ranking list. Consider grace_lower as 35 and grace_upper as 45. The users in grace_depth are zig-zig or zag-zag rotated as they are the left or right child of their parents respectively. So that they can be moved two levels high from their current position, just above their parent node, thus preventing them from dropping steep into the list for a limited number of times as shown in the Figure.3(*b*) If user '*x*' is within the grace_depth then the user is zig/zag rotated. Thus, her parent '*y*' becomes her child in the splay tree. Though the rank changes, the parent's reputation, and trust values remain unchanged, thus, the chances of losing up-votes and voters trust are very less. In other words, the probability of '*y*' to get pushed down in the ranking list is very less. Now, user *y*'s expected reputation value in the next move becomes *x*'s expected reputation value. Trust_value is a constant value which represents voters trust on users based on their status. Rotated weak users' weekly rank is determined by

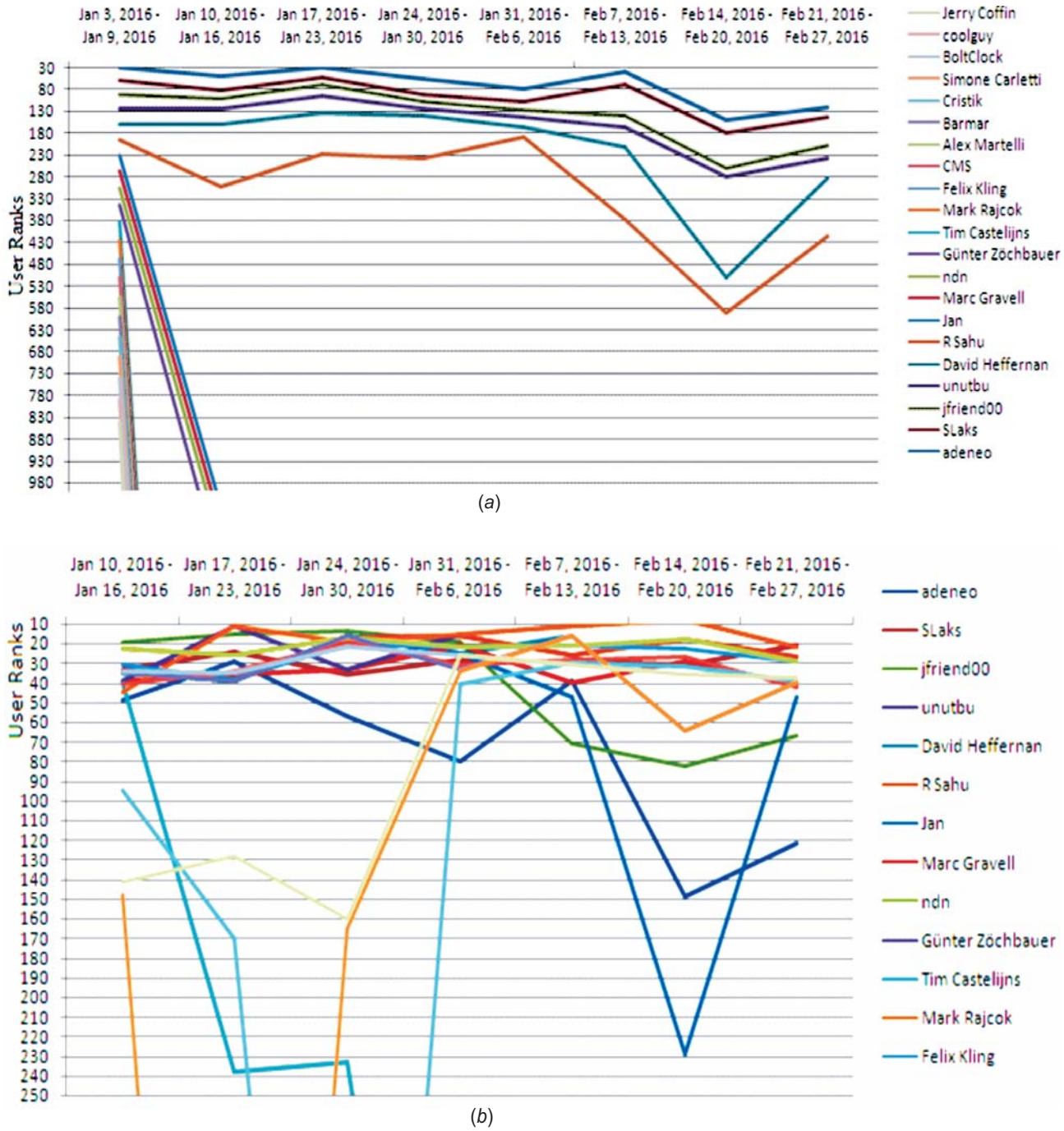


Figure 3: (*a*) StackOverflow users' weekly rank from Jan to Feb '16 before applying Rank_Improvement algorithm (*b*) StackOverflow users' weekly rank from Jan to Feb '16 after applying Rank_Improvement algorithm

*Expected_Reputation_After_Rotation - ((Trust_Value_For_Ranks * Reputation_Change)/Φ)* where Φ is a constant.

The experimental results show that our algorithm has increased weak user performance by increasing their chances to remain in the ranking cap. In our work, 85% of users are made to retain in higher ranks which are almost 60% more than the existing question-answering website.

6. CONCLUSION AND FUTURE WORK

Collaboration activities in e-learning environment require trust relationships among co-learners that obtained through reputation and ranking. The splay tree is a self-adjusting binary search tree where highly active/ranked users are near the root node. In our paper, the splay tree is used to represent user reputation. And users are arranged in the splay tree based on their ranks. Highly rated user occupies the root node. When users are in grace_level, then that indicates they are in the critical level of getting ignored or losing the trust of voters. Identify low ranked users with positive ranking vector using Bayesian approximation method. Thus, to save them from getting very low ranks, zig/zag rotation is done which raised their grade levels. In our work, we examined existing StackOverflow data in our Re-Ranking algorithm. This result shows that 60% of users saved from drowning. In our future activities, users are grouped based on the subject of expertise and are ranked accordingly using multiple splay trees. One user may have more than one subject of interest. Thus, a user can be represented as a node in more than one splay tree. These users connect the splay trees forming a splaynet. Ranking in splaynet is our future goal.

7. ACKNOWLEDGMENT

The authors would like to thank StackOverflow for providing the data used in this paper.

REFERENCES

[1] [1] Wei Zhou, Zilong Tan, Shaowen Yao and Shipu Wang. "A Splay Tree-Based Approach for Efficient Resource Location in P2P Networks. The Scientific World Journal," Article ID 830682. 11 pages. http://dx.doi.org/10.1155/2014/830682. Research Article. 2014

[2] Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler and Zvi Lotker. "SplayNet: Towards Locally Self-Adjusting Networks," IEEE/ACM TRANSACTIONS ON NETWORKING. IEEE. 1063-6692. 2015.

[3] Liangbin Li, Jinlin Wang, Erli Niu and Xue Liu. "Improving Content Authentication Efficiency of Online Multimedia Service. Second International Conference on Networks Security. Wireless Communications and Trusted Computing," 978-0-7695-4011-5/10 IEEE. DOI 10.1109/NSWCTC.2010.171. 2010.

[4] Dimitris Antoniou, Ioannis Kampouris, Evangelos Theodoridis and Athanasios Tsakalidis. "Splaying for Compression: An Experimental Study," Panhellenic Conference on Informatics 978-0-7695-4389-5/11. IEEE.DOI 10.1109/PCI.2011.19. 2011

[5] Haopei Wang, Zhen Chen, Feng Xie and Fuye Han. "A Data Structure for Content Cache Management in Content-Centric Networking," Third International Conference on Networking and Distributed Computing. 2165-5006/12 IEEE. DOI 10.1109/ICNDC.2012.11. 2012.

[6] Zouheir Trabelsi and Safaa Zeidan. "Splay Trees based Early Packet Rejection Mechanism against DoS Traffic Targeting Firewall Default Security rule," Iguacu Falls. Brazil. ISBN: 978-1-4577-1017-9. DOI http://doi.ieeecomputersociety.org/10.1109/WIFS.2011.6123123. pp:1-6. 2011.

[7] Safaa Zeidan and Zouheir Trabelsi. "A Survey on Firewall's Early Packet Rejection Techniques. International Conference on Innovations in Information Technology," 978-1-4577-0314-0/11 IEEE. 2011.

[8] A Mercy Gnana Rani and A Marimuthu. "Key Insertion and Splay Tree Encryption Algorithm for Secure Data Outsourcing in Cloud.," World Congress on Computing and Communication Technologies. 978-1-4799-2877-4/14 IEEE. DOI 10.1109/WCCCT.2014.14. 2014.

[9] Chen Avin, Bernhard Haeupler, Zvi Lotker, Christian Scheideler and Stefan Schmid Ben Gurion. "Locally Self-Adjusting Tree Networks.," IEEE Computer Society. pp: 395-406. DOI 10.1109/IPDPS.2013.40. 2013.

[10] Zouheir Trabelsi and Safaa Zeidan. "Multilevel Early Packet Filtering Technique based on Traffic Statistics and Splay Trees for Firewall Performance Improvement," IEEE ICC - Communication and Information Systems Security Symposium. 978-1-4577-2053-6/12 IEEE. 2012.

[11] Neil Y. Yen, Franz F. Hou, Louis R. Chao and Timothy K. Shih. "Weighting & Ranking the E-Learning Resources," Ninth IEEE International Conference on Advanced Learning Technologies. 978-0-7695-3711-5/09 IEEE. DOI 10.1109/ICALT.2009.36. 2009.

[12] Somaya Arianfar. Pekka Nikander. and Jörg Ott. "On content-centric router design and implications," ACM CoNext Workshop ReARCH'10 Proceedings of the Re-Architecting the Internet Workshop. ACM New York. Article No. 5. pp: 20. 23. 24. ISBN: 978-1-4503-0469-6 DOI 10.1145/1921233.1921240 . 2010.

[13] Adelson-Velskii. G and E. M. Landis. "An algorithm for the organization of information," Proceedings of the USSR Academy of Sciences 146: 263–266. 1962.

[14] Daniel Dominic Sleator and Robert Endre Tarjan. "Self-Adjusting Binary Search Trees. Journal of the Association for Computing Machinery," Vol. 32. No. 3. pp. 652-686. A T&T Bell Laboratories. Murray Hill. NJ 0 1985 ACM 0004-541 1/85/0700-0652. 1985.

[15] M. Waldvogel. G. Varghese. 1. Turner. B. Plattner. "Scalable High Speed IP Routing Lookups," In Proceedings of the ACM SIGCOMM (SIGCOMM '97). pp. 25-36. ISBN: 0-89791-905-X DOI: 10.1145/263105.263136. 1997.

[16] J. Dıaz, J. Petit and M. Serna. "A survey of graph layout problems," Journal ACM Computing Surveys (CSUR) Surveys Homepage archive. Volume 34, Issue 3, pp: 313-356. ACM New York. NY. USA DOI: 10.1145/568522.568523. 2002.

[17] L. H. Harper. Optimal assignment of numbers to vertices. J.SIAM . (12):131–135. 1964.

[18] J. Leitao, J. Marques, J. Pereira and L. Rodrigues. "X-bot: A protocol for resilient optimization of unstructured overlay net-works," IEEE Transactions on Parallel and Distributed Systems. Vol 23. pp: 2175-2188. 2012.

[19] G. Mitchison and R. Durbin. "Optimal numberings of an N X N array," SIAM J. Algebraic Discrete Methods. Volume 7. No.4. pp:571–582. 1986.

[20] R. Ravi, A. Agrawal and P. N. Klein. "Ordering problems approximated: Single-processor scheduling and interval graph completion," In Proc. ICALP. 1991.

[21] T. Srinivasan, M. Nivedita and V. Mahadevan. "Efficient Packet Classification Using Splay Tree Models," IJCSNS International Journal of Computer Science and Network Security. Volume 6. No.5. pp. 28-35. 2006.

[22] Viraj Anchan, Sarang Deshpande, Deep Doshi and Akshat Kedia. "Ranking Algorithm. International Journal of Advanced Research in Computer and Communication Engineering," Vol. 4. Issue 4. Copyright to IJARCCE DOI 10.17148/IJARCCE.2015.4456 248. 2015.

[23] Muralidhar Medidi And Narsingh Deo. "Parallel dictionaries on AVL trees. Parallel Processing Symposium. Proceedings," Eighth International. IEEE ISBN:0-8186-5602-6. pp: 878 – 882. DOI: 10.1109/IPPS.1994.288203. 1994.

[24] Aleksandrs Slivkins, Filip Radlinski and Sreenivas Gollapudi. "Ranked Bandits in Metric Spaces: Learning Diverse Rankings over Large Document Collections," Journal of Machine Learning Research. 14 pp: 399-436. 2013.

[25] Robert Kleinberg, Aleksandrs Slivkins and Eli Upfal. "Multi-armed bandits in metric spaces," In 40th ACM Symp. on Theory of Computing (STOC) . Pages 681–690. 2008.

[26] Mohd Anwar and Jim Greer. "Facilitating Trust in Privacy-Preserving E-Learning Environments," IEEE Transaction on Learning technologies; VOL. 5. NO. 1. 1939-1382/12 IEEE. 2012.

[27] Luca de Alfaro, Thomas Adler, Ashutosh Kulshreshtha and Ian Pye. "Reputation system for open collaboration," Commun ACM Aug; 54(8): pages 81–87. doi: 10.1145/1978542.1978560. 2011.

[28] X L Liu. "Ranking online quality and reputation via the user activity," Research Center of Complex Systems Science. University of Shanghai for Science and Technology. Shanghai 200093. PR China. 2015. http://dx.doi.org/10.1016/j.physa.2015.05.043. 2015.

[29] D. Movshovitz. "Analysis of the reputation system and user contribution on a question answering website: stack overflow," IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. ASONAM'13. Niagara. Ontario. CAN Copyright 2013 ACM 978-1-4503-2240-9 /13/08. 2013.

[30] Adler B, de Alfaro L. "A content-driven reputation system for the Wikipedia," Proc. Of the 16th Intl. World Wide Web Conf. (WWW 2007); ACM Press. 2007.

[31] Adler B, de Alfaro L. Pye I, Raman V. "Measuring author contributions to the Wikipedia," Proc. of WikiSym 08: International Symposium on Wikis; ACM Press. 2008.

[32] Druck G, Miklau G and McCallum A. "Learning to predict the quality of contributions to Wikipedia," Proceedings of AAAI: 23rd Conference on Artificial Intelligence. 2008.