# Enable Collaborative Learning: An Improved E-Learning Social Network Exploiting Approach

Zhi-Mei Wang
Wenzhou Vocational & Technical College
Department of Computer
Wenzhou,325035, Zhejiang
China

Ling-Ning Li
Shanghai JiaoTong University
Department of Computer Science and Engineering
No.1954, HuaShan Road, 200030, Shanghai
China

*Abstract:* In this paper we propose an improved E-Learning Social Network Exploiting Approach based on Hebbian Learning Law, which can automatically group distributed e-learners with similar interests and make proper recommendations, which can finally enhance the collaborative learning among similar e-learners. Through similarity discovery, trust weights update and potential neighbors adjustment, the algorithm implements an automatic-adapted trust relationship with gradually enhanced satisfactions. It avoids dicult design work required for user preference representation or user similarity calculation. Hence it is suitable for open and distributed e-learning environments. Experimental results have shown that the algorithm has preferable prediction accuracy and user satisfaction. In addition, we achieve an improvement on both satisfaction and scalability.

*Key–Words:* Collaborative learning, Social network, E-learning, Self-organizing Community

## 1 Introduction

E-Learning which breaks the traditional classroom-based learning mode enables distributed e-learners to access various learning resources much more convenient and flexible. However, it also brings disadvantages due to distributed learning environment. Thus, how to provide personalized learning content is of high priority for e-learning applications.

An effective way is to group learners with similar interests into the same community [1]. Through strengthening connections and inspiring communications among the learners, learning of the whole community will get promotion. To achieve a better performance and a higher scalability, the organizational structure of the community would better be both self-organizing and adaptive [2].

Based on the investigation on the behavior of real students, we found out that learners have strengthened trusts if they always share common evaluations or needs of learning re- sources, which is very similar with the Hebbian Learning theory proposed by Hebb D.O. based on his observation on bio-systems [3].

In this paper, we present an improved E-Learner communities self-organizing algorithm relying on the earlier work by F. Yang [4]. The algorithm uses corresponding feedback to adjust relationships between learners, aiming to find similar learners and provide facilities in their collaboration. Experimental results have shown that the algorithm has preferable prediction accuracy and user satisfaction. In addition, we achieve an improvement on both satisfaction and scalability.

## 2 Framework of Collaborative Agent

### 2.1 Agent Feature

Each agent in the community holds a set of resources, which are rated by the user. A bigger rating number means a better evaluation on one resource. Each agent can only be aware of its neighbors, while other agents are out of sight and can not be communicated with directly. No central agent or server exists, so it does not matter any agent breaks down or quits from the community. Hence, it forms a pure P2P environment.

We defined $TNL^i$ to be the trusted neighbor list of user $u_i$, storing the identification information of $u_i$'s trusted neighbors and the corresponding trust weights. Number of an agent's neighbors is limited, since one can not be acquainted with everybody in the real society. We define MON (Maximum Of Neighbors)to signify this limit.

### 2.2 Query Behavior

During the operation of the system, a particular learner could be given lots of candidate resources for

reading. He could then send out a request to his trusted neighbors asking for their advices. Such kind of request will be referred to "query".

If the neighbor nds that the queried resource has not been evaluated, it will then forward this query to its neighboring agents. The depth that a recommendation query will be forwarded is controlled by TTL (time-to-live), to avoid unlimited recurrence. Every time a query has been forwarded, the TTL will be decreased by one. Once the TTL reaches zero or the evaluation on the queried resource has been found, the forward process will cease and the corresponding result will be sent back along the query path.

# 3 Self-Organizing Weight Update Algorithm

## 3.1 Trust Weight Update algorithm

On initialization, each agent will connect to several other agents randomly. These agents will be set as the original trusted neighbors of the agent and assigned a certain trust weight from 0 to 0.1. The topology of the community can be illustrated as Fig. 1.

Figure 1: Structure of Trusted Neighbor List

Once an agent have a query to send out, it will choose the candidates based on M Roulette-Wheel Rule. That is, each neighbor gets a probability to be judged chosen or not. The probability is M multiplied by the ratio of the neighbor's trust weight to the sum of all the weights, and it should not be greater than 100%.

$$ {}^{*}p_j^i = \frac{w_{i,j}}{\sum\limits_j w_{i,j}} \cdot M, \quad p_j^i = \begin{cases} {}^{*}p_j^i & \text{if } {}^{*}p_j^i \le 1 \\ 1 & \text{if } {}^{*}p_j^i > 1 \end{cases} \tag{1} $$

The math. expectation value of the number of the chosen neighbors can be computed as:

$$ E = \sum_j 1 \cdot p_j^i \le \sum_j {}^{*}p_j^i = M \tag{2} $$

Figure 2: Updating of Trust Weights Among Neighbors

When $u_i$ sends a query on $r_k$ to $t$ neighbors,$ne_1$, $ne_2$,$\cdots$,$ne_t$, the query spreads along $t$ pathways and will get $t$ ratings $v_k^1$, $v_k^2$,$\cdots$,$v_k^t$ back before TTL decreases to 0. We compute the recommendation rating on $r_k$ as:

$$ {}^{*}v_k^i = \sum_{j=1}^{t} \frac{w_{i,j} \cdot v_k^j}{\sum_{j=1}^{t} w_{i,j}} = \frac{\sum_{j=1}^{t} w_{i,j} \cdot v_k^j}{\sum_{j=1}^{t} w_{i,j}} \tag{3} $$

${}^{*}v_k^i$ shows the degree of the recommendation from others, by which users can make a determination whether the resource $r_k$ is worth reading or not. While the rating $v_k^j$ denotes an individual view from the neighbor $ne_j$. After reading the resource $r_k$, user $u_i$ makes an rating $v_k^i$ himself. The difference between $v_k^j$ and $v_k^i$ is the key to update trust weights between users.

Based on the Hebbian learning law, we define the trust weight increment in the recommendation network $\Delta w_{i,j}$ by Equation 4

$$ \Delta w_{i,j} = \eta \cdot v_k^j v_k^i \tag{4} $$

where $\eta$ is the learning rate.

We hope a great difference between $v_k^j$ and $v_k^i$ makes $w_{i,j}$ decrease while a little difference makes it increase. So a threshold T1 is set to determine the update direction.

$$ \kappa = \begin{cases} 1 & \text{if } |v_k^j - v_k^i| \le T1 \\ -1 & \text{if } |v_k^j - v_k^i| > T1 \end{cases} \tag{5} $$

If $\kappa > 0$, we say $u_i$ get a positive response. At the same time, greater difference should lead to more decrease, and less difference should lead to more increase. So $h$ is defined to emphasize these differences.

$$ h = 1 + |2|v_k^j - v_k^i| - 1| \tag{6} $$

Besides, if two users have similar ratings in the higher interval or lower interval at [0,1], that means they both specially "like" or "dislike" this resource. It makes sense to strengthen the connection between these two users. So Equation 5 should be modified as:

$$\kappa = \begin{cases} 1 + |v_k^j + v_k^i - 1| & \text{if } |v_k^j - v_k^i| \leq T1 \\ -1 & \text{if } |v_k^j - v_k^i| > T1 \end{cases} \quad (7)$$

The learning rate should include the two factors above, besides, a pre-defined parameter $\beta$ is need to control the overall rate. Hence,

$$\eta = \beta \cdot \kappa \cdot h \quad (8)$$

The updated weights could then be calculated through Equation 9.

$$w_{i,j} = w_{i,j} + \Delta w_{i,j} \quad (9)$$

### 3.2 Potential Neighbor Structure Adaption

In order to speed the community organization process, we add a Potential Neighbor List (PNL) to store the neighbors with similar interests but without direct connections.

If a long query pathway ($length > 1$) returns a similar rating with the current user, we can deduce the end user $u_e$ in the pathway is qualied for a potential neighbor. The strategy of management for the PNL is as follows:

- If $u_e \in PNL^i$, then update its trust weight $w_{i,e}$:

$$w_{i,e} = w_{i,e} + \Delta w_{i,e}, \quad (10)$$

  where $\Delta w_{i,e}$ is calculated by Equation 4.

- If $u_e \notin PNL^i$ and $PNL^i$ is not full, insert $u_e$ into $PNL^i$ and calculate the trust weight as shown in Equation 11.

$$w_{i,e} = \Delta w_{i,e} \quad (11)$$

- If $u_e \notin PNL^i$ and $PNL^i$ is full, calculate $w_{i,e}$ by Equation 11, and check if there exists any potential neighbor $ne_l$ with a lower trust weight than $w_{i,e}$. If there is, insert $u_e$ into $PNL^i$ and delete $ne_l$. Otherwise, just ignore it.

After each update of the trust weight (in TNL as well as in PNL), check whether there exists any user $u_k$ in $PNL^i$ with the highest trust weight $w_k^{max}$ in $PNL^i$ . If there is any such user, insert it into the trusted neighbor set $TNL^i$ if the list is not full. Otherwise, check whether there exists any user with lowest trust weight $w_k^{min}$ in $TNL^i$ and lower than $w_k^{max}$ . If there is any such user, switch it with user $u_k$.

The update strategy is shown in Fig. 2, in contrast with Fig. 1.

## 4 Experiments and Evaluations

We define the term "satisfaction" as a main measurement of performances achieved by agents' behaviors. Each positive recommendation should lead to increasing of the receiver's satisfaction, otherwise, to decreasing.

So the satisfaction is related to the trust weight increment. In order to limit the satisfaction from -1 to 1 and create a nonlinear convex curve due to the principle of diminishing marginal utility, the satisfaction is calculated as:

$$\Delta^* w_{i,j} = \Delta^* w_{i,j} + \eta \cdot {}^* v_k^i v_k^i, \quad (12)$$

$$S = 2arctan(\rho \cdot \Delta^* w_{i,j})/\pi \quad (13)$$

where $\rho$ is set to control the rate of the increment and $\Delta^* w_{i,j}$ is initialized to be 0.

The Hebbian Learning algorithm implemented by Yang [4] was used as a benchmark in contrast with the improved algorithm specified in this paper. The former was referred to "Traditional H. L." while the latter to "Improved H. L.".

As the total behaviors of the learners increase, the average satisfaction of the community climbs as illustrated in Fig.3 with 500 users involved. After 5000 behaviors performed, the performance of the Improved H. L. will exceed the Traditional H. L., which proves its effectiveness in satisfying the recommendation need of learners.

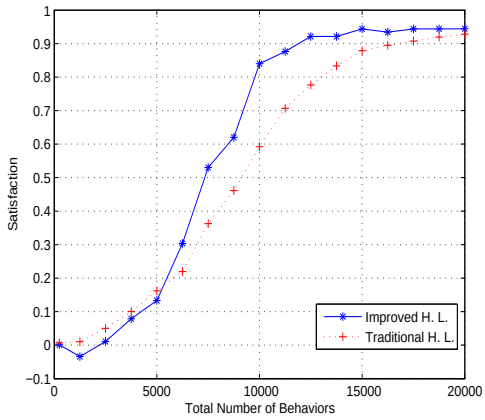


Figure 3: Comparison of Satisfaction between Improved and Traditional Hebbian Learning Laws

Since each user only sends the query to the users selected, Hebbian Learning algorithm has a much lower calculation complexity than the memory based recommendation algorithm such as the Item-based Collaborative Filtering [5],which always has a time

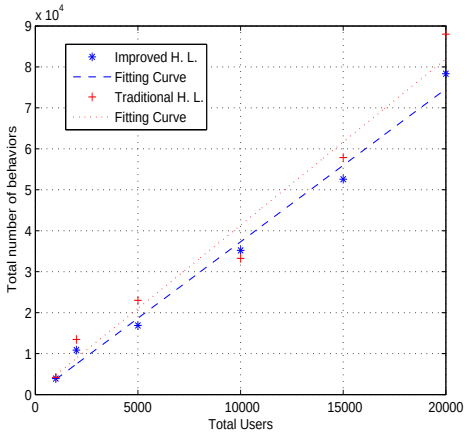


Figure 4: Scalability of the Self-organizing Algorithm

complexity of $\Omega(n^2)$. The Fig.4 shows that the behaviors required by the Hebbian Learning algorithm increased almost linearly in relation to the number of users. In addition, the Improved H. L. achieves a little better performance than the Traditional H. L.

## 5 Conclusion

This paper has introduced a novel method to find and organize similar learners in an E-Learning community. This method is based on the Hebbian Learning law, and takes improvements relying on the earlier work by Dr. Yang [4].

The algorithm presented in this paper avoids dicult design work required for user preference representation or user similarity calculation, while reflect user preferences accurately. In addition, because the community is organized based on P2P communication and local interaction, it is suitable to work in open and distributed environments.

In our future work, we would research on how to enable agents to reason from common sense knowledge-base based on the Ontology theory, which make them more smart to recommend resources to users intelligently and automatically.

*References:*

[1] R. Shen, F. Yang, and P. Han, A dynamic self-organizing e-learner communities with improved multi-agent matchmaking algorithm. *the 16th Australian Joint Conference on Artificial Intelligence* 2003, pp. 590–600.

[2] P. Turner and N. Jennings, Improving the scalability of multi-agent systems. *the First International Workshop on Infrastructure for Scalable Multi-Agent Systems* 2000, pp. 246–262.

[3] D. O. Hebb, *The Organization of Behaviour* 1949.

[4] F. Yang, Analysis, Design and Implementation of Personalized Recommendation Algorithms Supporting Self-organized Communities. *PhD thesis* 2005.

[5] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, Item-based collaborative filtering recommendation algorithms. *the 10th international conference on World Wide Web* 2001, pp. 285–295.