A Competitive Information Recommendation System and Its Behavior

Yasuhiko Kitamura[1], Toshiki Sakamoto[1], and Shoji Tatsumi[1]

Osaka City University, Osaka 558-8585, Japan
{kitamura,sakamoto,tatsumi}@kdel.info.eng.osaka-cu.ac.jp
http://www.kdel.info.eng.osaka-cu.ac.jp/

Abstract. Information recommendation systems draw attention of practitioners in B-to-C electronic commerce. In an independent recommendation system such as in www.amazon.com, a user cannot compare the recommended item with ones from other information sources. In a broker-mediated recommendation system such as in www.dealtime.com, the broker takes the initiative of recommendation, and the information provider cannot recommend its item directly to the user.
In this paper, we propose a competitive information recommendation system consisting of multiple animated agents that recommend their items competitively, and discuss the advantages through showing a prototype developed for restaurant recommendation. Each agent recommends restaurants from its own point of view and the user tells good or bad about them. In our competitive information recommendation system, the user can compare items recommended from multiple agents, and the information providers can recommend their items directly to the user through its animated agent. We also show that the competitive nature affects the output depending on the number of participating agents.

1 Introduction

Electronic Commerce (EC) is one of most successful application domains of the Internet. We can access a large number of shopping sites that deal with various goods through a Web browser. A simplest and easiest way of starting a shopping site is just to create a Web page on which items for sales are listed up. When there are too many items to be contained in plain Web pages, we may deploy an information retrieval tool to help customers find their preferred items.

Recently, to make the shopping sites more attractive, new technologies are added to them. Recommender systems initiatively recommend items to customers by using collaborative and/or content-based filtering techniques [2]. They try to reduce customers' burden of searching for their preferred items. Life-like animated agents, such as developed by Extempo (www.extempo.com), are also available to navigate customers to their preferred items in the site. They can build a close and friendly relationship between the shop and the customers.

Most of conventional shopping sites, such as amazon.com, are running in an independent and closed manner. In such a site, customers receive information

for sales items from the site only and any devices for comparing the items with those from the other sites are not facilitated.

To make the comparison easy, a number of comparison shopping sites, such as Bargain Finder, Jango[3], DealTime (www.dealtime.com) and so on, have been developed. However, such a site is run by a third party, which is independent from buyers and sellers, and the design of how to compare and what (attribute) to compare depends on the third party. It is told that many owners of shopping site are not happy with such comparison services because they just raise price competitions ignoring the other additional services offered by the sites.

In this paper, we propose a new multiagent based platform for EC where multiple shopping sites or information recommendation sites are integrated in a flexible and interactive manner. The platform provides a virtual space where multiple animated agents, each of which is delivered from an information site, interact with each other and the user to recommend items in a competitive manner.

In this platform, the customer can compare recommended items with those by other agents and find a preferred one by watching competitive recommendations performed by multiple agents on a browser. Through interactions with the customer, agents can learn his/her preference and use it for further recommendations. From a viewpoint of shopping site, this platform provides a virtually open market place where agents can interact with the customer, and the agents can directly recommend items to the customer without intervention of an information broker such as a comparison shopping site.

In Section 2, we show a prototype of competitive information recommendation system called Recommendation Battlers and how multiple animated agents recommend items in a competitive manner. In Section 3, we discuss the system architecture of Recommendation Battlers and in Section 4, we discuss the rational recommendation method employed in the system. In Section 5, we show a macroscopic behavior of Recommendation Battlers through simulation experiments and show how both utilities of agents and the user change depending on the number of agents and the number of items. Finally, we discuss related works and future work in Sections 6 and 7 respectively and summarize this paper in Section 8.

2 Recommendation Battlers: A Prototype for Competitive Restaurant Recommendation

Recommendation Battlers is a prototype of competitive information recommendation system. As shown in Fig. 1, two restaurant recommendation agents; Peedy and Genie appear on a Web browser. Responding to a request from the user such as "I would like to find a good Indian restaurant near Shinsaibashi station," Peedy starts to gather relevant restaurant pages from Yahoo! Gourmet[1] and extracts their addresses and average cost. It then calculate the distance from the

[1] http://gourmet.yahoo.co.jp/

nearest station by referring to the addresses. Genie also do the same process but it uses another information source called Kansai Walkerplus[2].

After gathering restaurant information from Web sites, Peedy and Genie start to recommend restaurants in a competitive manner. A typical dialogue is given below.

(1)User: "I would like to find a good Indian restaurant near Shinsaibashi station."
(2)Genie: (After gathering information from the Kansai Walkerplus) "Do you like Restaurant A? The average cost is around 2,000 Yen and the distance from Shinsaibashi station is 90m."
(3)User: "Good."
(4)Peedy: (After gathering information from the Yahoo! Gourmet) "Do you like Restaurant B? The average cost is around 2,500 Yen and the distance from Shinsaibashi station is 80m."
(5)User: "No good."
(6)Peedy: "Do you like Restaurant C? The average cost is around 2,200 Yen and the distance from Shinsaibashi station is 80m."
(7)User: "Good."
(8)Genie: "I am sure that you like Restaurant D. The average cost is around 1,500 Yen and the distance from Shinsaibashi station is 70m."
(9)Peedy: "I have no more restaurant to recommend."

The recommendation is performed by two character agents interacting with each other and the user. An agent recommends a restaurant not only by showing the Web page that contains the restaurant information, but also by telling comments about the average cost and the distance from the nearest station. When needed, it asks whether the user likes the restaurant or not by showing a dialogue box. The user responds to the question by telling "Good" or "No good." Referring to the responses, the agents try to learn the user's preference about the cost and the distance and use it for further recommendations. When either of agents has no more restaurant to recommend, the process terminates.

The order of recommendation depends on the agent's strategy. For example, in the above dialogue, Peedy recommends Restaurant B before Restaurant C although Restaurant C apparently looks better than Restaurant B from the viewpoint of the average cost. In this case, Restaurant B is more beneficial for Peedy because we assume the agent receives more fee from the information provider when it succeeds to broker the restaurant. In this sense, agents in Recommendation Battlers are not cooperative but are competitive. In Section 5, we analyze the macroscopic behavior of agents and show how both utilities of agents and the user change depending on the number of agents and the number of items. However, in any way, agents should recommend restaurants in a rational manner such as recommending items in decreasing price order. We discuss more about a rational recommendation method we employ in Section 4.

Recommendation Battlers has the following advantages.

[2] http://www.walkerplus.com/kansai/gourmet/

Fig. 1. A snapshot of Recommendation Battlers. Peedy on the left shows a Web page of an Indian restaurant from YAHOO! Gourmet and says "Do you like Restaurant Gautama? The average cost is around 800 Yen and the distance from Shinsaibashi station is 170m." Genie on the right also shows a Web page of another restaurant called "New Light" from Kansai.Walkerplus.com and the user compare the two restaurants.

– It provides a platform where multiple information sources are integrated in an interactive manner. By preparing another character agent, we can add another information source into this platform.
– It provides a virtual space where multiple animated agents can recommend items in an active and interactive manner. They are just like sales persons who come from different companies. Each sales person talks about his/her goods from his/her own viewpoint. By comparing the talks, we can understand the advantage and disadvantage of goods more than just by hearing from one.
– It provides a space shared among agents. A response from the user to an item recommended by an agent helps other agents learn the user's preference.
– It provides a friendly and easy-to-use interface to the user. Animated agents recommend items to the user in a friendly and active manner, and the user just watches the process of recommendation performed on a browser. Even when requested a response, he/she just does it by telling "Good" or "No good."

3 System Architecture

Recommendation Battlers consists of multiple agents, a recommendation blackboard, and a browser with animated characters, as shown in Fig. 2.

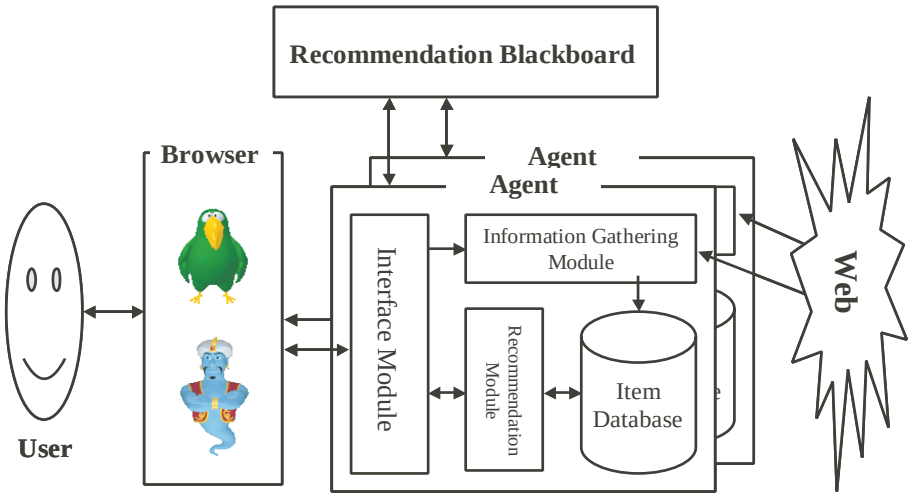

Fig. 2. System Architecture of Recommendation Battlers

3.1 Agent

Agents play the most important role to gather information from WWW information sources and to recommend items through the character interface. An agent consists of an information gathering module, an item database, a recommendation module, and an interface module.

Collecting Information The information gathering module gathers information from a WWW information source by using an information extraction wrapper. The information extraction wrapper consists of MetaCommander[4] and Template Filter.

MetaCommander is a tool to download Web pages by interpreting a given script. It downloads not only simple Web pages designated by URLs but also those that are accessible through the CGI. This module automatically translates a request from the user into a MetaCommander script depending on the target Web site.

For example, let us assume the user submits a request such as "I would like to find a good Indian restaurant near Shinsaibashi station." The module first extracts keywords "Indian" (a recipe name) and "Shinsaibashi" (a place name in Osaka). Currently we just use a simple keyword matching method between keywords in the user's request and those in the tables of recipe and place names, so the system may misinterpret a request such as "I want to find an Italian restaurant in the Indian quarter."

An example of MetaCommander script is shown as follows.

```
getURL( "http://gourmet.yahoo.co.jp/bin/g_searcho",
  "a1" = "15" ,
  "a2" = "270004" ,
  "oc" = "0" ,
  "jl" = "0205" ) {
  file( "yahoo/searchResult.html" ) {
    print
    }
}
```

`getURL` is a MetaCommander command to get a Web page through the CGI. The first parameter is the URL of Web page and the other parameters are optional for the CGI depending on the Web site. In this case, "a1=15" means Osaka, "a2=270004" means Shinsaibashi, and "jl=0205" means Indian food. The information gathering module has a table that translates a recipe or place name into a site-dependent code such as "15" for Osaka, "0205" for Indian food, and so on. The command `file` is to specify the name of file that stores the downloaded Web page.

Template Filter is a tool to extract the designated data by filtering the data from a Web page through a template. Many of Web-based information sites provide information in a fixed format in HTML, so our Template Filter works effectively for such a site.

Gathered items are stored in the Item Database as a set of records in the XML format as below.

```
<?xml version="1.0" ?>
<restaurantData>
  <name>Gautama</name>
  <recipe>Indian</recipe>
  <address>1-18-2 Higashi Shinsaibashi, Chuo-ku, Osaka </address>
  <budget>700--800 Yen (Lunch) <br /> 1100--1500 Yen (Dinner) </budget>
  <url>www.gautama.co.jp</url>
</restaurantData>
```

Currently, we use a very rigid method for wrapping Web information sources. Depending on the target Web source, we have to write a program by using Meta-Commander and Template Filter. We, of course, can reduce the programming task by using these tools much more than by just using a naive programming language such as C or Java.

Recommending Items Each agent controls the Web browser to display a Web page as a recommendation to the user. At the same time, it also controls the character to utter a comment about the recommendation. When it needs to get a feedback on the recommendation from the user, the character asks a question to the user through a dialogue box. By using a character, we can recommend items in an interactive and friendly manner.

In Recommendation Battlers, we use Microsoft Internet Explorer for the browser and Microsoft Agent as the character interface as shown in Fig. 1. The detail of recommendation method is discussed in Section 4.

3.2 Recommendation Blackboard

The recommendation blackboard is used to share information among agents. Recommendation blackboard shares the following information.

Request from the user: It keeps requests for recommendation from the user. For example, when the user submits a request "I would like to find a good Indian restaurant near Shinsaibashi station," a record `<request> <location> Shinsaibashi </location> <recipe> Indian </recipe> </request>` is stored in the blackboard.

Recommended items: It keeps information on items recommended by agents as a set of records consisting of the recommended item, the list of their attributes and values, and the agent that recommended the item.

Feedback from the user: It keeps feedbacks on recommended items from the user as a set of records consisting of the recommended item and the feedback from the user. The feedback is given by "Good" or "No good."

4 Rational Recommendation

In Recommendation Battlers, multiple agents recommend items in a competitive manner, but also they need to do it in an rational order. For example, when the user has accepted a restaurant whose cost is 2,000 Yen and the distance from the nearest station is 150m, he/she is unlikely to accept another candidate whose cost is 3,000 Yen and the distance is 300m. Agents need to recommend items in a way that the user likely to accept, and we propose a rational recommendation method.

The process of information recommendation is as follows.

1. A request from the user is stored on the recommendation blackboard.
2. Each agent accesses the recommendation blackboard to get the request.
3. Each agent gathers information that meets the request from its own Web information source.
4. An agent proposes an item to the user. If there is no agent that has appropriate items to propose, then the flow terminates.
5. The user gives a feedback to the recommended item when needed.
6. Each agent accesses the recommendation blackboard to get the feedback from the user.
7. Go to step 4.

At the step 4 of the above procedure, each agent monitors the other agent's proposals and the user's responses. It then proposes a new item that is more appropriate given the course of dialogue. We here use a rational recommendation method to make agents propose appropriate items.

To simplify the discussion, we make the following assumptions.

- Each of item targeted has only two attributes (cost and distance). We associate item p with cost c and distance d using the tuple $< p, c, d >$
- The lower an attribute value is, the better it is for the user.
- The user can respond to any proposal using either "Good" or "No good."

Our method categorizes items into three groups: R, I, U.

R**:** A group of items that the user is likely to accept.
I**:** A group of items that the user is likely not to accept.
U**:** A group of items for which it is unknown whether they would be accepted or not by the user.

Items placed in R are those whose attribute values are better than those of the items already accepted. On the other hand, items placed in I are those whose

attribute values are worse than those of the items already accepted. Items that do not fall into R or in I are placed into category U. For example, let us assume that only one item $< p_1, 2000, 150 >$ has been accepted. It follows that item $< p_2, 1500, 100 >$ would be placed into R because its attribute values are better than those of p_1. Item $< p_3, 3000, 200 >$ is placed in I because its attribute values are worse than those of p_1. Item $< p_4, 1500, 200 >$ and $< p_5, 2500, 100 >$ are placed in U because it is unknown which attribute is more important to the user at this time.

It is obvious that items in I should not be proposed to the user. On the other hand, items in R and U could be of interest to the user. For an item in U, it is unclear if the user will accept it or not, so the agent needs to ask for confirmation, as in utterances (2), (4), and (6) in Section 2. It is highly likely that the user will accept an item in R, so the agent does not need to ask for confirmation as in utterance (8) in mentioned Section 2.

In the rational recommendation method, the three groups, R, I, and U, must be updated according to the last proposal and the response from the user. We here explain how to update the groups.

Proposal from *R* Let us assume that $< C, 2200, 80 >$ has been accepted and one agent proposes $< D, 1500, 70 >$. As D is in R, the agent does not need to seek the user's confirmation. When the agent proposes D, group boundaries are updated as shown in Fig. 3. Items in region $(x > 1500)$ and $(y > 70)$ become members of I, items in region $(x \leq 1500)$ and $(y \leq 70)$ become members of R,and the remaining items fall into U. This update corresponds to utterance (8) mentioned in Section 2.

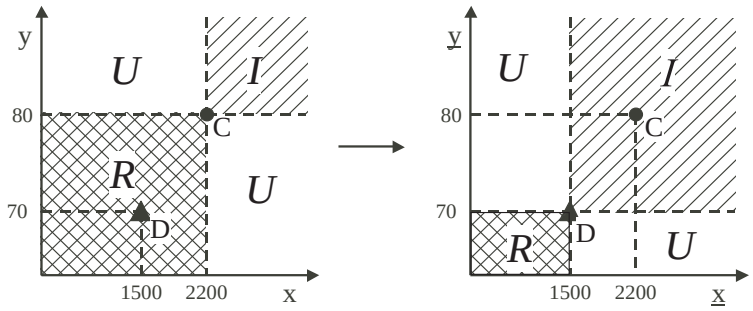

Fig. 3. Update of boundaries when a proposal is made from R.

Proposal from *U* Let us assume that $< A, 2000, 90 >$ has been accepted and the agent proposes $< C, 2200, 80 >$. As C is in U, the agent needs to get confirmation from the user. If the user answers "Good," the boundaries change as shown in Fig. 4. Items in region $(x > 2000)$ and $(y > 90)$ and region $(x > 2200)$

and ($y > 80$) become members of I, and items in region ($x \leq 2000$) and ($y \leq 90$) and in region ($x \leq 2200$) and ($y \leq 80$) become members of R. The remaining items fall in U. This corresponds to utterances (2) and (6) mentioned in Section 2.

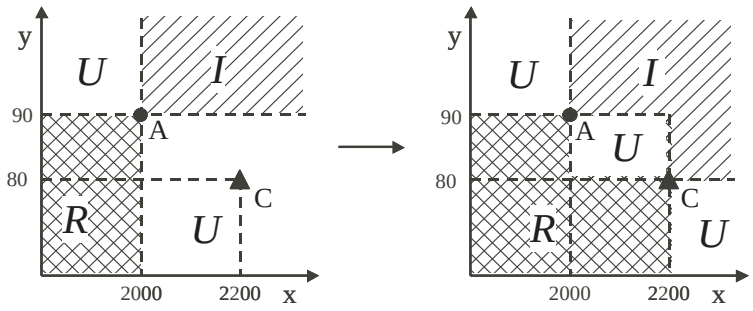

Fig. 4. Update of boundaries when a proposal is made from U and the response is "Good."

In the above situation, if the user answers "No good," boundaries change as shown in Fig. 5. As B is not accepted, boundaries for R do not change. Boundaries for I are updated in the same way as when the user answers "Good." This corresponds to utterance (4) mentioned in Section 2.

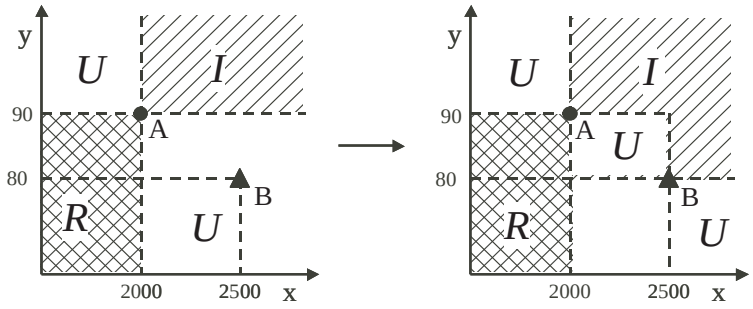

Fig. 5. Update of boundaries when a proposal is made from U and the response is "No good."

With the rational recommendation method, an agent can propose items that are appropriate given the course of dialogue. There may be multiple items in regions R and I, but how to choose one of them depends on the agent's strategy. If the agent is rewarded for securing the user's acceptance, it may well propose items in decreasing order of the reward amount.

Here we discuss cases where agents deal with only two attributes for simplicity of discussion, but we can apply the above method to more than three attributes cases.

5 Macroscopic Behavior of Recommendation Battlers

In this section, we show the competitive nature of Recommendation Battlers affects the output (an item that is finally accepted by the user) of the system from a macroscopic view. It is supposed that the behavior of Recommendation Battlers changes depending on the number of agents. When there is only a single agent, it is clear that it behaves like an independent recommendation system and no competition occurs. As the number of agents increases, the agents become more competitive to recommend items to the user, and that leads to an increase of the user's utility and an decrease of the agents' utility. In this section, we verify the above supposition through simulation experiments.

5.1 Simulation Settings

As settings for simulation, we have the following assumptions.

- Each item C_i has three independent attributes $x_{i,1}, x_{i,2}, x_{i,3}$ and each attribute takes a value from 0 to 1 at random.
- The user's utility, when he/she accepts an item C_i, is calculated from the first and second attribute values as follows.

$$U_u(C_i) = \frac{1}{2}x_{i,1} + \frac{1}{2}x_{i,2}, \tag{1}$$

  where C_i is an item whose attribute values are specified as $x_{i,1}$ and $x_{i,2}$. In a restaurant recommendation system, these attributes correspond to the average cost and the distance from the nearest station.
- The agent's utility, when its recommended item is finally accepted by the user, is identical with the third attribute value of the item.

$$U_a(C_i) = x_{i,3} \tag{2}$$

  In a restaurant recommendation system, this attribute corresponds to the reward from the an information provider when the user finally accepts the recommendation.
- Each agent recommends items following the method mentioned in Section 4. When it has more than two items to recommend, it chooses one with the highest $U_a(\cdot)$.

Our simulation experiment is performed as follows.

1. We prepare items with randomly generated attributed values and equally assign them to each agent.
2. We then run a system to get an item that is finally accepted by the user and calculate the utility values of the user and the wining agent.
3. We perform the above experiments 1000 times and calculate the average of the utilities.

5.2 Results

We show the utilities of the user and the winning agent in Fig. 6 and Fig. 7 respectively when we change the number of agents and the number of items.

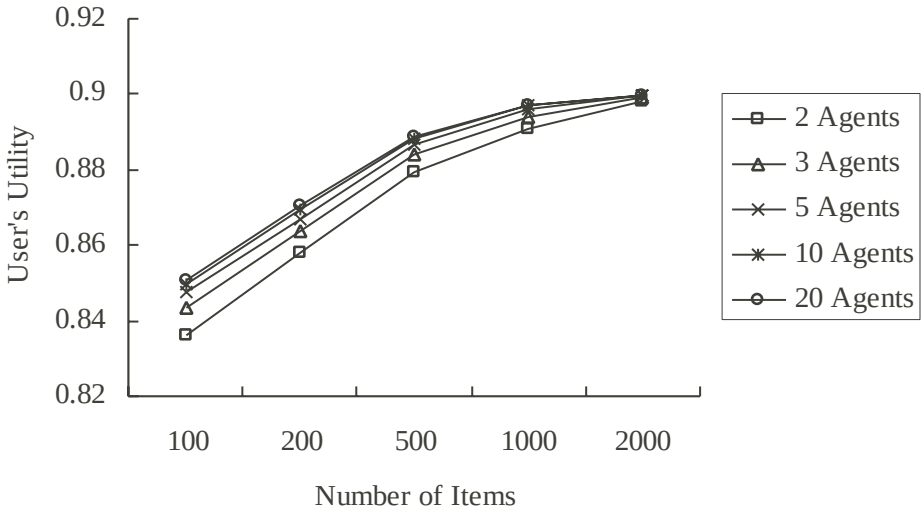

Fig. 6. User's Utility

Fig. 6 and Fig. 7 show that the user's utility increases but that the agents' utility decreases as the number of agents increases. This is because more agents raise more competitions among them and this makes a benefit to the user. The user can get an item that is more preferred. The results also show that both of agent's and the user's utilities increase as the number of items increases. This is because the probability that the finally accepted item is beneficial for both of the user and the agent increases as the number of items increases.

6 Related Work

Information recommendation system provides information to the user from the system side. This is contrasting with conventional information retrieval systems in which the user needs to operate a system by forming and submitting a query. In other words, for a user to obtain information, the user is passive in an information recommendation system, and he/she is active in an information retrieval system. In general, an information recommendation is effective when the user does not have clear requirement or preference nor has a prior knowledge about information sources. Hence, information recommendation systems facilitate a function to estimate the true preference of the user and a function to interact with the

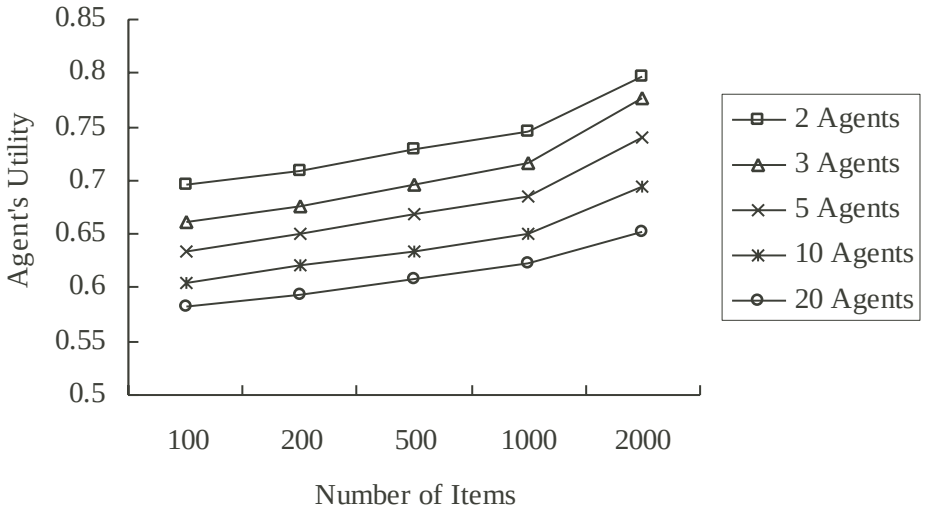

Fig. 7. The Winning Agent's Utility

user. ATA(Automated Travel Assistant)[6] and Expert Clerk[7] are examples of information recommendation systems.

ATA is a system for assisting the user in his/her travel planning. In ATA, CCA(candidate/critique agent) learns the user's model through conversational interactions with the user. CCA represents the user's model as a multi-attribute utility function and learns it by adjusting the function and the weight.

ExpertClerk is a system that navigate a customer on the Internet by using a commodity database and a life-like agent. In ExpertClerk, a life-like agent talks with the customer in a natural language by using two navigation modes; NBA(Navigation by asking) and NBP(Navigation by proposing). NBA produces questions that gain most information on the user's preference by using a hierarchical concept base. NBP displays three contrasting goods to the user to know the user's preference from another viewpoint.

Both of ATA and ExpertClerk are information recommendation systems that try to recommend items to the user through interactions with him/her, but each of them is a single agent system in which the number of employed agent is only one.

Inhabited Market Place[1] has been developed by Andre and her colleagues at DFKI. In this system, multiple animated agents appear on the display and each of them tells its own opinion on an item. For example in a car dealing, an agent tells about the speed of car and another agent tells about the fuel consumption. Hearing and comparing the opinions, the user makes his/her own decision. Agents in this system just tell their opinions following a scenario and do not change them according to the responses from the user. Their work mainly emphasize the advantage of multiple agents as a presentation media.

7 Future Work

Our future works are summarized as follows.

Efficient recommendation. From a viewpoint of the user, he/she wants to obtain preferred items as soon as possible rather avoiding a long transactions with the agents. We need to continue to study how to reduce the number of recommendations when there are a large number of items to recommend.

Shared Ontology. Recommendation Battlers provides a platform where multiple agents compete to recommend items. In the framework, an agent can use responses from the user to items recommended by other agents. To facilitate this mechanism in a general context, agents need to mutually understand the attributes of recommended items. To this end, we need a common ontology that can be shared by agents.

Conversation Capability. In our current prototypes, the user interact with agents by saying only "Good" or "No good," but this is restrictive as a relevance feedback mechanism. For example, a user may respond "No good" because of an experience of bad taste not because of the price or the location of restaurant, but the agent cannot understand the true meaning of "No good." For more complex interactions with the agents, natural language processing or conversation capability is required for agents.

Scalability. In this paper, we just discuss Recommendation Battlers consisting of two or three agents. We need to further study cases where many agents compete to recommend items. At first, it seems difficult that a single user interacts with many agents one time because of at least two reasons. First, the user gets tired of a long transaction of recommendation involving many agents. Second, it is difficult to display many agents on a single browser on the user's PC at once.

A possible solution maybe employ a middle agent between the user and the recommendation agents. At first, a user interacts with a small number of agents and the middle agent records the interactions. When there are more agents to recommend items, they first interact with the middle agent, and the middle agent checks them whether they have at least one item worth to recommend referring to the history of previous recommendations. Only when the middle agent permits, a recommendation agent can directly interact with the user.

8 Conclusion

We proposed a competitive information recommendation system where multiple animated agents recommend items in a competitive and interactive manner. We discussed the advantages of the framework through showing a prototype applied to restaurant recommendation. We also showed how its macroscopic behavior was affected by the number of participating agents.

Including the future work discussed in the previous section, we also continue to work to show the applicability of this framework to real-world domains.

Acknowledgement

This work is partly supported by grants from NEDO (New Energy and Industrial Technology Development Organization), Hitachi Cooperation, and the Ministry of Education, Culture, Sports, Science and Technology.

References

1. Andre, E., Rist, T.: Adding Life-Like Synthetic Characters to the Web. Cooperative Information Agents IV, Lecture Notes in Artificial Intelligence 1860. Springer (2000) 1–13
2. Balabanovic, M., Shoham, Y. Fab: Content-Based, Collaborative Recommendation. Communications of the ACM, 40(3). (1997) 66–72
3. Doorenbos, R.B., Etzioni, O., Weld, D.S. A Scalable Comparison-Shopping Agent for the World-WIde Web. International Conference on Autonomous Agents (1997) 39–48
4. Kitamura, Y., Nozaki, T., and Tatsumi, S. A Script-Based WWW Information Integration Support System and Its Application to Genome Databases, Systems and Computers in Japan, 29(14). (1998) 32–40
5. Kitamura Y., Yamada T., Kokubo T., Mawarimichi Y., Yamamoto T. and Ishida T. Interactive Integration of Information Agents on the Web. Cooperative Information Agents V, Lecture Notes in Artificial Intelligence 2182. (2001) 1–13
6. Linden, G. N. L., Hanks, S. Interactive Assessment of User Preference Models: The Automated Travel Assistant, Proceedings of the Sixth International Conference on User Modeling. (1997) 67–78
7. Shimazu, H. ExpertClerk: Navigating Shoppers' Buying Process with the Combination of Asking and Proposing. Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (2001) 1443–1448.