A Multiple-Bidding Support Framework for Bidding and Browsing Information

Hiromitsu Hattori, Ryota Yamada, Tadachika Ozono and Toramatsu Shintani
Department of Intelligence and Computer Science
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555 Japan.
{hatto,ryota,ozono,tora}@ics.nitech.ac.jp

ABSTRACT

Because of the increasing sophistication of Internet auctions, a user can participate in many different auctions held around the world, each of which offers a wide variety of items. In this paper, we present multiple-bidding support framework based on multiagent system, which can support bidding and browsing information. To extract information on items from real-life Internet auction sites, we implement a Web wrapper component which can be connected with a template generator. In this paper, we present a process of generating a template.

Keywords

Multiagent System, Bidding Support Agents, and Browsing Support

1 Introduction

Increasingly, there has been a growing interest in Internet auctions as a means of electronic commerce. Because of the increasing sophistication of Internet auctions, a user can participate in many different auctions held around the world, each of which offers a wide variety of items. However, there are two difficulties with bidding at multiple auctions. One is determining bidding strategies for winning the items. It is difficult for common users to determine a bidding price depending on the progress of multiple auctions. The other is collecting information on the many items. To compare the price of items on different auction sites, users must access each site, search for desired items, and get their prices. To overcome these difficulties, we have constructed a multiple bidding support framework that consists of *MultiBidder* and *MultiHammer*. *MultiBidder* is an extension of *BiddingBot* [2] that enables bids to be made on any combinations of items. In this paper, we focus on the *MultiHammer*, a system which can support extracting information on items from real-life Internet auction sites.

In *MultiHammer*, multiple agents can extract information on items simultaneously and present integrated information. Users can access *MultiHammer* using an appropriate interface for their browsing environment, and then they can browse many information on items efficiently. Here, because of the difference in the description format of HTML document, each agent behaves as a Web wrapper [4]. To extract information, agents need data on the document structures. To determine the meanings of extracted information, agents need data on the semantics of the information. We use these data as a template for extracting information. However, it is difficult and time-consuming for users to develop this template because they must analyze the description formats of the HTML documents in detail. To make it easier for users, we have developed a template generator (the basic idea is proposed in [3]). Using our template generator, users can semi-automatically generate a template.

2 Multiple-Bidding Support Framework

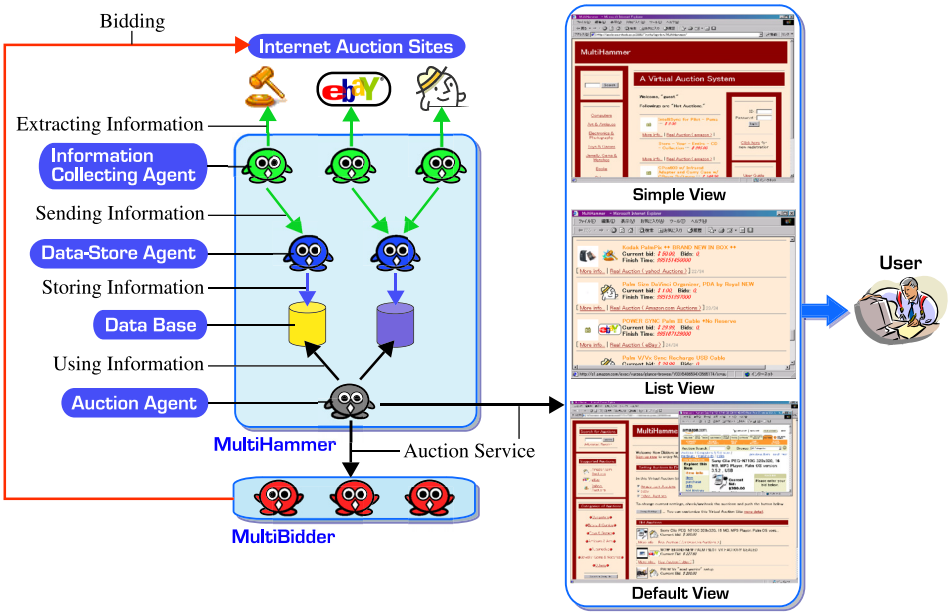

Figure 1: Multiple Bidding Support Framework

Figure 1 shows our multiple bidding support framework, which consists of *MultiBidder* and *MultiHammer*. In this figure, we emphasize an architecture of *MultiHammer*. This framework has been developed using Java language and *MiLog* (A Logic-based Intelligent Agent Framework) [1] which enables us to develop an intelligent web-agent easily.

MultiHammer consists of three types of agents, i.e., auction agents, data storing agents, and information collecting agents. The information collecting agents extract information from various real-life Internet auction sites. Because these agents are implemented as mobile agents, each can work at a different host for efficient information extraction. The data storing agent stores the collected information in a database using a SQL. The auction agent provides auction services using the stored information. Users can access the auction agent via web browser. The auction agent provides three different types of interfaces, i.e., "Default View", "List View", and "Simple View." Using the default view, users can browse many information efficiently, and there are enough links to other information sources (e.g., the web page representing detailed information on item). The list view presents information on items as a list form. Users can browse information roughly using this interface. Since there is no large data, the simple view is suitable for accessing from PDA (e.g., Palm Pilot). Furthermore, users can browse many information smoothly because the information collecting agents extract information in parallel with the browsing information.

3 The Process of Information Extraction

Figure 2 shows an outline of the process for a template generation. The process consists of following four steps.

(i) Extract two HTML documents on two different items from the same auction site and convert them into tree structures. To generate a template, at least two HTML documents described based on the same description format must be input. To convert each document into a tree structure, each HTML tag and character string is regarded as a node. In the following steps, we represent the two input documents as X and Y. The tree structures of X and Y are represented as P and Q, respectively.

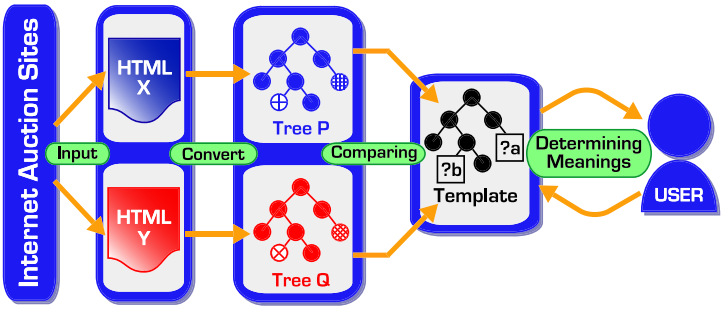

Figure 2: Process of the Template Generation

(ii) Compare P and Q, and obtain common description pattern R. In P and Q, we trace and compare the nodes from the root-node to each leaf node according to a depth-first search algorithm. Concretely, we compare p_i and q_i, which are elements of P and Q, to obtain r_i, an element of R. If $p_i = q_i$, then $r_i = p_i$. If $p_i \neq q_i$, then r_i = \$DIFF, which means that nodes p_i and q_i are characteristic information.

(iii) Extract information by matching common description pattern R to tree structure P. If r_i = \$DIFF (in Figure 2, we represent such nodes as "?a" and "?b"), we extract p_i and add it to list S_R. S_R is the list of extracted information, so it represents the characteristic information of document X.

(iv) Determine the meaning of the extracted information. In this step, we determine the meaning of each extracted information $s_i \in S_R$. If s_i is required information, we determine meaning l_i by referring to P, and add l_i to list L_R. On the other hand, if s_i is not required, l_i is "USELESS" and is added to L_R. As a result, L_R is the list of the meanings of S_R. For example, when the extracted information list S_R = ['PalmPilot','\$42.00','2days,16hours'] is extracted, we would determine the meanings L_R = ['Item Name','Current Price','Time Left'].

Although we must determine in step (iv) the meaning of the information, the template can be generated without analyzing the description format in detail.

4 Conclusion

In this paper, we present multiple-bidding support framework based on multiagent system, which can support bidding and browsing information. To extract information, we implement a template generator for a Web wrapper in our framework. We present the process of the template generation.

References

[1] Fukuta, N., Ito, T., and Shintani, T., : MiLog:A Mobile Agent Framework for Implementing Intelligent Information Agents with Logic Programming, In *Proceedings of the First Pacific Rim International Workshop on Intelligent Information Agents (PRIIA2000)*, pp.113–123, 2000.

[2] Ito, T., Fukuta, N., Shintani, T. and Sycara, K., BiddingBot: A Multiagent Support System for Cooperative Bidding in Multiple Auctions, In *Proceedings of the 4th International Conference on Multiagent Systems (ICMAS-00)*, pp.399–400, 2000.

[3] Yamada, R., Hattori, H., Ozono, T. and Shintani, T., "MultiHammer: A Virtual Auction System based on Information Agents," In *Proceedings of the Pacific Asian Conference on Intelligent Systems (PAIS-2001)*, pp.73–77, 2001.

[4] Wells, D., "Wrappers.," `http://www.objs.com/survey/wrap.htm`, 1996.