International Journal of Information Technology and Knowledge Management
July-December 2010, Volume 2, No. 2, pp. 196-199

A NOVEL APPROACH FOR AUTOMATIC DETECTION AND UNIFICATION OF WEB SEARCH QUERY INTERFACES USING DOMAIN ONTOLOGY

Anuradha[1] & A.K Sharma[2]

A large amount of information on the Web, stored behind search interfaces cannot be indexed by general-purpose search engines as it is dynamically generated through querying databases. Such databases called Hidden Web or Deep Web contains high quality information. Deep web contents are generated only when queries are asked via a search interface, rendering interface integration a critical problem in many application domains, such as: semantic web, data warehouses, e-commerce etc. Many different integration solutions have been proposed so far. This paper proposes a technique to detect and construct an integrated query interface that integrates a set of web interfaces over a given domain of interest. It provides users to access information uniformly from multiple sources. The proposed strategy does that by focusing the crawl on a given topic; by judiciously looking at domain ontology which leads to pages that contain domain specific search forms in integrated manner.

Keywords: Hidden Web (Deep Web), Integrated Search Query Interface, Hidden Web Crawler, Domain Ontology, Classification.

1. INTRODUCTION

Searching on the Internet today can be compared to dragging a net across the surface of the ocean. While a great deal may be caught in the net, there is still a wealth of information that is deep, and therefore, missed. The reason is simple: Most of the Web's information is buried far down on dynamically generated sites, and standard search engines never find it. Traditional search engines create their indices by crawling surface Web pages. To be discovered, the page must be static and linked to other pages. Traditional search engines can not "see" or retrieve content in the deep Web — those pages do not exist until they are created dynamically as the result of a specific search. Because traditional search engine crawlers can not probe beneath the surface, the deep Web has therefore been hidden. The deep Web is qualitatively different from the surface Web. Deep Web [4] or Hidden Web [5], the set of such pages is estimated to be around 500 times the size of the "surface web". Besides the larger volume than the surface web, the deep web also has a higher quality and a faster growth rate [4]. Lawrence and Giles [6] estimated that 80% of all the data in the Web could only be accessed via form interfaces. Currently, there exist a number of standard tools, such as search engines and hierarchical indices that can help people in finding information. However, a large number of the web pages returned by filling in search forms are not indexable to most search engines since they are dynamically generated by querying a back-end (relational) database.

Consider, for example, a user who wants to search for some information, such as price of a used car, before he shops on the Web. Since such information only exists in the back-end databases, the user first needs to discover the URLs of the used car sites, go to the homepages, send queries through HTML forms of different sites, extract the relevant information from the result web pages, and finally compare or integrate the results from multiple sources. Therefore, there arises the need for new information services that can help users to fill search form in integrated manner. To minimize user effort in such an information retrieval process, the problem of automatically detecting the search interface of particular domain is explored.

The form-based query interfaces are designed for user interaction, providing a programmatic way to access web databases under query interfaces and integrate search system over Web databases has become an application in high demand. Such an interface would provide uniform access to the data sources of a given domain of interest. Currently, such integrated user interfaces exist, but they are manually constructed or generated using techniques that are not fully automated. Though one might pursue a manual creation of a unified interface for a few query interfaces (four or five), it is hard to do so for a large number of interfaces without errors and in a periodical manner. Aim of this paper is to construct a integrated query interface which contains all distinct fields of all schemas and arrange them in a way that facilitates users to fill in the desired information.

One of the problems faced here is that search interfaces are very sparsely distributed over the Web, even within specific domains. For example, a topic-focused best-first

[1]Department of Computer Engineering, YMCA Institute of Engineering, Faridabad, INDIA

[2]Department of Computer Engineering, YMCA Institute of Engineering, Faridabad, INDIA

Email: [1]anuangra@yahoo.com, [2]ashokkale21@rediffmail.com

crawler was able to retrieve only 94 movie search forms while crawling 100,000 pages related to movies. Therefore, it is useful to develop a strategy for visiting such web pages that are more likely than the average page to have a search interface.

The paper has been organized as follows: Section 2 describes proposes a method to detect the search interface by looking at form tags and after a form is detected on a web page, next task is to detect searchable (search interface to web database) and discard or non-searchable (login, registration, subscription, purchasing, polling, posting, etc.). Section 3 describes classification of searchable forms for particular domain. Section 4 construct an unified search interface by accumulating query interfaces to the Base interface. Section 5 draws the conclusion and describes the future research.

2. PROPOSED WORK: INTERFACE DETECTION & UNIFICATION

A search interface allows a user to search some set of items without altering them. The user enters a query, by typing or selecting options, to describe the items of interest. Results might be a page linking to items, a page containing items or a single page. The items found should match the query. The majority of search interfaces on the Web are HTML forms. It is easy to find large numbers of HTML forms, by crawling the Web and scanning crawled pages for form tags. Therefore the problem is : given an HTML form, is it a search interface?

This proposed architecture automatically detects the domain specific search interfaces by looking the domain word in the URLs, then Title and after that attributes of the source code. Feature space model described in paper classifies the web pages into a set of categories using domain ontology. This technique is based on analyzing attributes and the terms they contain by producing a set of rules relating the category of the document, its terms and their frequencies. After training our classifier, this technique is able to automatically produce a list of documents, which are the most distinguishing for each category.

2.1. Interface Detection

1. For all available URLs (URL List)
2. Check for the string login or registration or signup in URL

   If (present) Then " it is a login/ registration form"; discard URL and Go to step 1 .

   Else Goto step3.
3. Download the source code and save.
4. check the source code for tag <form> and </form> // Form analyses

   if (present) Goto step 5

   Else " It is a simple web page". Goto step 1.
5. check URL and source code
   // Search interface analysis

   if (input-type="login" or "registration" or" sign up")or (having password control)

   then Goto step 1.

   Else Goto step 6.
6. check source code if (input-type="submit" or "search" or "advance search" or "submit query" or "find" or form action="cgi")

   then Goto step 7.

   Else Goto step 1.
7. Display list of URLs i.e the list of web search interfaces.

3. DOMAIN CLASSIFIER

Domain classifier automatically merges the domain wise-search interfaces as described below.

3.1. Attribute Extraction from Search Interface

A standard HTML Web form consists of form tags—a start form tag <form> and an end form tag </form> and these tags contains form fields or attributes. Attributes may include radio buttons, check boxes, selection lists, and text boxes. The <select> tag opens a selection list, and the </select> tag closes it.

```
<select name="makeone" id="makeone">option value="0">
Select Make</option>
<select name="modelone" id="modelone"
disabled="disabled"> <option value="0">Select Model</
option></select>
```

Attribute a1="Make", Attribute a2="Model" are extracted by looking at the labels of source code of URL" http://www.autonagar.com

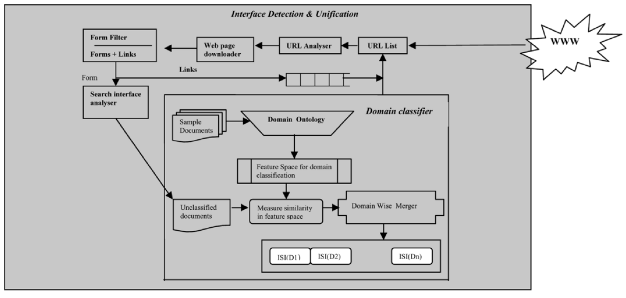

Fig.1: Architecture for Automatic Detection and Unification of Web Search Query Interfaces

3.2. Domain Ontology

Ontology is a specification of conceptualization that is description of concepts and their relationships. Ontology is composed of concepts, attributes and the relation among concepts. Concept is anything that can be described. It can be a real, fictive, concrete or abstract. Ontology in this research defines the concepts about Web page (document) categories. The main function of ontology is to provide the knowledge base needed for the classification. For defining the features, sample documents are taken and features (attributes) are extracted. Features in domain D defines: word D lies in URL or D lies in <title>……..</title> tag. Or D has 80% of following attributes a1, a2, a3,…….an.

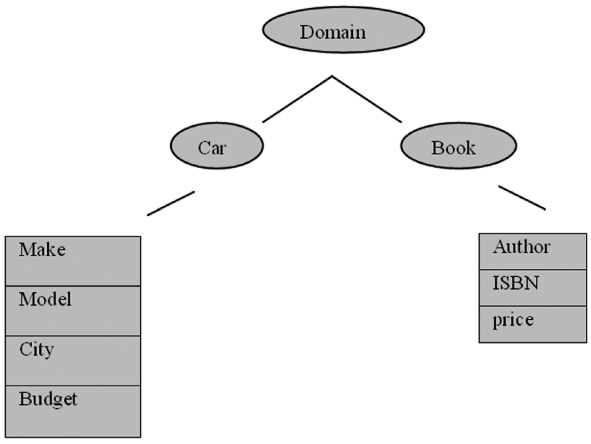

Fig. 2: Domain Wise Features

3.3. Feature Space Model

Feature Space Model is general model for representing text documents. Text documents in VSM are defined as n-dimensional feature vectors. Let T be a text document, then {(d, w), (a1, w1),(a2,w2) …… (an,wn)} is a feature vector of T where d is the domain from available domain list and w is the weight of d . a1,a2,….. are the attributes of search interface of particular domain and w1, w2,…. are respective weight of attributes. This representation requires a method to weight each term and to measure the similarity between two document feature vectors [14, 17]. Ontology concepts in VSM are also represented by n-dimensional feature vectors. Every leaf concept has list of term that manually defined by human-expert when defining the ontology or automatically resulted from predefined document collection. Weight of each term is determined by average frequency in documents with the same category.

3.4. Similarity Measure

A document similarity measure assesses the degree to which a document matches a reference feature vector. This metric is usually used to evaluate documents in order to filter those documents that are not relevant to the information need. In the vector space model, the cosine formula is the most widely used similarity measure. This similarity measure calculates the difference in direction between two feature vectors, which is basically the angle between these feature vectors, irrespective of their length. Given a web page Ti and set of concepts descriptions (models of information class) that are leaf nodes in the ontology, the classification of web page Ti according to cosine formula[13] is given by

$$V = \underset{v_j \Sigma V}{\operatorname{argmax}} \, Sim(Ti, v_j)$$

4. DOMAIN-WISE MERGER

The integration of search interface schemas is typically accomplished by Schema Matching that identifies semantic correspondences among interface attributes and Schema Merging constructs a unified schema given the discovered mappings of the attributes. Such a integrated schema should encompass all unique attributes over the given set of interfaces, and should be structurally and semantically well formed. Given a set of Web interfaces in the same domain, the problem is to construct automatically a unified query interface which contains all (or, alternatively, the most significant) distinct fields of all schemas and arrange them in a way that facilitates users to fill in the desired information.

The process of integrating query interfaces can be carried out in the following steps:

1. Extract query interfaces from web pages [2].
2. Represent the all query interfaces in form of trees such that the attributes and their parent-child relationships are obtained.
3. Compute semantic relationships between the fields in different interfaces in the same domain. In recent years, a considerable effort has been dedicated to finding these semantic mappings [2, 7, 8, 9, 10, 11, 12]. The accuracies of these automatic methods to identify the semantic mappings between fields can be as high as 90%.
4. Integrating the interfaces in the same domain into a unified interface [16].

4.1. Query Interface Representation

A query interface specifies a domain in terms of attributes and the relation between different attributes. It is typically organized in form of a tree where each node presents an attribute and each attribute is a specialization of its parent as shown in fig 3.

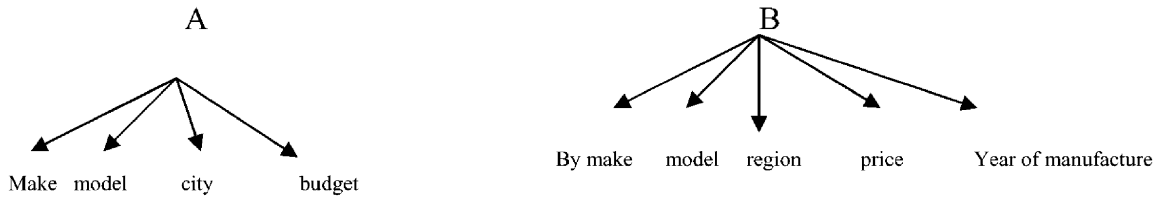

Fig. 3: Hierarchical Representation of Two Query Interfaces A and B in Car Domain

4.2. Semantic Matching

Merging of the search interfaces is based upon the semantic mappings stored in Domain-specific Mapping Knowledge base in order to generate the integrated Search Interface (ISI). [1] used three methods for semantic mapping i.e fuzzy matching, domain specific thesauras and data type matching . In fuzzy matching strategy, the matcher compares two strings and returns a similarity value in the range [0, 1]. While using, Domain-specific thesaurus, the matcher returns either 0 or 1 as the similarity value. The matcher returns 0 if there exists no relationship between attributes A1 and A2 of two different interfaces. If any relationship (synonymy, hypernymy or meronymy) exists between two attributes of different interfaces, the matcher returns 1. Similarly, for Data type matching strategy, the matcher also return 1 as similarity value only if data type of two attributes of different interfaces are same else returns 0.

4.3. Schema Merging

In the merging process, two search interfaces in same domain are represented in hierarchical form. They are merged at a time and the first interface will be treated as base interface. The base interface will contain the result of all the matching processes. If each attribute of second query interface is matched with the attributes of base query interface then assign the name of attribute in unified interface as same as in base interface. If the number of attributes in any query interface is more than the number of attributes in base query interface then insert that attribute at that particular level (i) to the base interface. At the end the base query interface S1 will be the resultant integrated query interface.

- Several functions are defined on the elements of tree as well as on the set of merge operations to be applied by the algorithm for the merging of schema. In the proposed algorithm f is the attribute of base interface S1 and e is the attribute of new interface(S2, S3,…..). The proposed algorithm algorithm uses the following functions defined on elements of trees:

```
Algorithm merge(S1 , Sj)
Begin
  if S1 and Sj are empty trees then
Sj do
      return empty tree;
  if Sj is empty tree then
do
      return S1 ;
 level1 = depth(S1);
 levelj = depth(Sj);
 for i = 0 to level1 do
 begin
      for j = 0 to levelj do
      begin
            mergebylevel(i, j);
            return S1;
      end;
 end;
end;
```

```
Algorithm mergebylevel(i, j)
Begin
                  for each node e at level j in
                  begin
                        for each node f at level i in S1
begin
      save parent P.
            if relation(e, f) then
      begin
            continue;
                  end;
            end;

                  insert(P,e);

                  end;
```

Fig 4: Algorithm for Interface Integration

- relation(e, f): For attributes e and f, this function returns 1 or 0. if there exists semantic relationship between e and f, the function returns 1, otherwise 0.
- insert(P, e): It inserts new attribute to parent at that level in the base interface S1.

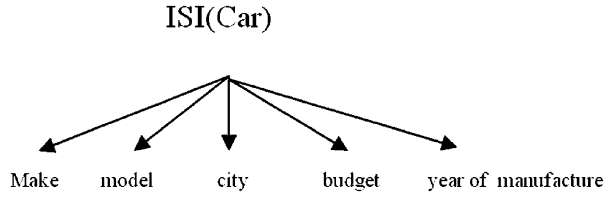

Fig. 5: Hierarchical Representation of Integrated Search Interface of Car Domain

This algorithm performs relation and insert operation to merge the interfaces and result into integrated search interface as shown in fig 5.

5. CONCLUSION AND FUTURE RESEARCH

This paper automatically detects the domain specific search interfaces by looking the domain word in the URLs, then Title and after that attributes of the source code. Feature space model described in paper classifies the web pages into a set of categories using domain ontology is presented. The problem of integrating large-scale collections of query interfaces of the same domain has been designed and developed by transforming a set of interfaces in the same domain of interest into a global interface such that all constraints are satisfied as much as possible. This interface

will permit users to access information uniformly from multiple sources of a given domain.

Though there are many recent research efforts on web query forms but there is still some important problems to be resolved: One such problem is how to achieve an important measure of crawling effectiveness which is the submission efficiency, i.e. automatic filling of unified search interface should take the frequent user pattern and query pruning techniques into consideration. Second is how to efficiently extract structured information from hidden web databases using these integrated search interfaces.

REFERENCES

[1] Anuradha, A. K. Sharma, Komal Kumar Bhatia: "Optimized Merging of Query Interfaces for Domain-specific Hidden Web" Proc. Third International Conference on Advanced Computing and Communication Technologies (ICACCT 2008).

[2] A. K. Sharma, Komal Kumar Bhatia "A Framework for Domain-Specific Interface Mapper (DSIM)", Communicated to International Journal of Information Technology and Web Engineering (IJITWE) Jan 2007.

[3] A. K. Sharma, Komal Kumar Bhatia: "Automated Discovery of Task Oriented Search Interfaces through Augmented Hypertext Documents" Proc. First International Conference on Web Engineering & Application (ICWA2006).

[4] BrightPlanet Corp. "The Deep Web: Surfacing Hidden Value."

[5] D. Florescu, A.Y. Levy, and A.O. Mendelzon. "Database Techniques for the World-wide Web: a Survey," SIGMOD Record 27(3), 59-74, 1998.

[6] S. Lawrence and C. L. Giles. Searching theWorldWideWeb. Science, 280(5360):98-100, 1998.

[7] Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to Map between Ontologies on the Semantic Web. In WWW, 2002.

[8] E. Dragut and R. Lawrence. Composing Mappings between Schemas using a Reference Ontology. In ODBASE, Pages 228– 246, 2004.

[9] Bergman, M.K., The Deep Web: Surfacing Hidden Value. 2000, BrightPlanet.com, Sullivan, D., Search Engine Sizes. The Search Engine Report, 2001.

[10] Do H.-H., Melnik S., and Rahm E.: Comparison of Schema Matching Evaluations, Proc. GI Workshop "Web and Databases", Erfurt, Oct. 2002.

[11] H. He, W. Meng, C. Yu, and Z. Wu. WISE-integrator: An Automatic Integrator of Web Search Interfaces for e-commerce. In VLDB, 2003.

[12] J. Madhavan, P. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In VLDB, 2001.

[13] I.Witten, A.Moffat, , & T.C. Bell, (1994) Managing Gigabytes: Compressing and Indexing Documents and Images. New York: Van Nostrand Reinhold.

[14] Masayu Leylia Khodra1*, Dwi Hendratmo Widyantoro, "An Efficient and Effective Algorithm for Hierarchical classificationof Search Results ", Proceedings of the International Conference on Electrical Engineering and Informatics Institute Teknologi Bandung, Indonesia June 17-19, 2007

[15] S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. In Proceedings of VLDB, Pages 129–138, 2001.

[16] B. He and K. C.-C. Chang. "Statistical Schema Matching Across Web Query Interfaces". In Proceedings of SIGMOD, Pages 217–228, 2003.

[17] Goran Nenadic, Simon Rice, Irena Spasic, Sophia Ananiadou and Benjamin Stapley. 2003b. Using Domain-Specific Verbs for Term Classification . Proceedings of ACL Workshop on Natural Language Processing in Biomedicine, Sapporo, Japan.

[18] Wright, Alex (2009-02-22). "Exploring a 'Deep Web' That Google Can't Grasp". New York Times.

[19] Michael, Lesk. How much Information is there in the World?.