Proposal of recommender system simulator based on small-world model

Ryosuke Saga*, Kouki Okamoto*, Hiroshi Tsuji**, and Kazunori Matsumoto*

**Kanagawa Institute of Technology, 1030 Shimo-ogino, Atsugi, Kanagawa, Japan*
***Osaka Prefecture University, 1-1 Gakuen-cho, Nakaku, Sakai, Osaka, Japan*
(Tel : +81-46-291-3235; Fax :+ 81-46-242-8490)
(saga@ic.kanagawa-it.ac.jp)

***Abstract*:** This paper proposes the development of a software simulator that allows a simulator's users to evaluate algorithms for recommender systems. This simulator consists of agents, items, Recommender, Controller, and Recorder, and it locates the agents and allocates the items based on a small-world network. The agent plays a role in a user in the recommender system and the recommender plays a role in the recommender system. Controller handles the simu lation flow that (1) Recommender recommends items to agents based on the recommendation algorithm, (2) each agent evaluates the items based on agents' rating algorithm using each item's and agent's attribute, and (3)Recorder ob tains the results of the rating and the evaluation measurement for the recommendation pertaining to such informatio n as precision and recall. This paper discusses the background of proposal and the architecture of this simulator.

Keywords: Recommender System, Multi-agent simulation, evaluation, small-world model**.**

I. INTRODUCTION

Recommender systems have been used for several applications and systems such as news sites, information sharing system, e-commerce and so on[1][2][3]. The systems offer benefits to consumer and item providers. These recommender systems help consumers in particular acquire new as well as preferable items and users can expect effective acquisition of the information. Therefore, an appropriate algorithm is needed for the recommender system.

Several researchers have proposed and developed many algorithms since collaborative filtering, one of the most successful technologies for recommender systems, was introduced and attracted many attentions [4][5][6][7]. However, developing and applying the algorithm of collaborative filtering are difficult. One reason is based on the difficulty of evaluating these algorithms. For example, the algorithms were developed for any purposes and have validated specified and limited datasets in many cases. Therefore, the generality of the algorithms is not clear. Another reason for the difficulty is that the validation of the algorithm needs massive dataset.

Therefore, we developed a multi-agent-like simulator for evaluating the collaborative filtering and it is described in this paper.

II. Motivation

Recommender systems aim to recommend preferable items to users from user profile [1][2][3]. The profile is constructed by analyzing the content, user's voting and rating, and access logs, etc. Two types of filtering algorithms are used for dynamic recommendation: content-based filtering and collaborative filtering [6]. Especially, collaborative filtering is the most successful algorithms, and its profile is based on relationships among users or items [7]. It has an advantage wherein collaborative filtering is applicable for any items because the algorithm does not need to analyze the content itself. Also, the hybrid algorithm combining collaborative filtering and content-based filtering [8] has also been developed.

Generally, the recommender system algorithms work better as the dataset that includes the user's rating and item information becomes more massive. However, the algorithms do not work for a small dataset because the dataset is insufficient for calculating similarity and predicting items (called cold-start problems).

Therefore, developing the algorithms has various problems associated with it. The first problem is a limited dataset. To evaluate the algorithms, we need to use various environments by collecting various datasets. However, collecting the various datasets is difficult because we generally do not make use of recommender systems and do not have the data source. Therefore, many researchers have utilized limited dataset such as MovieLens Dataset and EachMovie Dataset. the evaluation measurement may change according to the goals of the algorithm. The algorithm is developed for specified goals, and in order to evaluate the algorithm, the proper evaluation measurements and data set are needed [3]. Also, each method has strong/weak points,

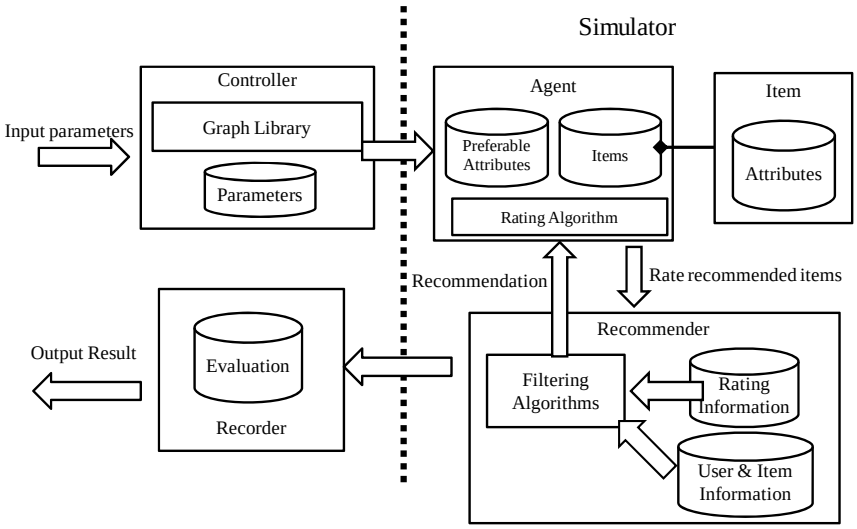

Fig.1. Simulator architecture

and if we apply the algorithms for other goal, we cannot easily judge whether any of the algorithms are suitable. In order to identify the suitable algorithm, to compare the algorithms is useful; however, an experiment with a limited dataset and with different goals is difficult.

Therefore, we build the simulator to enable the comparison of the algorithms. The requirements/goals of simulator are defined as the followings.

1. The simulator can build the evaluation environment for the recommender system.
2. The simulator can compare the filtering algorithms of collaborative filtering and content-based filtering.
3. The simulator can output the results of evaluations to compare the filtering algorithms.

III. OVERVIEW OF RECOMMENDER SYSTEM SIMULATOR

This simulator consists of agents, items, Recommender, Controller, and Recorder as shown in Fig. 1. In this simulator, a simulator user gives the number of agents, items, thresholds, and algorithms as parameters. Agent acts as the user of recommender systems, and the algorithm of collaborative filtering is modeled into Recommender. Recommender has information on agents and items' ratings for each user. Controller receives parameters from the simulator user and handles the simulator such as initialization, progression, and suspension. Controller accepts not only the number of agents and items, but also thresholds for the simulator environment, preference, and agent status.

The simulation steps are as follows:

1. The simulator user inputs parameters.
2. Controller initializes the status of the agents and items and configures the simulator based on the parameters.
3. Recommender calculates the user similarity and recommends items to agents.
4. The agents vote on the rating of recommended items and update the status.
5. Recorder compiles the result of recommendation and evaluates the recommendation using several measurements such as MAE, recall, precision, novelty, diversity, and discovery [9][10].
6. Controller updates agent status and preference.

The simulator regards step 3 to step 6 as one turn, and it goes through the steps and iterates the turns.

IV. ARCHITECHTURE OF SIMULATOR

1. Agents, Items, and Ratings

A. Definition of attributes

The key point of modeling recommender systems is the preferences of agents for items. This paper assumes that the agents and items have many attributes. Here, we assume that the agents have specified preferable genres and domain, and in order to express the assumption, the simulator lets agents have a positive real number for all attributes for their degree of preference. The simulator also defines the attributes whose degree of preference is over the *preference threshold* as preferable attributes. In contrast, an item has the 0/1 flags for each attribute.

B. Rating items

In this simulator, the agents and Recommender evaluate each item. For the agents, the rating r_{ij} of an

item *j* for an agent *i* is generally evaluated by the formulation,

$$r_{ij} = A(U_i, I_j, u_i)$$

Here U_i is the preferable attribute set of agent *i*. Also, I_j is the attribute set of Item *j* whose value is 1. u_i is the information about agent. Also, we call the function an rating algorithm. For an example of rating algorithm, we can configure it as follows

$$r_{ij} = \frac{\sum_u \frac{u}{\max(U_i)}}{|U_i \cap I_j|}$$

where |E| indicates the number of the elements of set *E*, *u* is the elements of attributes of U_i corresponding to the index of $U_i \cap \mathrm{I}$ and max(U_i) indicates the maximum of U_i, which is given by users as one of parameters.

On the other hand, Recommender, which is the implementation of the filtering algorithm, predicts the ratings of the items and recommends high-rated items to the agents. For example, the predicted rating of an item for the agent is generally shown in the following formulation on collaborative filtering:

$$r_{ij} = R(\, sim(i, u) \,, \mathrm{R})$$

,where r_{ij} is the rating of item *j* for user *i*, *sim*(*u*,*v*) indicates the similarity between user/item *u* and user/item *v* among a set of user *U* who evaluate the item *j*, and R shows the rating information. For example, GroupLens[4], which is a representative system using collaborative filtering, has the following formula.

$$r_{ij} = \bar{r}_i + \frac{\sum_{u \in U} (sim(u,i)(r_{uj} - \bar{r}_u))}{\sum_{u \in U} |sim(u,i)|}$$

In this formula, $\bar{r}_i$ is the average rating when a user *i* has already voted.

C. Status of Agents

One of the problems that requires attention is the *tiresome status*. This simulator allows agents to change the status in order to express the degree to which an agent is tiresome. In this simulator, agents have two statuses, *normal* and *tiresome*, and the trigger of changing status occurs in recommendation. The measurement for tiresome is calculated concretely by the evaluation formula, and when the measurement is less than the *tiresome threshold*, the agent changes the status to *tiresome*, otherwise it remains *normal*. The simulator implements the following formula, which is an example of the evaluation formula;

$$t = \sum_{i \in I} \frac{n_i r_{aj}}{rank_i}$$

Here, *I* is the set of recommended items from Recommender, r_{ij} is the rating of item *j* by agent *i* which is used for rating algorithm, $rank_i$ is the rank of the item *i*, and n_i is 0 if *i* has already been recommended; otherwise it is 1.

2. Building Simulator environment

The configuration of the simulator environment is one of the most important steps in the simulation process because an inappropriate environment leads to inappropriate and wasteful result.

The simulator environment is built during initialization and is updated after the recommendation is done. In order to configure the proper simulator environment, we utilize the structural (topological) features of the recommender system.

A. Initialize simulator environment from small world network.

Generally, communities tend to follow complex networks and, according to several references, they find that the networks tend to be small-world networks. Here, a small world is a phenomenon in a real world network, and the model has several features such as stability and compression of network. The structure was first formulated by Watts and Strogatz [10]. The structure appears in several networks, and the trend also appears in recommender system. Therefore, we create the environment based on the small-world network.

If we restrict the community to the recommender system, the network of the recommender system is scale-free network like it is in References [11][12][13]. Therefore, we regard an agent as a node and initialize agents and items according to generating a scare-free network. The algorithm in detail is as follows, given *n* agents and *m* items as simulation parameters from the user,

1. Create the *k*-core clique in order to generate the scale-free network, and allocate the C_0 common items among *k* nodes.
2. Add an agent (a_i (*i*=1, 2, …, *n*)) to the network according to the algorithm of BA(Barabasi-Albert) model which is a scale-free network, and allocate the C_j common items to the clique including a_i. Note that $\Sigma\, C_j = m$.
3. Iterate step 2 by allocating a_n.
4. Allocate the attributes of items to agents who have items in common.
5. Let Recommender calculate the similarity between the agents for recommendation.

B. Update simulator environment

The simulator needs to be update because the agents may have new interests and because Recommender identifies the new relationships between agents because of the rating of recommended items by agents. In this simulator, the agent's preferences are updated by a simple approach. When an agent regards an item as the preferable items, then we can assume naturally that the impression for the item is good. Therefore, when r_{ij}, which is the rating of item *j* by agent *i*, is over the *preferable item threshold*, then agent regards the item as the preferable item and the simulator adds a constant value to the attribute of agents that corresponds to those of the item.

On the other hand, Recommender has trouble indentifying new relationships between agents because the cost of calculation of similarity is normally very high. In order to constrict the complexity of calculating similarity, SketchSort, which is software for all pairs similarity search, is useful [14]. The basic idea of SketchSort is to combine Locality Sensitive Hashing (LSH) and Multiple Sorting Method (MSM). SketchSort takes as an input data points and outputs approximate neighbor pairs within a distance. SketchSort is so quick that the cost of calculating the similarity can be lower. Here, In order to uses the SketchSort, the rating data of agent is recorded by the matrix where the row indicates items and column indicates the agents. However, most agents have not evaluated many items yet so that the simulator cannot build the matrix. Therefore, using default voting [15] and filling the ratings of items which a agent has not evaluated yet, we build the matrix and utilize SketchSort for similarity calculation.

V. CONCLUSION

This paper has proposed a simulator to allow a user to evaluate the algorithms for recommender systems. In order to evaluate the algorithms, the simulator builds an environment of a virtual recommender system based on a complex network model from parameters; Recommender makes recommendations to agents, and the simulator evaluates and outputs the results though Recorder. In future work, we will validate the simulator by gaining a resurgence of the phenomenon of collaborative filtering, and we will test the usability.

REFERENCES

[1] Resnick, P. and Varian, H.R., (1997). Recommender systems. Commun. ACM, 40(3), 56-58.

[2] Adomavicius, G. and Tuzhilin, A., (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

[3] Saga. R., Tsuji, H, and Onoda, J., 2005. Agent System for Notifying Hotel Room Reservation Alternatives, *Proc. of 11th International Conference on Human Computer Interaction (HCII2005)*, Vol. 5, pp. 1-10, July 2005

[4]Resnick, P. et al., (1994). GroupLens: an open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work. Chapel Hill, North Carolina, United States: ACM, pp. 175-186.

[5]Sarwar, B. et al., (2001). Item-based Collaborative Filtering Recommendation Algorithms. Proc. 10th International Conference on the World Wide Web, 285--295.

[6]Pazzani, M. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review. 13(5-6) 393-408.

[7]Linden, G., Smith, B. and York, J., (2003). Amazon.com recommendations: item-to-item collaborative filtering. Internet Computing, IEEE, 7(1), 76-80.

[8] Claypool, M., Gokhale, A., and Miranda, T. (1999). Combining content-based and collaborative filters in an online newspaper. In Proceedings of the SIGIR-99 workshop on recommender systems: algorithms and evaluation.

[9]Herlocker, J.L. et al., (2004). Evaluating collaborative filtering recommender systems. ACM TRANSACTIONS ON INFORMATION SYSTEMS, 22(1), 5--53.

[10]Watts, D.J. and Strogatz, S.H. (1998). Collective dynamics of 'small-world' networks. Nature 393 (6684): 409-410.

[11]Albert, R. and Barabasi, A., (2002). Statistical mechanics of complex networks. Reviews of Modern Physics, 74, 47-97.

[12]Martin-Buldú, J., Cano, P., Koppenberger, M., Almendral, J., Boccaletti, S. (2007). The complex network of musical tastes. New Journal of Physics. 9,

[13] Cano, P., Celma, O., Koppenberger, M., Martin-Buldú, J. (2006). The Topology of music recommendation networks. Chaos An Interdisciplinary Journal of Nonlinear Science. 16,

[14] Tabei, Y., Uno, T., Sugiyama, M., and Tsuda, K.,(2010). Single versus Multiple Sorting in All Pairs Similarity Search, The 2nd Asian Conference on Machine Learning (ACML)

[15] Breese, J. S., Heckerman, D., and Kadie, C., (1998). Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the fourteenth conference on uncertainty in artificial intelligence, Madison, Wisconsin.