PCFinder: An Intelligent Product Recommendation Agent for E-Commerce

Bin Xiao, Esma Aïmeur and José Manuel Fernandez
Computer Science Department, University of Montreal
{xiaobin, aimeur, fernandz@iro.umontreal.ca}

Abstract

There are many e-commerce applications on the web. A common shortcoming is the lack of customer service and marketing analysis tools in most e-commerce web sites. In order to overcome this problem, we have constructed an intelligent agent based on Case-Based Reasoning (CBR) and collaborative filtering, which we have included in our product recommendation system, called PCFinder.

This system was four main characteristics. The first is applying novel methodologies based on CBR to an e-commerce application. We propose a heuristic to represent an Order-Based Similarity Measure, together with the method of weight modification and adaptation. The second is applying CBR and collaborative filtering techniques to make our intelligent agent more efficient and effective. We also apply clustering analysis techniques to assist our intelligent agent for grouping the customers according to their long-term profiles in order to analyse the user profiles (external attributes) and provide some suggestions of the items (internal attributes) of the product. The third is introducing a method for constructing product recommendation systems: from architecture to methodologies and from applied technologies to implementations. The last is providing a graphic-building wizard based on clustering analysis of the past purchasing history to the management staff for analysing the marketing tendencies.

1. Introduction

Electronic commerce is steadily becoming more important in changing the way people exchange products and services. It provides a convenient and easy way for both the consumer to buy things and the merchant to sell things. However, some problems remain unsolved if people wish to take complete advantage of this new paradigm. One of these problems is the lack of customer service featured in e-commerce applications. Currently, product support offered by most organizations to their Internet customers is of comparatively poor quality, if it exists at all. Certainly, most organizations' web sites offer their customers the possibility to query (as opposed to physically examine) the available products using catalogues, textual search engines or database interfaces. These tools, however, require the user to exert some efforts during the interaction phase. Most of the time, these tools are not enough for the consumer (i.e. the Internet user) if the number of products is substantial, if the products are similar to each other or if the consumer does not know the domain very well. Or many customers who could be lost and feel frustrated facing this large number of choices, the only thing they feel easy to do is just get off the site and never come back again [2]!

One of the solutions to this problem is to use *Product Recommendation Systems (PRS)*, which proactively suggest products to the user according to specified user preferences or requirements. PRS contribute to increase customer satisfaction, therefore to enhance brand recognition and improved market performance for the organization. One of the promising technologies for the conception of recommendation systems is *Case-Based Reasoning (CBR)* [1]. The purpose of this sub-domain of artificial intelligence is to conceive knowledge-based systems, which, in solving new problems, reuse and adapt solutions to prior similar problems [9].

In this paper, we first provide a brief overview of agent-based systems in e-commerce. Then we describe the three-tiered architecture of our prototype, PCFinder, whose purpose is to help a computer e-shop propose the most appropriate products to its clients. Based on that, section 4 presents the algorithms and methodologies used in PCFinder. It focuses on dynamic Order-Based Similarity Measure, weight modification, adaptation, profile and collaborative filtering, and clustering analysis. Section 5 describes the implementation of PCFinder. Finally, we discuss pros and cons of this method and propose some future directions.

2. Agent-Based Systems in e-Commerce

Electronic commerce is booming with increasing accessibility to the Internet in virtually every corner of the world. In the new generation of e-commerce, agent-based systems are becoming an attractive paradigm. Agents have demonstrated their tremendous potential in conducting various tasks in e-commerce, such as searching, buying and selling products, etc.

An *agent* can be anything that perceives its environment through sensors and acts upon that environment through effectors [13]. In the case of our PCFinder, the environment includes two parts. One is the database that saves product information, customer profiles and historical purchasing record for searching and recommending. Another one is the computer terminal for interaction with the user. The agent's percepts are the words of an *HTML (Hypertext Markup Language)* document acquired from using software sensors that connect through the Internet/Intranet utilizing *HTTP (Hypertext Transfer Protocol)*. The agent's actions are to determine whether an appropriate product has been found matching its criteria of seeking. It acts on the environment using output methods to update the user on the status of the search or the end result(s), which is hopefully the attainment of the goal.

According to the sources of data on which recommendation is based and the way in which that data is put into use, the majority of product recommendation systems are developed using *content-based* [10], *collaborative-based* [14], *constraint-based* [8] filtering or *knowledge-based* [5] methods as their underlying technologies. Recently, *Case-Based Reasoning* (*CBR*) [3] as a category of knowledge-based methods has become a promising technology in agent-based e-commerce systems [7][9].

Our system differs from other product recommendation systems in that it applies novel CBR methods, such as the Order-Based Similarity Measure, weight modification and adaptation. For example, if the customer is not satisfied with any of the recommended results, the Intelligent Travel Recommender (ITR) [12] suggests a new query by tightening or relaxing some of the query constraints; XMLFinder [9] asks the customer to select the attributes he is most dissatisfied with and then modifies the weight of these attributes. PCFinder employs an alternative method to increase the weight of the most important attribute and also provides a method to automatically adapt the result based on the customer's request. Another feature of PCFinder is a graphic-building wizard based on the clustering analysis of purchase records allowing the management staff to analyze market tendencies.

3. Architecture of PCFinder

Our PCFinder prototype uses a typical three-tiered architecture (see Figure 1). The first tier provides a way to present data to the user. The middle tier is in charge of retrieving the data from data sources and provides a well-defined interface for the first tier to access the data. The third tier contains the data sources. The three-tiered model is more flexible and scalable than the traditional client/server architecture, which is why we use it.

The first tier of our application includes personal computers with browsers. The intelligent agent, PCFinder, runs in the middle tier. It includes four modules, a management module, a search module, an adaptation module and a clustering analysis module. It also includes generative rules. The management module is in charge of interacting with the first tier, managing the others modules and maintaining the database in the third tier. The search module helps the customer to find the most appropriate product from the product base (case base) according to the customer's query and long-term constraints. The adaptation module modifies products to better fulfill the customer's needs when the customer thinks it is necessary to do so. The clustering analysis module analyzes and groups the information of customer profiles database and historical database, and generates rules so that management module can adjust some initial values, such as initial weight of the attribute. The third tier contains the database. It includes a case base that contains product data, a case constraint database that contains adaptation criteria, a customer profile database that contains customer's background and long-term constraints, and a historical database that contains the historical purchasing records of each customer.

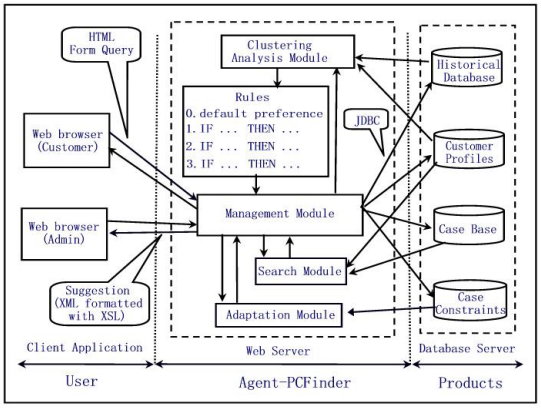

Figure 1. Three-tiered architecture of PCFinder

Table 1. Local similarity measure for the "processor speed" attribute

Serial Number (*i*)	1	2	3	4	5	6	7	8	9
Processor Speed (MHz)	700	750	800	850	900	950	1000	1200	1400
Similarity	0.6	0.8	1.0	0.8	0.6	0.4	0.2	0	0

The first tier sends HTML form queries to the middle tier by encoding commands and arguments in HTTP requests. The middle tier retrieves the data from the case base in the third tier or maintains the data in the third tier by JDBC. Then it responds with XML documents formatted with XSL. Using XML instead of HTML means that we can have dynamic content without sacrificing usability or interoperability.

4. Methodologies and Algorithms

In this section, we introduce methods and algorithms used in PCFinder, including CBR, clustering analysis, and profile and collaborative filtering.

4.1. Dynamic Order-Based Similarity Measure

Local similarity measures largely depend on the application domain, but they all serve the same purpose: to return an estimation between 0 and 1 and to indicate the similarity between a particular attribute of a case and its equivalent in the request [2].

Here, we present a method of Order-Based Similarity Measure. This method is based on *Order-Based Retrieval* [4]. The customer supplies a variety of information (preferred values, values to be avoided, maximum values and minimum values, for example) and we construct an ordering relation from this information. Then we can use this ordering to calculate the local similarity.

The advantage of Order-Based Similarity Measure is that it is easy to maintain the similarity attribute-value pair table. We can get the similarity attribute-value dynamically rather than from pre-initialization. For example, when a new attribute of some case is added in the case base, we do not need to update the similarity attribute-value pair table saved in XML documents or other databases.

Consider a set $V = \{a_1, a_2, \cdots, a_n\}$ of values for a specific attribute of some case. Assume that $a_{i-1} < a_i$ for each i, $1 < i \le n$, where n is the maximum number of possible values within the range acceptable to the customer. We say that i represents the serial number of value a_i in V. Let q be the customer's "ideal" value for this attribute. If $q \in V$, let m be its serial number, i.e. $q = a_m$. Assuming for simplicity that $q \in V$ whenever $a_1 \le q \le a_n$, we define the local similarity measure as follows.

$$S(q, a_i) = \begin{cases} 1 - |i-m| / \max[(n-m+1), m] & \text{if } q \in V \\ 1 - (n-i+1)/(n+1) & q > a_n \\ 1 - i/(n+1) & q < a_1 \end{cases}$$

For example, we wish to find a computer whose *processor speed* is 800MHz, but we are willing to consider speeds between 700MHz and 1000MHz. The possible values of *processor speed* and the corresponding local similarity measure are shown in Table 1. In this case, speeds above 1000MHz have zero similarity, so that $n = 7$, $V = \{700,750,800,850,900,950,1000\}$, $q = 800$, $m = 3$, and therefore $S(q, a_i) = 1 - |i-3|/5$.

4.2. Modification of the Weight

The Global Similarity Measure is computed typically by taking a weighted average of the local similarity measures. The *weights* ω_i provide some indication of the relative importance of the different attributes. These are the quantities that we want to modify when the user is not satisfied with some of the proposed specific attributes. Following a critique from the user, the main task of the system consists of computing how much change we should make to the weight(s) of certain attribute(s) that the user considers to be the most important to him. According to this, we should compute the similarity with the weight, Sim_i, of each attribute using this formula:

$$Sim_i = \omega_i S_i(q_i, c_i)$$

Where q_i and c_i are the i^{th} attributes of the query and the case respectively, and where S_i is the similarity measure between these attributes. Let Sim_* denote the similarity of the criticized attribute. Once the similarity of each attribute is computed, we cover one by one each attribute whose similarity is greater than Sim_*. We reduce the weight of these attributes as follows:

$$\omega_i \leftarrow \omega_i - (Sim_i - Sim_*) \quad \forall i \;\; Sim_i > Sim_*$$

Finally, if there were at least one attribute whose similarity is greater than Sim_*, the weight of the criticized attribute is increased by an amount equivalent to the sum of all reductions to the weights of the attributes that lost importance, so that the sum of all weights stays equal to 1:

$$\omega_* \leftarrow \omega_* + \sum (Sim_i - Sim_*) \quad \forall i \quad Sim_i > Sim_*$$

An example of weight modification is given in Figures 6 and 7 (section 5.3).

4.3. Adaptation

Our PCFinder works for the configuration of personal computers. It is a highly structured domain [15] in which it is possible to subdivide the problems that arise and their related solutions into more or less independent sub-problems and related sub-solutions.

We divide the attributes (also called sub-problems or sub-solutions) in the query and case base into three types. The first type consists of *independent attributes*, whose values can be changed without being restricted by other attributes, such as hard disk drive and computer model. The second type consists of *dependent attributes*, whose values depend on the independent attributes when they need to be changed, such as main memory and CPU. The third type consists of *related attributes*, whose values change automatically when the other attributes change, such as price. Some independent attributes have no dependent attributes associated with them. The relation between these types of attributes is illustrated in Figure 2.

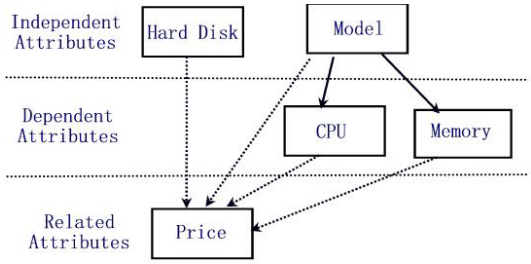

Figure 2. The relationship among independent attribute, dependent attribute and related attribute

Now, we introduce some scenarios about the method of our adaptation.

Adaptation of *Independent Attributes*:

1. Search the capability of independent attribute in the constraint database;
2. If the capability is not less than the customer's requirements, adapt the similar value of the independent attribute to the customer's required value; and adapt the value of the relevant related attribute;
3. Otherwise, return "cannot explicitly be configured".

Adaptation of *Dependent Attributes*:

1. Search the capability of independent attribute on which the dependent attribute depends in the constraint database;
2. If the capability is not less than the customer's requirements (if there is no conflict with the independent attribute), adapt the similar value of the dependent attribute to the customer's required value; and adapt the value of the relevant related attribute;
3. Otherwise, return "cannot explicitly be configured".

For example, if the consumer is satisfied with the *model*, *CPU* and some other attributes but not with *main memory*, the adaptation agent will search the *model* in the constraint database to check if this kind of *model* has the capability to install the desired value of *main memory*. If yes, the adaptation agent will create a new case that satisfies the consumer's demand. Otherwise, it will tell the consumer that the most similar case, which the search agent has found, is the most suitable for him.

We do not recommend adapting *related attributes* because this type of attributes is related to the other attributes. Its value is determined by the other attributes.

4.4. Profile and Collaborative Filtering

The customer's shopping habits, such as preferences or constraints, usually last a long time. We collect them in the *long-term profile*. This provides very important and useful information to us when we try to get the customer's requirement. Each user is associated with a single profile, and each profile contains user information such as personal identification and selection information.

When a consumer configures his computer, he can provide to the search agent a preferred value, a forbidden value, as well as maximum and minimum values for each attribute. For example, a consumer might prefer a notebook computer that has a *processor speed* of 900MHz but not 1000MHz; and he might expect a *price* range between $1500 and $2000. But such preferences are temporary, and therefore we collect them in the *short-term profile* [2].

PCFinder also applies collaborative recommendation based on user profile. A user profile stores the background of an individual user on the server as a profile database. The key issue in recommendation is the ability to combine a target user

with a group of other users that have a profile similar to the target user [6].

The three steps of collaborative recommendation are described as follows:

1. Identify the group in which the given target user belongs.
2. Produce a list of recommendable products or attributes. These products or attributes are ranked according to their appearance in the past purchasing history.
3. Recommend the top *n* recommendable products or attributes.

4.5. Clustering Analysis

Although there exist many kinds of clustering techniques, we use a simple one to illustrate how it supports product recommendation. More specifically, PCFinder groups customers according to the profiles of their background, such as the main intended use of computer and occupation of users, etc. These user's attributes are called *external attributes*. When a new customer signs up, PCFinder provides some suggestions about the computer configurations according to an analysis of the purchasing history of all previous customers who have a similar profile. These computer configurations are called *internal attributes*.

Some cases and solutions about clustering analysis are described as follows:

Case 1: PCFinder suggests the most popularly used internal attribute by clustering their external attributes. For example, PCFinder groups customers who use their computer for playing games and discovers that most of them buy computers whose *processor speed* is 1700MHz. Hence, such computers are recommended to new customers who intend to use their computer for playing games.

Case 2: PCFinder suggests the most popularly used internal attribute and gives this internal attribute a higher weight than the others by clustering their external attributes. For example, PCFinder groups customers who work as university professors and discovers that most of them have bought either Compaq or SONY. Hence, these two brands are recommended to new customers who are university professors, and the initial weight of "*brand*" is increased.

Case 3: This case is more complicated than the first two. In cases 1 and 2, clusters take account of a single attribute. In this case, PCFinder maps groups of related user profiles to groups of related computer configurations. For each new customer, PCFinder finds out to which group he belongs when he fills out his profile form. By analysing cluster correlations between external and internal attributes, PCFinder recommends the configuration of the cluster that correlates best with the customer's external attributes. See Figure 3.

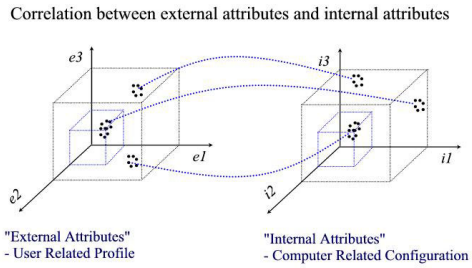

Figure 3. The correlation between external and internal attributes

For example, PCFinder could group customers who are university students, who use computers to play games and whose age is between 21 and 30. Using techniques from Cases 1 and 2 above, PCFinder discovers that most of these customers have bought SONY or Compaq computers whose *processor speed* is 1700MHz, operating under the *Home Edition of Windows XP*. Therefore, these configurations are recommended to new customers who have the same profile as this group.

5. Implementation of PCFinder

In order to illustrate the architecture and methodology of our product recommendation system, we constructed an intelligent agent – PCFinder – that runs on an online notebook computer store to provide suggestions to customers as well as management staff members.

5.1. Running Environment, Developing Tools and Domain

To implement our online computer store with PCFinder, we chose Apache Tomcat 4.0 as our Application Server, which was used by 62% of the websites on the Internet in December 2002 [11]; we use JavaServer Pages and Java Beans as our developing tools. We also apply XML as a Standard Generalized Markup Language, which is transformed to HTML by a XSLT processor. One hundred and fifty cases are stored in a relational database. Each case includes the following eight attributes: processor speed, memory, hard disk drive, display size, multimedia, operating system, brand and price.

5.2. Registration and Constraints Suggestion

When a new customer registers, the system asks him to fill out the form of necessary contact information and optional profile information (See Figure 4). As a first response, the agent provides suggestions about long-term constraints (See Figure 5) according to the profile information. If the customer is not satisfied with these suggestions, he can modify the long-term constraints information by himself. This makes the interaction more comfortable for the user, who does not need to fill in all these data.

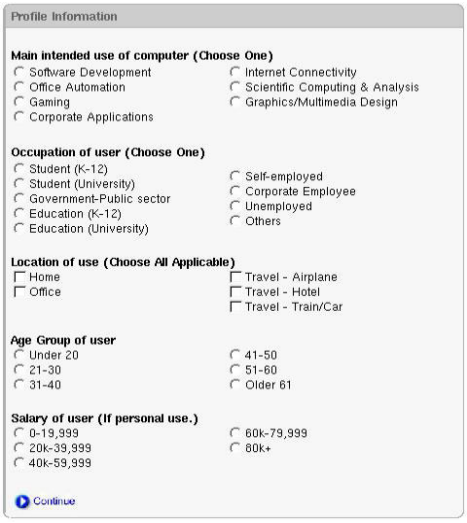

Figure 4. Profile information form

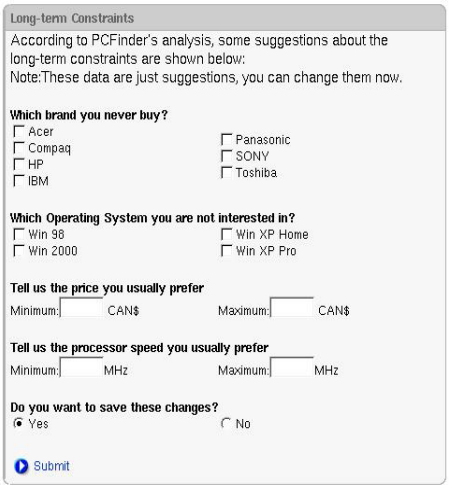

Figure 5. Long-term constraints form

5.3. Product Recommendation

In this section, we explain how the agent achieves the product recommendation process. PCFinder provides two ways to assist the customer in refining the result. The first way is weight modification. The product consists of several attributes. Different attributes have different weight in the customer's mind. According to CBR theory, the product is recommended with respect to the integration of similarities of the whole attributes. One of the possibilities occurs if the most important attribute has the same weight as the other attributes at a time when the customer is unsatisfied with the result. In this case, weight modification can be applied to help the customer find a more satisfactory solution. Consider the situation illustrated in Figure 6, for instance. The solution proposed as "result" is closest in similarity to the customer's "query" among all 150 cases in the case base. But the customer declares himself unhappy with the switch from IBM to Compaq. Therefore, PCFinder reduces the weight of all the attributes that had a local similarity higher than that of the brand (in this case, all the attributes) and increases the weight of "brand". The case base is searched again with these new weights and the best match that is found is illustrated in Figure 7.

Another way to increase the consumer's satisfaction is adaptation of the result. Some attributes of a product can be adapted, others cannot. Therefore, if the customer is still not satisfied with the attributes, some of them can be adapted. In this case, PCFinder helps the consumer in adapting the recommended computer until he is satisfied. In our system, the attributes of memory, hard drive, multimedia and operating system can be adapted. Figure 8 shows the result of adaptation starting from the unsatisfactory solution that was previously offered in Figure 7. It is important to point out that this is the only time that the system allows itself to recommend a product that may not be in the case base, which explains why a solution so close to the consumer's query had not been proposed earlier.

5.4. Graphical Analysis

To visualize the relationship between user profiles (external attributes) and the product attributes (internal attributes), we construct a graphic-building wizard for management staffs' marketing research. After selecting any items (one or more) of the user profiles and any one attribute of the product (See Figure 9), we get a statistical diagram (See Figure 10).

The X-axis represents the selected attribute of the product. The Y-axis represents the quantity of sales. There are two curves in this diagram. For example, the selected item is playing games (the intended use of the computer is to play games); the selected attribute of the

Item Description	Query	Result	Local Similarity	Weight	Select
Processor	1600	1700	0.93	0.125	○
Memory	512	256	0.75	0.125	○
Hard Drive	30	30	1.0	0.125	○
Display	15"XGA TFT	15"XGA TFT	1.0	0.125	○
Multimedia	DVD	DVD	1.0	0.125	○
OS	Win 2000	Win 2000	1.0	0.125	○
Brand	IBM	Compaq	0.5	0.125	◉
Price	2001-2500	2355.95	1.0	0.125	○
Model	-	**EVO N800C P4-1700**	-	-	-
Company	-	**eshop.com**	**Match Rate:**	**89.7%**	-

Figure 6. Before weight modification

Item Description	Query	Result	Local Similarity	Weight
Processor	1600	1600	1.0	0.071
Memory	512	256	0.75	0.093
Hard Drive	30	30	1.0	0.062
Display	15"XGA TFT	15"XGA TFT	1.0	0.062
Multimedia	DVD	CD-RW	0.75	0.062
OS	Win 2000	Win XP Pro	0.33	0.062
Brand	IBM	IBM	1.0	0.522
Price	2001-2500	2313.95	1.0	0.062
Model	-	**THINKPAD A31 P4M-1600**	-	-
Company	-	**eshop.com**	**Match Rate:**	**91.9%**

Figure 7. After weight modification

Item Description	Query	Result	Local Similarity	Weight
Processor	1600	1600	1.0	0.071
Memory	512	512	1.0	0.093
Hard Drive	30	30	1.0	0.062
Display	15"XGA TFT	15"XGA TFT	1.0	0.062
Multimedia	DVD	DVD	1.0	0.062
OS	Win 2000	Win 2000	1.0	0.062
Brand	IBM	IBM	1.0	0.522
Price	2001-2500	2463.95	1.0	0.062
Model	-	**THINKPAD A31 P4M-1600**	-	-
Company	-	**eshop.com**	**Match Rate:**	**99.6%**

Figure 8. After adaptation of the result

product is processor speed. This chart compares the sales between computers intended for playing game and not. According to this analysis, it is easy to know which attribute is more important for the customers in a specific group. Hence, the initial weight value or the recommended product attribute will be modified.

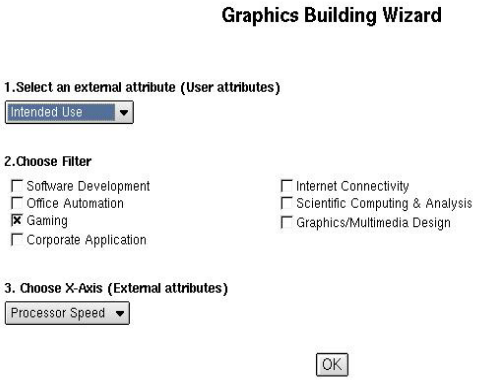

Figure 9. User interface of graphic-building wizard

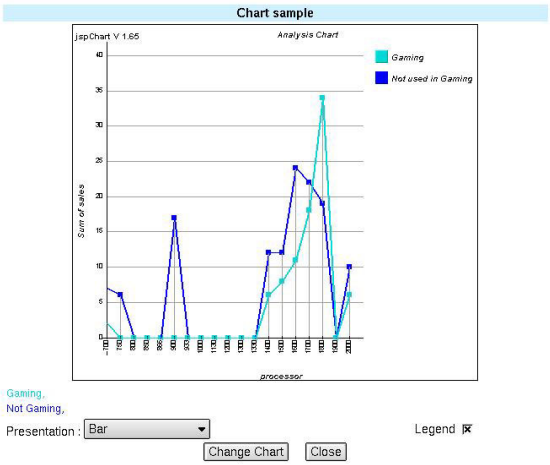

Figure 10. Statistics diagram generated by graphic-building wizard

6. Conclusions and Future Work

In this paper, we started by raising the problem of lack of customer support in electronic commerce applications on the Internet. Then, we provided an overview of some major methods to solve this problem: case-based reasoning, collaborative filtering, and clustering techniques, etc. In particular, systems based on case-based reasoning are a promising

approach for the creation of agents that are able to recommend products according to the specific requirements of customers.

In order to illustrate our methods, we constructed an architecture to an intelligent agent and implemented an online computer store, which includes 150 cases in the case base and provides two kinds of friendly user interface to assist both customers' shopping behavior and management staffs' marketing research.

In order to test the validity of the methods used in our PCFinder, an experiment was carried out. In this experiment, 36 people (of professional and non-professional backgrounds) were invited to evaluate our system. Based on the test results, the following conclusions can be drawn. Most people are satisfied with the Order-Based Similarity Measure, and they also believe that both weight modification and adaptation can improve its performance. Furthermore, applying either short-term or long-term constraints resulted in more satisfactory recommendations (than not applying them).

As for the future work, we intend to make more detailed studies, such as extending our work to more application domains, making PCFinder have the ability to communicate with other agents and to extract semantic information, etc. Therefore, PCFinder can potentially provide a large range of services to customers and increase their satisfaction with existing ones. This results in increased sales, making the company more profitable.

References

[1] Aamodt, A. and Plaza, E., "Case-based reasoning: Foundational issues, methodological variations, and system approaches", *AI Communications,* Vol. 7(1), 1994, pp. 39-59.

[2] Aïmeur, E. and Vézeau M., "Short-Term Profiling for a Case-Based Reasoning Recommendation System", *ECML'2000*, Spain, May 2000, pp. 1-12.

[3] Bergmann, R., Schmitt, S., and Stahl, A., "Intelligent customer support for product selection with case-based reasoning", *E-Commerce and Intelligent Methods*, Physica-Verlag, 2002, pp. 322-341.

[4] Bridge, D., "Product Recommendation Systems: A New Direction", *ICCBR 2001*, R.Weber and C.G.von Wangenheim (eds.), 2001, pp.79-86.

[5] Burke, R., "Knowledge-based Recommender Systems", *Encyclopedia of Library and Information Systems,* A. Kent (ed.), Vol. 69, Supplement 32. New York: Marcel Dekker, 2000.

[6] Cunningham, P., Bergmann, R., Schmitt, S., Traphöner, R., Breen, S. and Smyth, B., "Intelligent Support for Online Sales: The WEBSELL Experience", *ICCBR 2001*, Harbor Center in Vancouver, British Columbia, Canada. 31 July 2001, pp. 87-93.

[7] Guttman, R. H., Moukas, A. G., and Maes, P., "Agent-Mediated Electronic Commerce: A Survey", *Knowledge Engineering Review*, Vol.13, No.2, 1998.

[8] Kumar, V., "Algorithms for Constraint Satisfaction Problems: A Survey", *AI Magazine*, 13(1), 1992, pp. 32-44.

[9] Ma, Y. and Aïmeur, E. "Intelligent Agent in Electronic Commerce XMLFinder", *10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Research, Knowledge Media Networking Workshop*, MIT, Cambridge, MA, June 2001, pp. 273-278.

[10] Moukas, A., "Amalthaea: Information Filtering and Discovery using a Multiagent Evolving System", *Journal of Applied AI*, Vol. 11, No. 5, 1997, pp. 437-457.

[11] Netcraft: http://www.netcraft.com/survey/

[9] Ricci, F., Arslan, B., Mirzadeh, N. and Venturini, A., "ITR : A Case-Based Travel Advisory System", *6th European Conference (ECCBR 2002)*, Aberdeen, Scotland, UK, September 2002.

[13] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach,* Prentice Hall, Upper Saddle River, N.J., 1995, pp. 31.

[14] Shardanand, U., and Maes, P., "Social Information Filtering: Algorithms for Automating 'Word of Mouth' ". *CHI'95 Conference on Human Factors in Computing Systems*, ACM, 1995, pp. 210-217.

[15] Stahl, A. and Bergmann, R., "Applying Recursive CBR for the Customization of Structured Products in an Electronic Shop", *5th European Workshop on Case-Based Reasoning*, Springer Verlag, 2000, pp. 297-308.