

Product Information Meta-search Framework for Electronic Commerce Through Ontology Mapping

Wooju Kim¹, Dae Woo Choi², and Sangun Park³

^{1,2}Department of Information Industrial Engineering, Yonsei University,
134 Shin-Chon Dong, Seoul 120-749, Korea
wkim@yonsei.ac.kr, qorwkr@nate.com

³Department of Management Engineering, KAIST,
207-43 Cheongryang, Seoul 130-012, Korea
mascon@kgs.m.kaist.ac.kr

Abstract. The Semantic Web and Web services provide many opportunities in various applications such as product search and comparison in electronic commerce. We implemented an intelligent meta-search and comparison system for products through consideration of multiple attributes by using ontology mapping and Web services. Under the assumption that each shopping mall offers product ontology and a product search service with Web services, we proposed a meta-search framework to configure a customer's purpose, make and dispatch proper queries to each shopping mall, evaluate search results from malls, and show the customer the product list with a ranking. Ontology mapping is used for generating proper queries for malls that have different taxonomies of product categories. Also we implemented an inference based search engine using ontology and Web services for each mall.

1 Introduction

The Semantic Web and Web services provide many opportunities in various applications using the Internet. We expect that product search and comparison in electronic commerce will be one of the killer applications of such technologies. So far most product comparison systems over multiple shopping malls depend on a keyword-based search or manual collection of comparison information [13]. The meta-search engine [6, 10, 16] is one of those systems. However, shopping malls are offering their own search and directory service for their products in their Web pages, so the keyword-based meta-search for those malls may become redundant. Moreover, it brings inaccurate results because of difficulties of natural language processing [9].

Another problem of the keyword-based search is that it is very hard to configure a customer's exact preference for the product. For example, keywords are not enough to describe a customer's priorities and criteria for a product attribute such as 'a television whose monitor size is larger than 15 inches'. Therefore, it is difficult to provide an exact matching product to the customer because of incomplete representations of a customer's preference.

One more problem is that we can hardly evaluate and compare products with a keyword-based search from the customer's perspective. There have been increasing

efforts [1, 2, 4] in industrial and academic areas to overcome the above problems, but they still have limitations. For example, Mysimon.com categorizes and sorts products by various attributes in order to help customers easily understand product information. But it is difficult for a customer to sort and compare products based on multiple attributes at the same time.

We want to note that Web services can provide a new framework of the meta-search engine. Amazon.com is offering Web services that allow a variety of functions from retrieving information about products to adding an item to a shopping cart. As more shopping malls are expected to offer Web services, the meta-search engine will be able to use Web services to collect structured product information. Especially when the customer wants to compare products based on various attributes rather than simple price, Web services will be the most appropriate alternative for the meta-search infrastructure.

However, there are still some problems in making and sending a proper query that represents customer preference of a product for each shopping mall, integrating and comparing the results from various malls with Web services. These problems result from different taxonomies of shopping malls for product category names and product attributes. To overcome the problem of different taxonomy, the meta-search system needs to support semantic interoperability among malls. While the standardization and integration of ontology are still progressing, it seems impossible that a single taxonomy will be used in every mall. Therefore, we employed ontology mapping techniques [3, 5, 8] to assure semantic interoperability among malls. Ontology mapping is the process in which we start with two source ontologies and generate a new ontology that includes and reconciles all the information from the two source ontologies according to those semantic relations [12].

In this paper, we will suggest a meta-search and comparison framework under which we can consider multiple attributes of a customer's decision making when buying. This framework uses Web services of shopping malls to acquire structured product information, and compare products based on multiple attributes. Also we use ontology mapping to handle different taxonomies. We named the meta-search and comparison system Intelligent Product Information Search (in short, IPIS).

Section 2 briefly describes the architecture of IPIS. Section 3 explains the intelligent product search and comparison procedure using IPIS. Section 4 shows the prototype and search results. The last section provides conclusions.

2 Overview of the IPIS System

The objective of our research is to develop a meta-search framework capable of configuring a customer's preference and capable of finding and evaluating products based on the customer's preference from various shopping malls. To achieve this goal, the IPIS system should guarantee not only syntactic interoperability but also semantic interoperability between IPIS and shopping malls. The structure is as follows:

2.1 Syntactic Interoperability with Web Services

A keyword-based search cannot thoroughly interpret the contents of Web pages due to the limitation of natural language processing. Therefore it is quite hard to get

structured product information by a keyword-based search. But Web services can provide a method to gather structured information represented with XML from various malls. Syntactic interoperability in our approach means that we can get structured product information by sending a structured query to each shopping mall. Also we can easily construct a meta-search engine by using Web services because it is easy to send queries to and receive responses from multiple shopping malls with Web services. But, syntactic interoperability does not deal with matters of semantic interpretation [18]. Therefore even if we get structured product information, it does not guarantee that we can interpret the content when shopping malls use different taxonomies. With this regard, semantic interoperability is required to interpret the product information.

2.2 Semantic Interoperability with Ontology Mapping

Semantic interoperability is “the ability of information systems to exchange information on the basis of shared, pre-established and negotiated meanings of terms and expressions” [18]. Ontology mapping is one of the techniques for semantic interoperability. The difference of taxonomies among shopping malls brings about the following problems.

The first problem is the difference of product category names. For example, in the case of ‘television’, it is represented as ‘television’ in Open Directory Project taxonomy [14], while represented as ‘TV & HDTV’ in Amazon.com. The second problem is the difference of product attribute names. In addition to product names, product attributes are different in each shopping mall. The third one is the difference of data types and units of product attributes. For the same product attribute, each shopping mall may use different data types or units.

The purpose of using ontology mapping in our approach is to interpret different product category names, attribute names, data types and units of attributes. Also we are going to compare products based on that interpretation. Therefore, ontology mapping is separated into product category name mapping and attribute name mapping in the IPIS system.

2.3 Architecture of the IPIS System

IPIS System architecture (see Figure 1) consists of four key components. First, the *Interface* represents a customer’s search purpose in a structured format. The customer can input the desired product, attributes, and priorities of attributes in the *Interface*. The *Query Generator* transforms the customer’s configuration into queries for each shopping mall by using ontology mapping because shopping malls have different taxonomies. Shopping malls process the queries to search for proper products satisfying the customer’s input and return detailed information of the products to the *Product Evaluation Agent* through Web services. The *Product Evaluation Agent* evaluates and compares the results from shopping malls and shows the customer the list of products with a resulted ranking and attributes. Detailed functions of each module will be described in section 3.

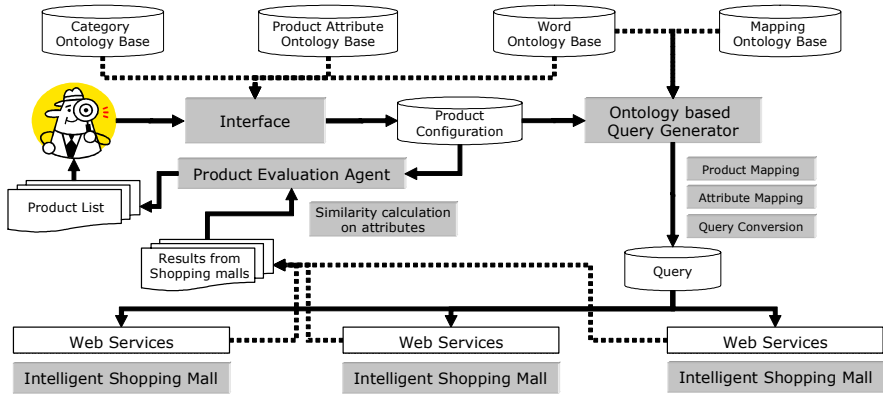


Fig. 1. Architecture of the IPIS System

2.3.1 Interface

The *Interface* is used for the precise configuration of a customer’s preference for a product. In order to represent a customer’s configuration, we used the Open Directory Project as the category ontology base and product attribute ontology base. The Open Directory Project (in short, ODP) is the largest, most comprehensive human-edited directory of the Web. It is constructed and maintained by the vast, global community of volunteer editors [14]. Figure 2 shows an example of product categories and attributes related to ‘Consumer Electronics’ in ODP. The customer can select the desired product in a given product category hierarchy. The customer can also input product attributes, constraints, and priorities by using the *Interface*.

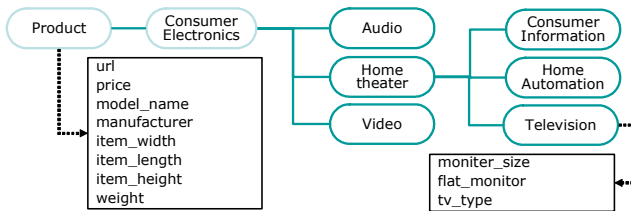


Fig. 2. Product Category and Attribute Example of the Open Directory Project

2.3.2 Query Generator

The *Query Generator* is the most important module in the IPIS system because an appropriate query for each shopping mall can give an appropriate answer. To make an appropriate query, the IPIS system transforms an original query generated from the customer’s configuration into individual queries for each mall because they are using different taxonomies for a product category. Figure 3 shows an example of original query and modified queries for each mall which are represented with RDQL [15].

The mapping ontology base is a mapping table constructed with matching product category names and product attribute names of shopping malls for the words of the ODP ontology. The purpose of the mapping ontology base is to enhance ontology mapping speed. The word ontology base is used to extend product category names

and attributes in ontology mapping. We constructed word ontology with the WordNet [11] ontology.

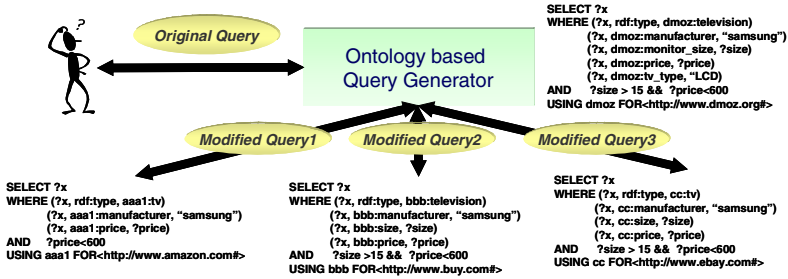


Fig. 3. An Example of Interpreted Queries for Various Malls

2.3.3 Product Evaluation Agent

The *Product Evaluation Agent* analyzes the product list from shopping malls, evaluates the products based on the customer’s configuration, and makes the sorted product list for the customer. The *Product Evaluation Agent* consists of two modules. The first module extracts the values of product attributes from the product list, and the second module evaluates the products by calculating similarities of extracted attributes based on the customer’s configuration.

2.3.4 Intelligent Shopping Mall

The *Intelligent Shopping Mall* receives queries from the *Query Generator* and returns the search results to the *Product Evaluation Agent* after query execution. The search system of the *Intelligent Shopping Mall* exploits an inference based search engine as one of the Semantic Web search engines that can perform various searches on product properties like a database query.

Figure 4 shows the architecture of an intelligent shopping mall using Web services. The inference based search engine executes the query received from the IPIS system. The category ontology base and the product ontology base store product category ontology and product attribute ontology respectively. Rules for inference which describe the relation between categories and attributes are stored in the rule base which consists of Jena rules for OWL axioms for reasoning [7, 17]. The product data base stores detailed product information of the shopping mall.

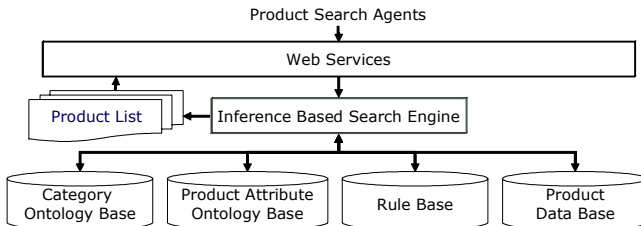


Fig. 4. Architecture of an Intelligent Shopping Mall using Web Services

3 Intelligent Product Information Search Procedure with Examples

This section describes the product search procedure of the IPIS system. The IPIS procedure consists of four steps: configuration of a customer's purpose, query generation, query execution, and product evaluation.

For a better understanding of the procedure, let us demonstrate the whole process with a consistent example of product search. In our example, the customer's search purpose is to find a television whose price is less than \$600, monitor size is larger than 15 inches, TV type is 'LCD' and manufacturer is 'Samsung'.

3.1 The Configuration of a Customer's Search Purpose

Let us describe the configuration process. First, a customer selects the desired product from the category ontology base, and then inputs attributes and his own priorities of attributes. For example, a customer selects 'television' from category ontology and selects 'monitor size', 'manufacturer', 'tv type', and 'price' as concerned attributes for decision-making. Then, the customer inputs the criteria of attributes such as '> 15 inches' for 'monitor size' and '< \$600' for 'price'. Finally, the customer assigns his personal priority to attributes with numbers between 1 and 10 such as 8 for 'price' and 10 for 'monitor size' which means that 'monitor size' is more important to him than 'price'. We employ a semantic tree structure to represent the customer's configuration including product category, names, attributes, criteria, and priorities of attributes in structured form. Figure 5 shows an example of the customer's configuration with a semantic tree structure.

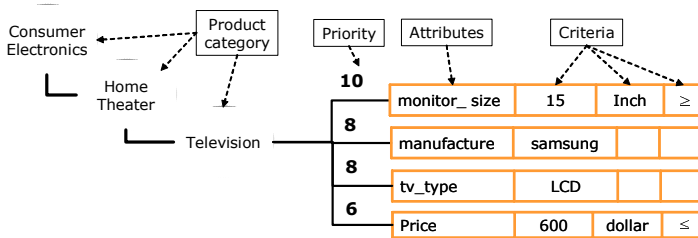


Fig. 5. An Example of a Semantic Tree for Customer Configuration

3.2 Query Generation using Ontology Mapping

An RDQL query can be generated from the semantic tree that represents the configuration of a customer's search purpose as follows.

```
SELECT ?x
WHERE (?x, rdf:type, dmoz:television)
      (?x, dmoz:manufacturer, "samsung")
      (?x, dmoz:monitor_size, ?size)
      (?x, dmoz:price, ?price)
      (?x, dmoz:tv_type, "LCD")
AND   ?size > 15 && ?price < 600
USING dmoz FOR <http://www.dmoz.org#>
```

The purpose of the query is to ask shopping malls about product information with the customer’s configuration. But each shopping mall may not give an appropriate answer to the above query because it uses a different taxonomy from ODP. Therefore, it is necessary to interpret the current query into the form that each shopping mall can understand.

To solve the above problems, the query generation procedure consists of three modules: *Product Mapping*, *Attribute Mapping*, and *Query Conversion* as shown in the flow chart of Figure 6. Let us describe the detailed procedure of *Query Generation*.

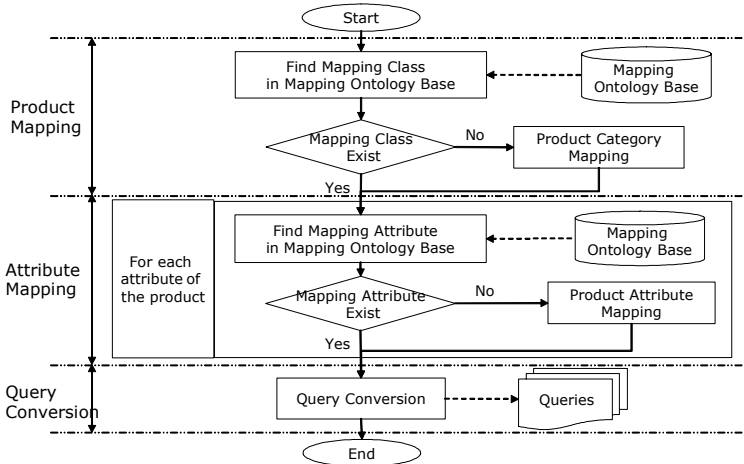


Fig. 6. A Flow Chart for the Query Generation Procedure

3.2.1 Product Mapping

The first step of *Product Mapping* searches for the matching product category name of each shopping mall in the mapping ontology base. If a matching product exists in the mapping ontology base, the time and cost of product mapping can be saved. In our example, there was no exact matching name for ‘television’ in the mapping ontology base. If there is no exact match like in the example, the module looks up the matching product category name from each shopping mall’s ontology in real time.

The procedure of searching for a similar product category name cannot be completed with simple string matching because each shopping mall has a different taxonomy from that of ODP. For example, the product represented as ‘television’ in ODP is represented in Amazon.com as ‘TV & HDTV’. In this case, we can solve the problem by using synonyms. But unconditional use of synonyms and similar words can cause a risk of selecting unexpected products because a word may have different meanings. For example, there are three different meanings of ‘television’ in WordNet. The extension of the product category name uses synonyms and coordinate terms in WordNet. But synonyms and coordinate terms can also be changed as the meaning of the product category name is changed. Therefore we need to determine the exact meaning of the product category name in the configuration and extend the product category name with appropriate synonyms and coordinate terms.

Product Mapping consists of detailed steps like analysis, extension, searching, and choosing. Let us describe these detailed steps.

(1) Analysis and extension of the selected product category name: We used hypernyms of the product category from WordNet to analyze the exact meaning of the product category name. In the example of ‘television’, to find out which one is right for the customer’s configuration among three meanings, we compared a serial category hierarchy that starts from the product category to the root category in ODP to the hypernym hierarchies of three different senses of the product category name in WordNet. Figure 7 shows the category hierarchy of ‘television’ in ODP and three category hierarchies of different meanings for ‘television’ in WordNet.

In Figure 7, WordNet hierarchy 2 has the same word ‘electronic’ in the middle level ‘electronic equipment’ with ‘Consumer electronics’ in the ODP hierarchy. In this manner, the analysis algorithm chooses WordNet hierarchy 2 as the exact meaning of ‘television’.

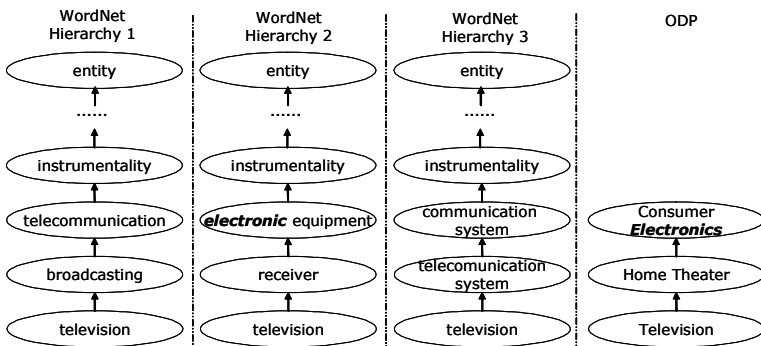


Fig. 7. Comparison between Category Hierarchies of WordNet and ODP

Once the exact sense of the product category name is determined, it extracts the coordinate terms of the selected meaning as an extension set of the product category name from WordNet. For example, the module brings {television receiver, television, television set, tv, tv set, idiot box, boob tube, telly, goggle box} from the WordNet ontology as an extension set of ‘television’.

(2) A Search for Matching Product Category Name: A search starts on each mall’s ontology with an extension set of ‘television’ that was generated in the previous step. We got ‘TV & HDTV’, ‘Internet TV’, and ‘All TVs’ from Amazon.com ontology. After the completion of search for category names, we need to delete redundant category names for the product. To do this, the algorithm generates serial hierarchies for category names by extracting all upper categories starting from the obtained category to the root category. We generated three hierarchies in our example as follows:

```
/Product/Electronics/Gadgets/Internet Appliances/Internet TV
/Product/Electronics/Audio & Video/TV & HDTV
/Product/Electronics/Audio & Video/TV & HDTV/All TVs
```


From the above result, we can identify that ‘All TVs’ is a subcategory of ‘TV & HDTV’. In this case, we delete the subcategory from the list to avoid a redundant search.

(3) Choice of Matching Product Category Name: After the completion of the search for matching product category names, we should decide which product category names are appropriate for the original product name. We compared the hierarchy of ‘television’ in ODP with the hierarchies of product category names in Amazon.com as in Figure 8 to choose proper names.

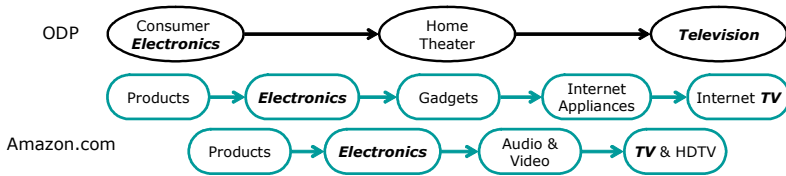


Fig. 8. A Comparison between Hierarchies of Product Category Names and ODP

With the direct comparison of words in the upper categories, we may not decide which one is more appropriate because both hierarchies have ‘electronics’. To find a more appropriate name, we extended upper category names by using synonyms and coordinate terms in WordNet ontology. From the result of upper category name extension, we found that ‘Audio & Video’ in the Amazon.com ontology is similar to ‘Home Theater’ in ODP. So, we selected ‘Home Theater’ as the final matching product category name. We constructed two measures - co-occurrence and order consistency - to estimate the similarity of the product category name to the product name in ODP. Co-occurrence is the calculation of all the similarities of product names by using the similarity of each category name in each category hierarchy. Order consistency is the calculation of the similarity based on comparison between a category order of the ODP hierarchy and category orders of the two hierarchies in Amazon.com. Even if two hierarchies have the same categories, they may have a different similarity along their category order.

Once *Product Mapping* is completed, the matching product category name is stored in the mapping ontology base to enhance next search and avoid the repetition of the same procedure for the same product category name.

3.2.2 Attribute Mapping

After the completion of *Product Mapping*, the *Attribute Mapping* module searches for matching attribute names in a similar manner.

The *Attribute Mapping* procedure is almost the same as the *Product Mapping* procedure except *Attribute Mapping* does not consider the hierarchy of attribute names because there is no attribute hierarchy in the shopping mall ontology. Therefore the similarity is calculated based on only the attribute name. First of all, the attribute mapping module tries to find a matching attribute name from the mapping ontology base. If it fails, the module starts the main attribute mapping procedure by finding matching attribute names in each shopping mall’s ontology. If it also fails, then it extends the attribute name by using the WordNet ontology like *Product Mapping* as

shown in section 3.2.1, and searches for similar attribute names in each shopping mall's ontology with synonyms and coordinate terms of the original attribute name in the WordNet ontology. If an appropriate attribute name was found, it stores the name in the mapping ontology base for the next search. For example, some attributes like 'price' and 'manufacturer' can be directly found in the Amazon.com ontology. But the attribute 'monitor_size' of ODP can be matched to 'screen_size' of the Amazon.com ontology by using attribute name extension with synonyms and coordinate terms.

If the *Attribute Mapping* module cannot find a matching attribute name from a shopping mall's ontology, then it deletes the parts of the query concerning the attribute name from RDQL query. In our example, the attribute 'tv_type' was deleted because there was no matching attribute in Amazon.com.

There are additional steps of *Attribute Mapping* in order to solve the difference of data types and units. In the data type transformation step, we classified data type into two types - string type and numeric type. Next we converted each type to the opposite type if it is needed. For example, if the attribute 'screen_size' is the string type in Amazon.com and the corresponding part of the query in the customer's configuration is 'monitor_size', then we convert the type screen_size to numeric type and make a new query for Amazon.com.

When the unit of an attribute is different, we applied pre-built unit changing rules to change the unit of the attribute. For example, one shopping mall may use 'lbs' as the unit of weight though a customer uses 'kg'. In this case, the unit can be transformed to the other unit by using pre-built unit changing rules.

3.2.3 Query Conversion

As the last step of query generation, the *Query Generator* generates queries for each shopping mall after the completion of all mappings. Figure 9 shows the original query from the customer's configuration and the interpreted query for Amazon.com after the completion of *Product Mapping* and *Attribute Mapping*. The product name was converted to 'TV & HDTV' from 'television' and the attribute 'monitor_size' was changed to 'screen_size'. The attribute 'tv_type' and its value 'LCD' are deleted from the query because there was no matching attribute name to 'tv_type' in the Amazon.com ontology.


<pre>SELECT ?x WHERE (?x, rdf:type, dmoz:television) (?x, dmoz:manufacturer, "samsung") (?x, dmoz:monitor_size, ?size) (?x, dmoz:price, ?price) (?x, dmoz:tv_type, "LCD") AND ?size > 15 && ?price < 600 USING dmoz FOR <http://www.dmoz.org#></pre>		<pre>SELECT ?x WHERE (?x, rdf:type, amazon:TV&HDTA) (?x, amazon:manufacturer, "samsung") (?x, amazon:screen_size, ?screen_size) (?x, amazon:price, ?price) AND ?screen_size > 15 && ?price < 600 USING amazon FOR <http://www.amazon.com#></pre>
--	---	--

Fig. 9. An Example of Query Conversion

After generating queries for each shopping mall, the *Query Generator* sends modified queries to corresponding malls.

3.3 Query Execution in Shopping Malls

Query Execution is performed in Web services based shopping malls that receive queries from the *Query Generator*. The search procedure of *Query Execution* depends on each shopping mall's policy and system. For example, some shopping malls may use a keyword-based search to answer the query, and other shopping malls may convert the RDQL query to a database query and get the answer from a database. As one of the alternatives for search engines, we constructed an inference-based product search engine that performs a search by using the product category and attribute ontology, a rule base, and product data base. We used Jena API [7] to implement the inference based product search engine.

The objective of the inference-based product search engine is to perform a query execution based on various attributes like a data base query execution. To achieve this objective, we translated the content of the product data base into an RDF file and executed an RDQL query on the translated RDF file. But we cannot perform accurate query execution based on attributes with direct translation of the data base. For example, if required information is separately stored in two tables instead of one table, we should integrate the information of two tables by joining them. In order to join the two tables, we need rules about the relation between the two tables like the foreign key constraint in DBMS. The rule base stores such rules that are required to generate an RDF file on which an RDQL query is executable. Those rules define the relationship between product categories and attributes. Figure 10 shows the architecture of the inference based product search engine. The *Rule Reasoner* translates the contents of the product data base into an RDF file by using the ontology base, rule base, and the product data base. The *Query Reasoner* performs the given RDQL query on the RDF document and other ontology bases, and completes the search on various attributes. The search result is returned to the IPIS system through Web services.

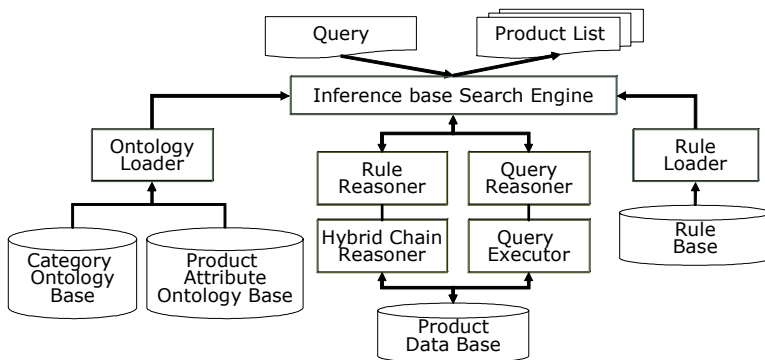


Fig. 10. Architecture of Inference Based Product Search Engine

3.4 Evaluation of Products

The *Products Evaluation* procedure consists of a product information analysis and similarity calculation. In the product information analysis step, the module receives the product information through Web services from shopping malls. Then it analyzes

the product information and extracts attribute names and values of each product. The module uses automatically generated RDQL queries to extract product information because the messages through Web services are represented with OWL. As the last step of the product information analysis, we calculated the distances of every attribute between the customer's configuration and the tracked product. The distance of each attribute is calculated in consideration of the attribute name, value, and criteria. The algorithm calculates the distance for attribute name and value by using the difference of strings and numbers. For criteria, the distance is 0 if the value satisfies a criterion of the attribute, 1 if not. When a unit is different, the module changes the unit with translation rules to calculate the distance. The distance of each attribute becomes normalized between 0 and 1.

In the next step, the similarity calculation module integrates every distance of the attributes to calculate the similarity of each product. We used a general similarity calculation method to evaluate products. The customer's priorities for each attribute are changed to attribute weights. We calculated the weighted sum of distances with the distances and weights of every attribute. Finally, we ranked products by their similarities to the customer's configuration. After the completion of *Product Evaluation*, it prints an ordered product list for the customer, as illustrated in Figure 11.

4 Implementation and Results

We developed an IPIS prototype which consists of an *Interface*, *Query Generator*, and *Product Evaluation Agent*. We also constructed two prototypes of intelligent shopping malls using Web services by implementing an inference based search engine and ontology with the information from Amazon.com and Buy.com. Therefore, users can search for and compare products of Amazon.com and Buy.com by using our prototype.

4.1 Implementation

Figure 11 shows the example of product search results and the functions of each small window in the IPIS system which is implemented using Java. In Figure 11, part number 1 is the window where a customer can choose a product category from a tree formed hierarchy. The hierarchy for the selected category appears in part number 2. The customer can input product attribute names, values, priorities, types and units in part number 3. The attributes that the customer can select are determined by the selected product category. For example, if a customer selects 'television' as product category, the attributes that the customer can select for 'television' are 'manufacturer', 'price', 'monitor_size', etc. Part number 4 shows a product list with the rank that is found with the configuration in part 3 from Amazon.com and Buy.com.

If a customer wants to add new criteria on the attribute 'monitor_size' to the configuration, the customer can pop up an edit window by clicking the 'Edit' button in part 3. Figure 12 shows the edit window to set an attribute's name, value, type and priority that represent the criterion 'monitor_size >= 15 (inch)'. The customer can assign priority with a number between 1 and 9.

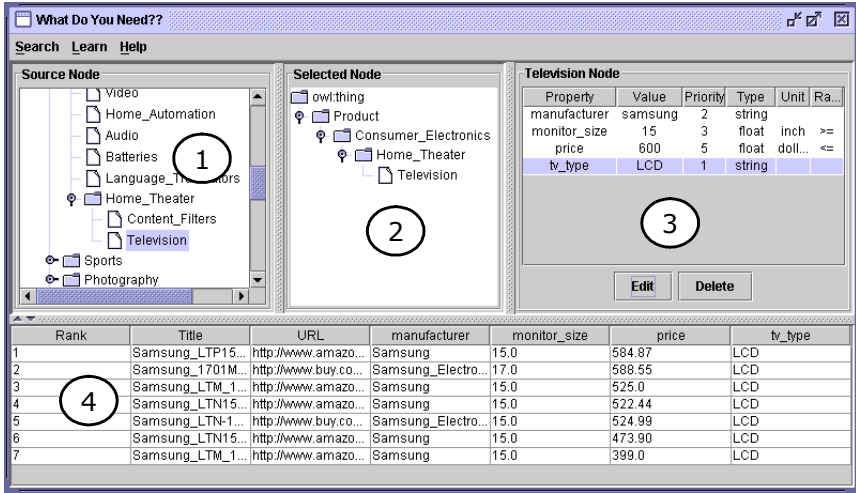


Fig. 11. Product Search Results in the IPIS System

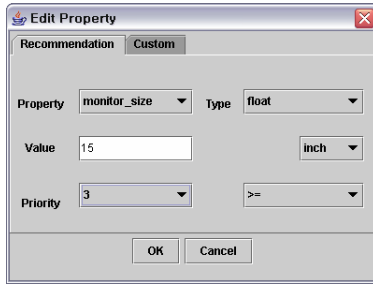


Fig. 12. An Edit Window for Attribute Input

4.2 An Example Comparing the IPIS System and the Keyword-Based Search

Table 1 shows an example that compares search results from the IPIS system and keyword-based search results from Amazon.com. The target product of the search is ‘television’ whose ‘manufacturer’ is ‘Samsung’, ‘tv_type’ is ‘LCD’, price is less than \$600, and ‘monitor_size’ is larger than 15 inches. From the experiment, we were able to find five products satisfying all criteria from Amazon.com with the IPIS system.

In the keyword-based search of Amazon.com, we made keywords as ‘television Samsung LCD 600 15’ to represent the target product. However there was no retrieved product when we included the desired price in the keywords as noted in the second and third row of Table 1. Therefore we excluded the price value and the monitor type value from the keywords, and we could get 19 products. In order to examine the accuracy of the keyword-based search, we manually checked each product to see if the product satisfied the customer’s configuration. We found 4 satisfactory products among them. When we added monitor size shown as the fifth row of Table 1, we got

3 satisfactory products among 4 products in total. In this example, the IPIS system was more accurate than the keyword-based search. Therefore we expect that customers using the IPIS system can save the time required for various keyword-based search trials to acquire accurate results.

Table 1. Product Search Results in the IPIS system and Amazon.com

Result of IPIS	Keyword Based Search Results of Amazon.com		
# of products	Keywords	# of products	# of products satisfying criteria
5	television samsung LCD 600 15	0	0
	television samsung LCD 600	0	0
	television samsung LCD	19	4
	television samsung LCD 15	4	3

4.3 Discussion

One of limitations in our system is that we assumed every shopping mall uses Web services. In a commercialized system, the IPIS system should support traditional shopping malls which are based on HTTP. Also, most shopping malls do not provide their own ontologies for product category names and attribute names. Actually we constructed each shopping mall's ontology by collecting required product information from their Web pages.

In addition, we did not prove that our IPIS system was more accurate and efficient than the traditional keyword-based meta-search engine because the experiment was limited. We just tried to show an example of a better case. We are planning to conduct a more precise experiment to prove the advantages of the IPIS system.

5 Conclusion

We have designed and implemented an intelligent product information search framework using ontology mapping and Web services which has a taxonomy free architecture for meta-search and comparison. Although shopping malls of our systems have the same inference system, our architecture can be easily extended to heterogeneous systems by using the Semantic Web and Web services. We expect that our system will show the opportunities and challenges of the Semantic Web and Web services in electronic commerce.

Web services can be actively utilized in the area where information should be exchanged repeatedly and periodically such as in B2B exchange. However the difference in ontologies can be an obstacle of information exchanges. Ontology mapping is one of the techniques that solve the semantic difference. We expect that the integration of Web services and ontology mapping will be a powerful solution for automatic information exchanges between software programs and agents.

Acknowledgements. This work has been funded by the University Fundamental Research Program of the Ministry of Information & Communication in Korea

References

1. Ackerman, M., Billsus, D., Gaffney, S., Hettich, S., Khoo, G., Kim, D.J.: Learning Probabilistic User Profiles. *AI Magazine*, Vol. 18. No. 2 (1997) 47-56.
2. Aridor, Y., Carmel, D., Lempel, R., Soffer, A., Maarek, Y. S.: Knowledge Agents on the Web. *Lecture Notes in Computer Science*, No. 1860 (2000) 15-26.
3. Benetti, H., Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: An Information Integration Framework for E-Commerce. *IEEE Intelligent Systems*, Vol. 17. No. 1 (2002) 18-25.
4. Chen, Z., Meng, X., Zhu, B., Fowler, R.H.: WebSail: From On-line Learning to Web Search. *Knowledge and Information Systems*, Vol. 4. No. 2 (2002) 219-227.
5. Ehrig, M., Sure, Y.: *Ontology Mapping - An Integrated Approach*. *Lecture Notes in Computer Science*, No. 3053 (2004) 76-91.
6. Howe, A. E., Dreilinger, D.: Savvy Search: A Metasearch Engine that Learns which Search Engines to Query. *AI Magazine*, vol. 18. no. 2 (1997) 19-25.
7. Jena 2 Inference Support. <<http://jena.sourceforge.net/inference/index.html>>.
8. Kalfoglou, Y., Schorelmmmer, M.: Ontology mapping: the state of the art. *The Knowledge engineering review*, Vol.18. No.1 (2003) 1-32.
9. Lawrence, S., Giles, C.L.: Accessibility of Information on the Web. *Nature*, Vol. 400 (1999)107-109.
10. Lawrence, S., Giles, C.L.: Context and Page Analysis for Improved Web Search. *IEEE Internet Computing*, Vol. 2. No. 4 (1998) 38-46.
11. Miller, G. A., "WordNet a Lexical Database for English," *Communications of the ACM*, vol. 38, no. 11, 1995, pp. 39-41.
12. Noy, N.F., Musen, M.A.: The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. *International Journal of Human-Computer Studies*, Vol. 59. No. 6 (2003) 983-1024.
13. O'Keefe, R. M., McEachern, T.: Web Based Customer Decision Support Systems. *Communications of the ACM*, Vol. 41. (1998) 71-78.
14. Open Directory Project. <<http://www.dmoz.com>>.
15. Seaborne, A.: RDQL - A Query Language for RDF, W3C Member Submission. (<http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>) (2004).
16. Selberg, E., Etzioni, O.: The MetaCrawler Architecture for Resource Aggregation on the Web. *IEEE Expert*, Vol. 12. No. 1 (1997) 11-14.
17. Smith, M.K., Welty, C., McGuinness, D.: OWL Web Ontology Language Guide, W3C Recommendation. <<http://www.w3.org/TR/owl-guide/>> (2004).
18. Veltman, K. H.: Syntactic and Semantic Interoperability: New Approaches to Knowledge and the Semantic Web. *New Review of Information Networking*, Vol. 7 (2001) 159-184.