[1]THE IMPLEMENTATION OF KNOWLEDGE-BASED RECOMMENDER SYSTEM FOR ELECTRONIC COMMERCE USING JAVA EXPERT SYSTEM LIBRARY

In-Gook Chun, In-Sik Hong

Division of Information Technology Engineering, SoonChunHyang University
Asan-si, ChoongChungNam-do, 336-745, KOREA
chunik@sch.ac.kr

ABSTRACT

This paper deals the design and implementation of product recommender system on a Internet shopping mall. In e-commerce application, sometimes potential buyers may be interested in receiving recommendations about what to purchase. The mainstream of automated recommender system is collaborative filtering. Recently knowledge-based approach is proposed. In this paper, we present a knowledge-based product recommender system. The knowledge base of product domain and inference engine is implemented by JESS and Java servlet. The system gathers the user's requirements on a particular product by questioning the user and consulting its knowledge base to find the items that best meet the user's requirement. For the recommender system, the design of product knowledge base are discussed. As a sample implementation, cellular phone recommender system is built where the system tries to find a phone model that meets user's requirements best.

1. INTRODUCTION

Recommender systems provide recommendations to potential buyers[8][1][4]. Most widely used techniques are collaborative filtering. Collaborate filtering is a technique for recommendation items to a user based on similarities between the past behaviour of the user and that of like-minded peoples. It assumes that human preferences are correlated, thus if a group of like-minded users likes it then a user may also like it. On the contrary, a knowledge-based recommender asks a user about the requirement of wanted products and reasons about what products meet the user's requirements based on the answers. It exploits its knowledge base of the product domain[8].

Each technique has advantages and disadvantages. Collaborative filtering can make personalized recommendations and their prediction quality improves over time as the database of users' preference get larger and larger. In contrast, collaborative filtering needs substantial amount of users' preference database in order to make useful recommendations and if user's preferences change, then it may not recommend useful items because it exploits past data. Knowledge-based approach does not need a initial database of users' preference. For some products like cars, houses, computers, knowledge-based is more appropriate by nature. For example, a user that want to buy a new car, may consider the car's several features like type, size, price more important than what cars others people buy. But In order to make good recommendations, it should have the product domain knowledge and the knowledge should be stored and organized in a inferable way. Thus it needs a knowledge engineer[9].

In this paper, a knowledge-based recommender system is designed and implemented using JESS(Java Expert System Shell), a recent variant of CLIPS that integrates Java object manipulation with rule-based inference[3]. JESS functions as a inference engine, a task that includes processing the products knowledge base, drawing conclusions, and preparing questions for the user in response to user input. JESS supports both forward and backward reasoning and has the full features of a programming language, thus providing flexibility for future development of the recommender system. In addition, if we use full-featured expert system shell, it separates the knowledge base from the programming of the interface, communication, and inference portions of the system. Doing so allows the knowledge base to be

[1] This work is supported in part by the Ministry of Information & Communication of Korea ("Support Project of University Information Technology Research Center" supervised by KIPA)

extended and modified without requiring modifications to the underlying system.

2. ARCHITECTURE

Our architecture consists of a user interface module, a inference engine, a knowledge base of the product domain, a customer database. The user interface module interacts with users. It asks a user about what features of the product they need and collects the answers from users. To the user, our recommender system appears as a guided interrogation supported by images that illustrate answer alternatives.

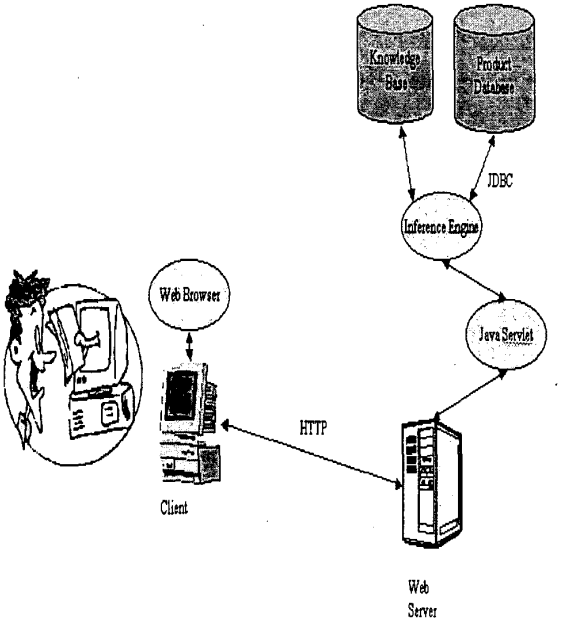

Figure 1. The architecture of the recommender system

A typical recommendation session involves answering about 10 questions, choosing from two to ten alternatives at each point. These user's response data are transferred to the JESS inference engine. The questions concern features of the products being recommended. Each alternatives is illustrated by an image. At the conclusion of the dialog, the recommender system presents the customer with a list of possible products based upon expert opinion. Each product in the final list is ranked with a score that indicates the degree that the product meets the customer's requirements. Customer's information is stored in a database. The database is accessed using the JDBC(Java DataBase Connection) capability of the Java servlet.

3. THE STRUCTURE OF KNOWLEDGE BASE

A product domain knowledge is stored as a form of facts and rules in a JESS knowledge base file. It is somewhat akin to a relational database, especially in that the facts must have a specific structure. As a sample implementation, the product knowledge about cellular phone models is implemented. The knowledge base describes a cellular phone model using a lot of binary and numeric features. The features of the phone model is implemented with the "template" and "slot" concept of JESS[3]. A template is like a class definition in object-oriented languages. In a template definition, many named fields are defined. The fields are traditionally called slots. In our implementation, the cellular phone model has the following template definition.

```
(deftemplate phone
        (slot score (default 0) (type INTEGER))
        (slot picture)
        (slot model)
        (slot service_company)
        (slot manufacturer)
        (slot price (type INTEGER))
        (slot wap (default no))
        (slot email (default no))
        .....
)
```

Each phone model described in the knowledge base has a score slot which registers a matching degree between the customer's requirements and the features of the phone model. If there is a match between the customer's preference and the features of the product, then the score is increased. Otherwise, the score is decreased.

Color information of phone models are encoded a little differently in the knowledge base. Each color is given a color index number. If two colors are similar, then the index numbers of the two colors are also adjacent numbers. For example, white and gray are the similar colors, so they are given the similar numbers, maybe 0 and 1. On the contrary, white and black is considered as totally different colors, so they are given 0 and 15. So if we calculate the absolute value of the difference between customer's preferred color number and the particular product's color number, we can measure the color distance value. If the distance value is large, high penalty value is subtracted from the matching score.

Using above template definition, the knowledge base for a particular phone model can be built by filling necessary slots. For example, if the model has the e-mail capability, the value of "email" slot is yes. If the price of the model is 2000 dollars, then the value of "price" slot is 2000. One example for a particular model is as follows.

```
(phone  (model "LGC-EOF")
        (picture "011-LGC-EOF.gif")
        (service_company 011)
```

(color white) (colornumber 0)
(price 3000) (design round) (manufacturer LG)
(ftype folder) (pim yes) (pclink yes) (mybell yes)
(ear_mic yes)
(power_battery yes) (auto_answering yes)
(recording yes)
)

The above fact tells that the phone's model name is 011-LGC-EOF, it's service company is 011, it's color is white and etc. Currently data of about 100 phone models are stored in our knowledge base file. In addition to the facts, a lot of rules are defined. The rule represents the knowledge of human expert selling cellular phones. An example is that generally younger generations like the round design than the box type. The above knowledge is represented as the following rules.

IF the age of customer is young AND the design of a phone model is round THEN increase the score of the phone model.

The above rule says that if the age of the customer is young, then the score of a round-designed phone models is increased.. The following is a JESS language representation of the above rule.

```
(defrule check_design
 (customer (age young))
 ?a <- (phone (design round))
=>
 (bind ?news (+ 1 ?s))
 (modify ?a (score ?news))
)
```

4. IMPLEMENTATION

The system employs a multimedia user interface that is accessible with standard web browsers. Our recommender system uses some components of the NIH reptile indentification system[7]. Our recommender system is a client/server system, in which the client software supports the user interface and the geographically separated server software supports the expert system. The web server include a standard HTML compatible web server and a servlet engine. The web server handles communications with the client and accesses the images and the HTML documents as necessary. The servlet engine supports the expert system shell. The servlet has the user interface module which is called by JESS program. Firstly a first query screen is generated by JESS program with the help of Java object. All user inputs, which originate with the client software, are received by the web server and are handed in turn to the expert system shell running under the servlet engine. Output generated by the expert system is handed to the web server and then relayed via Internet to the client. Currently the web server is Apache and the servlet engine is ApacheJServ version 1.1b3.

The sequence of query to the customer is defined using a decision tree. The decision tree is defined as rules in the knowledge base file. The first question given to a user are defined as follows.

```
(defrule first-query
 (initial-fact)
=>
 (new QueryPanel "service company." 6
  (create$
   "011" "011.gif" "(service_company 011)"
   "017" "017.gif" "(service_company 017)"
   "016" "016.gif" "(service_company 016)"
   "018" "018.gif" "(service_company 018)"
   "019" "019.gif" "(service_company 019)"
   "dontcare" "char.gif" "(service_company dontcare)"
  )
  (fetch "ReteControl")
 )
)
```

It asks users what mobile service company they prefer. Both QueryPanel and ReteControl are Java classes which are defined in a java servlet code[7]. Within a JESS code, we can make a object of Java class and we can call the methods of the class. The above rule generates the following first query figure. If the customer selects one of the menu items, a new fact reflecting the customer's choice is created and added to the working memory of the expert system.

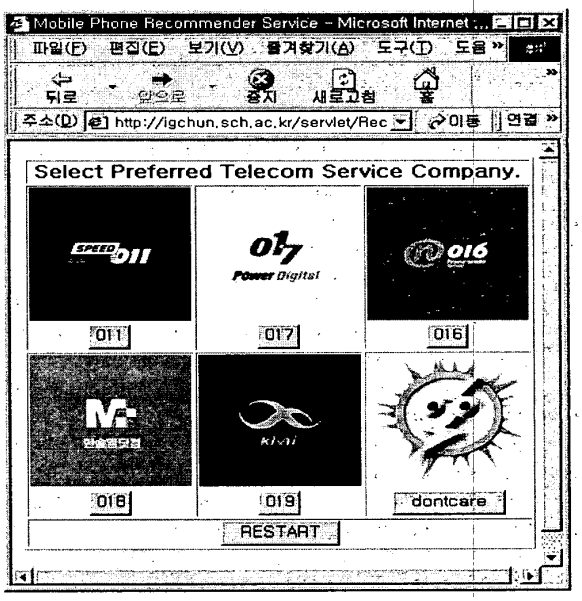

Figure 2. Query screen of the recommender system.

Next query screen is defined in the following rule representing a part of decision tree. The rule fires only if the fact about the service company is activated in the working memory. Later query screens proceed like this.

```
(defrule manufacturer_rule
  (service_company ?type)
=>
  (new QueryPanel "Select Prefered Manufacturer" 8
    (create$
      "Samsung" "samsung.gif" "(manufacturer samsung)"
      "LG" "lg.gif" "(manufacturer LG)"
      "Motorola" "mo.gif" "(manufacturer motorola)"
      "Hyundai" "gul.gif" "(manufacturer hyundai)"
      "Hanhwa" "wchan.gif" "(manufacturer hanhwa)"
      "SK" "sk.gif" "(manufacturer sk)"
      "Sanyo" "sanyo.gif" "(manufacturer sanyo)"
      "dontcare" "char.gif" "(manufacturer dontcare)"
    )
    (fetch "ReteControl")
  )
)
```

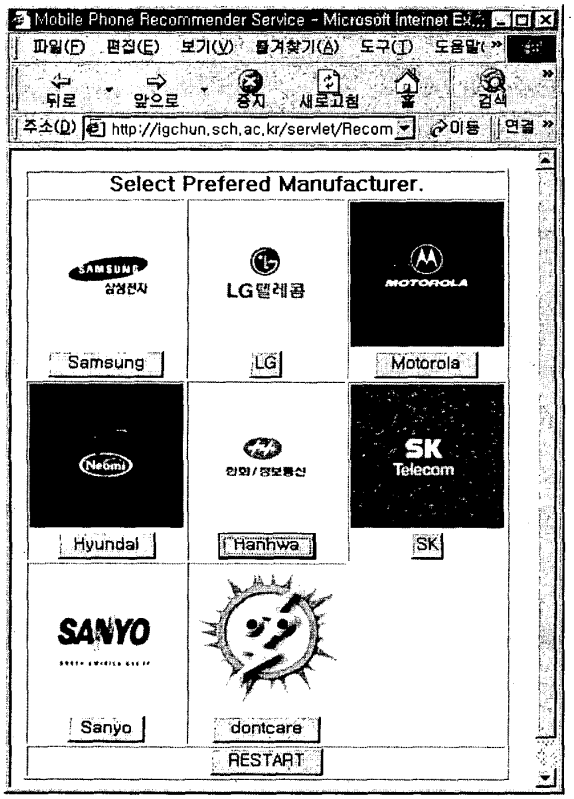

Figure 3. Query screen of the recommender system.

Each phone model described in the knowledge base has a score slot which registers a matching degree between the customer's requirements and the features of the phone model. If there is a match between the customer's preference and the attribute of the product, then the score is increased. Otherwise, the score is decreased. If the user answers as "dontcare", the score is not changed. At the start, all of the phone models are candidates for the recommendation. The facts representing each phone model are created in a working memory. As a customer responds to questions and so the answers of the customer are collected, the score of each phone model is recalculated and changed. After the final question is answered by the user, a competition process is started where only one fact with the maximum score should survive. The survived final fact is presented to the customer as a recommendation product with the matching score.

The score is the measure of similarity and the similarity between the requirements of customers and the features of candidate products is often measured by treating each as a vector and computing the cosine of the angle formed by two vectors. We adopt this formalism to our method. The expression for calculating the score can be given as follows.

$$\cos(\vec{a},\vec{b}) = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\| * \|\vec{b}\|}$$

where $\vec{a},\vec{b}$ denotes a customer vector and a product vector respectively and "·" denotes the dot-product of the two vectors[1][4].

After the all queries are presented to the customer, the recommender system tries to determine which product will meet the customer's requirement best. It is implemented internally as a competing process. Every phone model is a candidate at first. After the customer responds the last question, only a phone model having a high matching score can survive. This competing process is implemented by inserting and retracting the facts related with the phone model to the working memory of the expert system. The system compares the scores of the two facts related with particular models, and retracts the fact with a lower matching score from the working memory. Finally the fact with a highest matching score will survive and it is presented to the customer as a recommendation like the following figure.

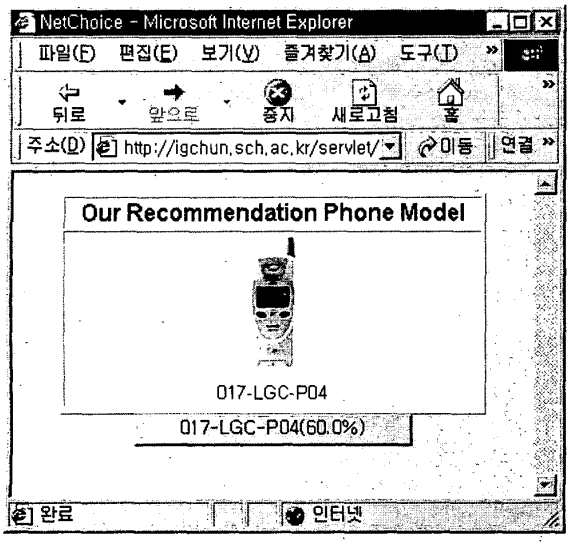

Figure 4. Result screen of the recommender system

5. CONCLUSION

In this paper, a sample implementation of a knowledge-based recommender system for electronic commerce is described. The system consists of the knowledge base of the product domain, inference engine and user interface module. Inference engine is implemented using JESS which is a Java expert system shell. A sample internet shopping mall which sells cellular phones is built and recommender system is implemented on the top of it. One of the possible knowledge base format for the recommender system is described.

Recently the integration effort that combines collaborative filtering and knowledge-based approach is being introduced[2]. So the future work is to build the hybrid recommender system which integrates knowledge-based and collaborative filtering recommender systems using JESS.

6. REFERENCES

[1] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, *"Analysis of Recommendation Algorithms for E-Commerce"*, GroupLens Research Group / Army HPC Research Center, Department of Computer Science and Engineering University of Minnesota, 2000.

[2] Burke, R. "Integrating knowledge-based and collaborative-filtering recommender systems". *AAAI Workshop on Artificial Intelligence for Electronic Commerce WS-99-01*, AAAI Press, Menlo Park, California, 1999.

[3] Ernest J. Friedman-Hill, The JESS User Manual, http://herzberg.ca.sandia.gov/jess/FAQ.html , 2001.

[4] Jack Breese, David Heckerman and Carl Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, WI, Morgan Kaufmann Publisher, July, 1998.

[5] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin (WPI) "Combining Content-Based and Collaborative Filters in an Online Newspaper", *ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[6] Mark O'Conner, Jon Herlocker "Clustering Items for Collaborative Filtering ", ACM SIGIR '99 Workshop on Recommender Systems, August 19, 1999.

[7] Ralph F. Grove and Arthur C. Hulse, *"An Internet-Based Expert System fro Reptile Indentification,"* http://herzberg.ca.sandia.gov/jess/, 1998.

[8] Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P. and Riedl, J., Grouplens, "An open architecture for collaborative filtering of netnews". *Proceedings of the ACM Conference of Computer Supported Cooperative Work*, pp. 175-186, 1994.

[9] Thomas Trans and Robin Cohen, "Hybrid Recommender Systems for Electronic Commerce", University of Waterloo, 1999.