ScoreTree: A Decentralised Framework for Credibility Management of User-Generated Content

Yang Liao, Aaron Harwood, Kotagiri Ramamohanarao

▶ **To cite this version:**

Yang Liao, Aaron Harwood, Kotagiri Ramamohanarao. ScoreTree: A Decentralised Framework for Credibility Management of User-Generated Content. 11th Distributed Applications and Interoperable Systems (DAIS), Jun 2011, Reykjavik, Iceland. pp.249-256, 10.1007/978-3-642-21387-8_20 . hal-01583582

HAL Id: hal-01583582

https://inria.hal.science/hal-01583582

Submitted on 7 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

ScoreTree: A Decentralised Framework for Credibility Management of User-Generated Content

Yang Liao, Aaron Harwood and Kotagiri Ramamohanarao

The Department of Computer Science and Software Engineering
The University of Melbourne, Australia
Email:{liaoy, aaron, rao}@csse.unimelb.edu.au

Abstract. Peer-to-peer applications are used in sharing User-Generated Content (UGC) on the Internet and there is a significant need for UGC to be analysed for credibility/quality. A number of schemes have been proposed for deriving credibility of content items by analysing users' feedback, mostly using centralised computations and/or semi-decentralised approaches. In this paper, we propose our P2P schema, ScoreTree, that decentralises a relatively complex credibility management algorithm by aggregating distributed evaluations and delivering an estimate of credibility for each interested content item. Our experiments show that our schema compares favourably with existing decentralised approaches including a gossip message based implementation of ScoreFinder, and a widely adopted P2P application called Vuze.

1 Introduction

User-Generated Content (UGC) is an increasingly important information source on the Internet. UGC applications process individual data streams from a large number of Internet users and make this information available globally, e.g., Social Networking, Collaborative Content Publishing, File Sharing, Virtual Worlds and other collaborative activities. The value or utility of the information from these applications is dependent to the information credibility – users need to be able to ascertain the credibility/quality of the information in the UGC.

Because it is impossible to manually rank the credibility of large collections of shared content items by any single party, a number of UGC applications allow the users themselves to provide feedback, or to *score* the content items that other users have provided. Recent advances such as in [8] have been made towards more sophisticated methods for aggregating the users' feedback, e.g. by eliminating bias and other anomalous (undesirable) user behavior that can be identified in a set of scores.

Decentralised or *Peer-to-peer* (P2P) approaches have been widely used in recent years for sharing contents contributed and/or generated by users. The long term continuation of P2P content sharing applications raises the need for a decentralised credibility management schema. Addressing this need, we propose our decentralised schema, *ScoreTree*, in this paper. Experimental results show that our method convergence fast, and is more robust against churn and network conditions, in comparison to other current proposals [8, 9, 1].

2 Background and Related Work

2.1 Trust Management and Collaborative Filtering

The quality and authenticity of shared items may be inferred by *Trust and Reputation Management* systems, like P2PRep [2], which is designed for Gnutella-styled unstructured peer-to-peer networks and collect reputation votes by flooding requests, and EigenTrust [6], where a peer has number of trusted peers by manually rating or accumulated interaction experiences, so that the goal of EigenTrust is to infer the global rank of trustworthiness for each peer. The Trust/Reputation Management Model is for managing the trustworthiness of individuals rather than the quality of shared items, hence we need to improve this model, such that items from the same user may be discriminated on their quality.

A very close field to our research is *Collaborative Filtering* [13, 11], for predicting the score that a user may give to a new item by aggregating opinions from other users. This objective is very similar to Web Link Analysis, nonetheless no globally agreed rank for each node is maintained, instead different predictions are given to different users according to their profiles. By combining the methods of Collaborative Filtering and Trust Management, we have proposed our Annotator-Article Model.

2.2 The Annotator-Article Model

The Annotator-Article model is proposed in our earlier work [8] for credibility management applications, where two types of entities are considered: *Articles* that are available for annotation and *Annotators* who annotate the articles, i.e., score them. The evaluation towards an article could be nominal ranks or numeric scores.

We have proposed an iterative algorithm, called *ScoreFinder*, which is applied for offsetting the bias from each annotator and adaptively selecting scores from credible users. The pseudo-code of this algorithm is shown in Algorithm 1. The algorithm iteratively updates the expertise level and the bias level of each user, and then calculates the weighted average of scores to each item using the recent expertise levels and the bias levels.

The input parameters for this algorithm is score matrix $\mathbf{S} = (s_{ij})$, a two-mode proximity matrix that donates the scores that each user i gives to each article j. The scores in $\mathbf{S}$ are linearly normalised into the range between 0 and 1; a higher s_{ij} denotes a better rating. A discriminate function, δ_{ij}, is also defined to determine if a score exists between user i and article j. The value of δ_{ij} equals to 1 if the score exist, and 0 otherwise. The output of this algorithm is vector $\mathbf{r} = (r_j)$, by each r_j, between 0 and 1, denotes the consensus evaluation to the j-th article. A higher r_j indicates a better evaluation of article j.

The expertise levels and the bias-removed scores of users in every iteration are the intermediate results of the algorithm, denoted by vector $\mathbf{e}^\tau$ and matrix $\mathbf{S}^\tau$ respectively, where τ denotes the current iteration number.

The value of γ that is used to control the influence of expertise levels is a trained constant, and the convergence criterion, ϵ, is the difference of the Sum of Squared Errors (*SSE*) between the results in the two iterations. More details about selection of τ are available in [8].

Algorithm 1 The Algorithm of Centralised ScoreFinder schema

$\tau \leftarrow 0$
$\mathbf{r}^{\tau} \leftarrow$ AverageScores($\mathbf{S}$)
repeat
  $\tau \leftarrow \tau + 1$
  $\mathbf{S}^{\tau} \leftarrow$ BiasRemoval($\mathbf{S}, \mathbf{r}^{\tau-1}$)
  $\mathbf{e}^{\tau} \leftarrow$ ExpertnessEstimation($\mathbf{S}^{\tau}, \mathbf{r}^{\tau-1}$)
  $\mathbf{r}^{\tau} \leftarrow$ WeightedAverage($\mathbf{S}^{\tau}, e^{\tau}$)
until SSE($\mathbf{r}^{\tau}, \mathbf{r}^{\tau-1}$) $< \epsilon$
$\mathbf{r} \leftarrow \mathbf{r}^{\tau}$

The formulas corresponding to procedures BiasRemoval, ExpertnessEstimation and WeightedAverage are

$$s_{ij}^{\tau} = s_{ij} - \frac{\sum_j \delta_{ij}\left(s_{ij} - r_j^{\tau-1}\right)}{\sum_j \delta_{ij}},$$

$$e_i^{\tau} = 1 - \left(\frac{\sum_{j=1}^{n} \delta_{ij} |r_j^{\tau-1} - s_{ij}^{\tau}|}{\sum_{j=1}^{n} \delta_{ij}}\right)^{\gamma},$$

$$r_j^{\tau} = \frac{\sum_i \delta_{ij}\, e_i^{\tau}\, s_{ij}^{\tau}}{\sum_i \delta_{ij}\, e_i^{\tau}}$$

3 Decentralised Credibility Management

3.1 Calculating Weighted Average in a Peer-to-Peer Network

A straightforward approach to calculate a weighted average in a P2P network is to nominate a peer, which is in charge of collecting source values from all other peers and propagating the result. The peer is usually nominated by searching the unique identity of the file using the *Distributed Hash Tables* (DHT), as the method that is used in the credibility management component in *Vuze* [1]. In an Internet scale application, a peer that is in charge of managing credibility of very popular items needs to face a very large number of peers sending and receiving the scores, and becomes a bottleneck of the system. A comparison of message numbers between our schema and Vuze's schema is shown later in the experiment section.

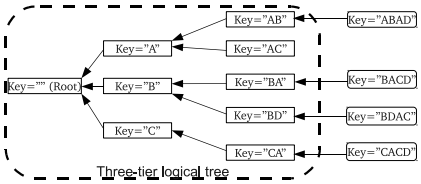
Gossip messages are an efficient way to calculate weighted average without introducing bottlenecks. As discussed in [7, 10], each peer initialises the local, temporary result by its local source value, and continuously exchanges a portion of its local temporary results with each known neighbour. After a sufficient time period, temporary results in all peers will converge to a consistent value, which is the accurate average value of all source values. There are two disadvantages of concern when using gossiping. First, the loss of messages, e.g. due to packet loss, may lead to numerical inaccuracies and requires additional messaging to overcome. Second, the convergence is highly dependent on the network topology; a low connectivity network can take a long time to converge because of the limited propagation speed between partitions of the network.

The *Prefix Hash Tree* (PHT), proposed in [12], is an approach to build a tree structure on the peers, which are organised in a DHT, for hierarchically aggregating and searching data that is distributed over the peers. A peer in structured network usually has a unique identity, like its IP address, to be addressed. Assuming that the hash value is represented by a string $(a_1, a_2, a_3, ..., a_l)$ where l is the length of the string, a sequence can be built by selecting the first k-th characters: $Key_k = (a_1, a_2, ..., a_k)$ for all $0 \leq k \leq l$. Since Key_{k-1} is a function of Key_k, and all peers have $Key_0 =$ "", a tree structure can be built on these keys involving all real peers as leaf nodes, as the example shown in Figure 1(a). We use term *logical node* to call a tree node that corresponds to such a key. Because there is a predictable and unique path between every pair of nodes in a tree, each peer can expect the edge through which data from another given

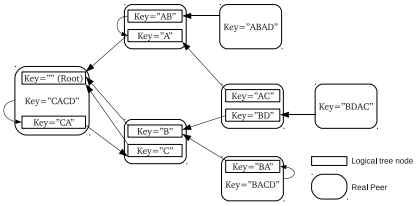
[1] Vuze is a widely used peer-to-peer file sharing application based on an open-source project. The application can be downloaded from `http://www.vuze.org/`. An extension component [1] is provided for managing the quality of shared items by analysing scores given by users.

peer comes, and the maximum length of the paths is strictly limited to $2l$. The two disadvantages of the gossip-message approach can be overcome by exchanging data along these paths, as we demonstrate in the next section. Therefore, we build our ScoreTree schema by mapping logical nodes to real peers, as shown in Figure 1(b). A peer with Key_k sends a message by the DHT to Key_{k-1} for searching its parent peer.

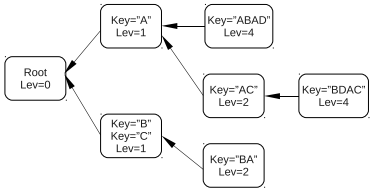
There can be circles in such a tree because a logical node and its ancestors may be mapped to the same peer, as the example shown in Figure 1(b). We use three rules to remove circles in such a tree: (1) we use the hosted PHT node that is on the highest level to represent a peer, (2) if a peer hosts multiple PHT nodes on the same level of the PHT, the PHT node having the minimum key value is selected representative and (3) only the edges between the PHT nodes as the representatives are kept, and other PHT edges are disregarded. Figure 1(c) shows an example of selecting tree edges by applying these three rules.



(a) An example of PHT among logical nodes



(b) Using DHT to map logical nodes into real peers



(c) From a node tree to a peer tree

Fig. 1. Building Prefix-Hash Trees

3.2 Tree-based Average Calculation

The key challenges for implementing ScoreFinder algorithm are to calculate the weighted average of scores and to update the estimates of the bias levels and the expertise levels in a P2P network. We depict the principle of our weighted average algorithm in Figure 2, where each peer in the P2P network is deemed to be a node in a tree. Despite the size of the peer-to-peer network, a tree node can only see a handful of the neighbour nodes – its parent node and its child nodes. Note that a peer who does not contribute s_k and e_k can still join the tree structure by letting $s_k = 0$.

Our approach considers each edge from a peer to another peer as the *delegate* for all nodes that are reached via that edge. Because of the uniqueness of the path between each pair of nodes, there is no overlap between node sets that are delegated by each

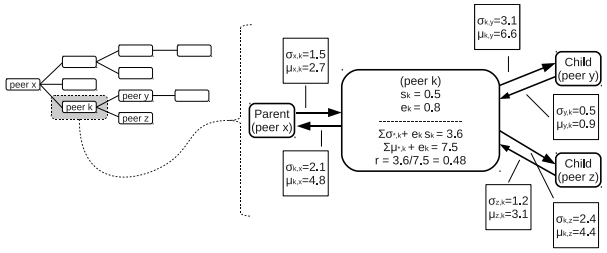


Fig. 2. An example showing the process of calculating the weighted average value on peer k by exchanging data with its parent node and children nodes. By $\sigma_{x,y}$ and $\mu_{x,y}$ denote the weighted sum values and the sum of weights sent from node x to node y. Article number j is omitted from s_{ij} and r_j because only one shared item is considered in this example.

edge of peer k and each node in the tree (except peer k itself) must be delegated by an edge of peer k. Furthermore, if all its edges have been the delegates of the nodes behind them to peer k, these edges are also the delegates of nodes behind them to all k's neighbours. All peers like peer k periodically send to neighbours the sum of σ and μ from all the other edges as well as its local weight and weighted score:

$$\sigma_{k,x} = \sum_{y \neq x} \sigma_{y,k} + e_k s_k, \mu_{k,x} = \sum_{y \neq x} \mu_{y,k} + e_k, \tag{1}$$

where $\sigma_{k,x}$ and $\mu_{k,x}$ denote the sum of weighted scores and the sum of weights that are sent from peer k to peer x. Since peer k receives the weighted sum of all source values and the sum of weights (except e_k and s_k) from its neighbours, it can calculate the above expressions and calculate the weighted average of all source values in the tree:

$$r = \frac{\sigma}{\mu} = \frac{e_k s_k + \sum_{x \neq k} \sigma_{x,k}}{e_k + \sum_{x \neq k} \mu_{x,k}} = \frac{\sum_x e_x s_x}{\sum_x e_x}.$$

Note that σ and μ are equally calculated on all peers, such that all peers reach the same r in $2 \times l$ steps; where l denotes the depth of the tree. For reducing the load on the only root node, a dedicated PHT is built for every article being scored, and a peer accordingly joins a number of trees, each for an article it has scored; article identity is used to differentiate the map between keys and peers in trees for different articles, i.e., $Peer_x = DHT(Article_i + Key_x)$.

Our PHT-guided approach overcomes the two disadvantages of the gossip-message based approach: the tree structure provides convergence in a number of rounds equal to a constant times the tree depth and lost messages can be inferred by each peer explicitly knowing which other peers it expects messages from. For example, if a message carrying $\sigma_{k,x}$ and $\mu_{k,x}$ is lost, node x explicitly knows that data from the subtree that is delegated by node k is unavailable in this round of iteration, so it may use $\sigma_{k,x}$ and $\mu_{k,x}$ received in the last round or ask node k to retransmit the message.

DHT schemes, like Pastry [15] and Bamboo [14], usually provide replication of data items to peers, such that data is not lost from churn. Our ScoreTree schema packages all the scores from a user into a single data item, and uses the DHT to store this data package using the key of its owner; the data package is replicated by the DHT to a number of peers. When the original peer goes off-line, one replica is activated to join the computation on behalf of the peer left until it returns.

4 Experiments

4.1 Baselines, Datasets and The Experimental Environment

We compared out ScoreTree schema with three schemes in experiments:

- the centralised ScoreFinder algorithm that is introduced in [8]
- the schema introduced in [9] that decentralises the ScoreFinder algorithm by exchanging gossip messages between randomly assigned neighbours
- the decentralised schema used in Vuze that store the raw scores at a peer that is selected by DHT

The average scores are the baseline, and all new schemes should have a better accuracy than the average scores. Because our ScoreTree schema, the first schema and the second schema implement the same algorithm, they are expected to converge to the same results.

We use the MovieLens data set [3] to test our schema. The movie data set contains 10 million ratings for 10681 movies from 71567 volunteers; we randomly select a small part of annotators and movies from the data set. In terms of "oracle" scores, we calculate the arithmetic average score from all scores given for each movie as its true score. We use the *Mean Squared Error* (MSE) to evaluate the accuracy of all results from the tested schemes, and show the improvement from the baseline in the charts below.

We built a simulator to emulate a P2P network with at most two thousand peers, and run this simulator on a cluster computer consisting of 20 nodes. This simulator was configured to examine a number of situations, including different packet loss rates and different peer availability schemes. Our schema was implemented in this simulator, as well all other decentralised schemes we tested. A widely adopted DHT schema, Pastry, was used to organise the computation that is rely on DHT.

4.2 Experimental Results

Figure 3(a) and 3(b) show the change of accuracy according to the different scales of the selected data set. ScoreTree achieved similar accuracy to ScoreFinder in all scales of data sets, as well as the gossip-based approach. Because ScoreTree is a decentralised version of ScoreFinder without any change to its hypotheses and operations, ScoreTree is expected to achieve a similar accuracy to ScoreFinder.

A message may be lost in the direct route between any two peers and we simulate random message loss using a constant error probability over all packets sent. Figure 3(c) shows that the accuracy of ScoreTree is better than the gossip-message method when less than 25% messages are lost because no information is lost along with the lost messages; however when the proportion of lost messages exceeded 25%, the accuracy of ScoreTree rapidly decreased because of the broken tree-structure.

Churn is simulated by turning off a random selection of peers, and the results are shown in Figure 3(d). Our replication management significantly improved the robustness of the ScoreTree schema. Without replication management, ScoreTree achieved a better performance than the baseline when there are 80% peers on-line, but approx. 50% of peers can achieve the same accuracy by hosting the replicas of the off-line peers.

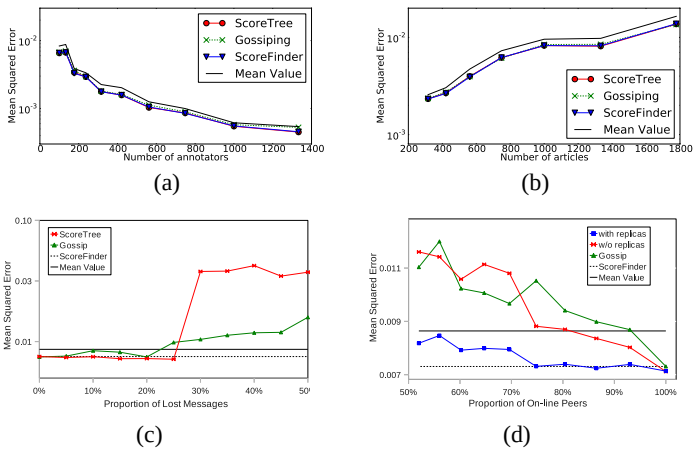


Fig. 3. Comparisons for: (a) number of annotators (number of articles held constant at 400), (b) number of articles (number of annotators held constant at 400), (c) lost messages and (d) proportion of on-line peers. Note that all Y-axes are logarithmically scaled.

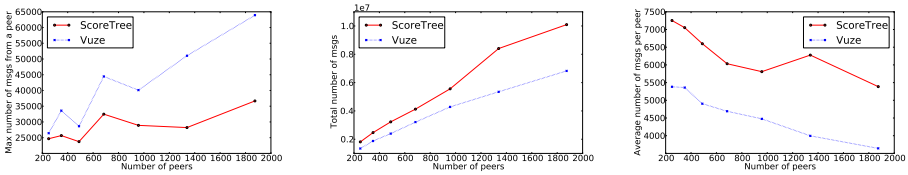


Fig. 4. Comparison of ScoreTree to the rating management module Vuze in terms of message overhead.

We also compare the message overheads between the rating management module of Vuze [1] and our ScoreTree schema in Figure 4, where numbers of all messages for maintaining the DHT/PHT and for computation are recorded. It shows that Vuze always has fewer total messages, but by introducing more peers into the network, the maximum number of messages sent from a Vuze peer increases more rapidly than in our schema. Both Vuze and ScoreTree generate less average messages per peer when the scale of the network increases, giving these two schemes good scalability.

5 Conclusions and Future work

A new decentralised schema, ScoreTree, is introduced in this paper addressing the problem of Credibility Management in p2p networks. The results of the experiments show a better scalability of ScoreTree than the other schemes, as well as a better robustness against network conditions and churn of peers in most situations. Our schema creates overheads for building the DHT between peers and for building the PHT on the DHT, but maximum workload of a peer is better controlled by ScoreTree, and the convergence speed is guaranteed. We note that there are schemes like [5] for building trees between peers without underlying structures, and the depth and width of the tree are controllable. This gives us a chance to reduce the cost for maintaining a tree structure. Integrity is also an issue of concern for our schema when adopted in practical applications. Influence from every node is propagated to all nodes in a tree by our schema; a malicious

peer can arbitrarily change the final result by manipulating σ and μ that it sends to the neighbours. We note that PeerReview [4] may be useful in accounting for the behaviour of every peer and identifying those peers that break the protocol.

References

1. Olivier Chalouhi and Tux Paper. Azureus - rating v1.3.1. `http://azureus.sourceforge.net/plugin_details.php?plugin=azrating`, July 2006.
2. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Managing and sharing servants' reputations in P2P systems. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):840–854, 2003.
3. GroupLens. Movielens data sets. `http://www.grouplens.org/node/73`, October 2006.
4. Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: practical accountability for distributed systems. *SIGOPS Oper. Syst. Rev.*, 41(6):175–188, 2007.
5. H. V. Jagadish, Beng Chin Ooi, and Quang Hieu Vu. Baton: a balanced tree structure for peer-to-peer networks. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 661–672. VLDB Endowment, 2005.
6. Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.
7. D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Annual Symposium on Foundations of Computer Science*, volume 44, pages 482–491. Citeseer, 2003.
8. Yang Liao, Aaron Harwood, and Rao Kotagiri. Scorefinder: A method for collaborative quality inference on user-generated content. In *Accepted by ICDE 2010*, 2010.
9. Yang Liao, Aaron Harwood, and Kotagiri Ramamohanarao. Decentralisation of scorefinder: A framework for credibility management on user-generated contents. In Mohammed Zaki, Jeffrey Yu, B. Ravindran, and Vikram Pudi, editors, *Advances in Knowledge Discovery and Data Mining*, volume 6119 of *Lecture Notes in Computer Science*, pages 272–282. Springer Berlin / Heidelberg, 2010.
10. M. Mehyar, D. Spanos, J. Pongsajapan, S.H. Low, and R.M. Murray. Distributed averaging on asynchronous communication networks. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 7446–7451, Dec. 2005.
11. John O'Donovan and Barry Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA, 2005. ACM.
12. S. Ramabhadran, S. Ratnasamy, J.M. Hellerstein, and S. Shenker. Prefix hash tree: An indexing data structure over distributed hash tables. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing*. Citeseer, 2004.
13. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM.
14. S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference*, pages 127–140, 2004.
15. A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, pages 329–350, 2001.