Automatic Extraction of Web Search Interfaces for Interface Schema Integration

Hai He, Weiyi Meng
Dept. of Computer Science
SUNY at Binghamton
Binghamton, NY 13902
{haihe, meng}@cs.binghamton.edu

Clement Yu
Dept. of Computer Science
Univ. of Illinois at Chicago
Chicago, IL 60607
yu@cs.uic.edu

Zonghuan Wu
Center for Adv. Compu. Studies
Univ. of Louisiana at Lafayette
Lafayette, LA 70504
zwu@cacs.louisiana.edu

ABSTRACT
This paper provides an overview of a technique for extracting information from the Web search interfaces of e-commerce search engines that is useful for supporting automatic search interface integration. In particular, we discuss how to group elements and labels on a search interface into attributes and how to derive certain meta-information for each attribute.

Categories & Subject Descriptors
H.3.5 [**Information Storage and Retrieval**]: Online Information Services – Commercial Services, Web-based Services. H.5.2 [**Information Interface and Representation**]: User Interfaces – User Interface Management Systems.

General Terms
Algorithms, Management, Performance, Design.

Keywords
Search interface representation, Search interface extraction, Search engine, Metasearch engine.

1. INTRODUCTION
This paper provides an overview of a technique for extracting information from the Web search interfaces of e-commerce search engines (ESEs) that is useful for constructing e-commerce metasearch engines (EMSEs). More specifically, the aim is to extract information that supports automatic search interface integration. In [1], we presented a model to capture the information on a search interface and a tool (WISE-Integrator) that can *automatically* build a unified search interface over multiple heterogeneous ESE search interfaces of the same product domain based on the model. However, in [1], the information in the model was manually obtained from each search interface. In this paper, we outline our technique for *automatically* constructing the model.

A Web search interface for e-commerce typically contains some HTML form control **elements** such as *textbox* (i.e., a single-line text input), *radio button*, *checkbox* and *selection list* (i.e., a pull-down menu) that allow a user to enter search information. Each *element* usually has a **label** – a descriptive text – associated with it. An element may have one or more **values**. For example, a selection list usually has a list of values (options) for users to select, and a radio button/checkbox usually has a single value. Logically, elements and their associated labels together form different **attributes** of the products in the underlying database of the search engine. Often an attribute may consist of one or more labels and elements. For example, the *author* attribute in Figure 1 has four elements including a textbox and three radio buttons. The label of an attribute is referred to as **attribute name**, and element(s) of the attribute are treated as **attribute domain**. If an attribute contains multiple elements, these elements may be related in some way. For example, among the four elements of *author* in Figure 1, the textbox is treated as **domain element** while the three radio buttons are treated as **constraint elements** since each of them specifies a constraint on the domain element. In addition to such explicit composition information of attributes, each attribute is also *implicitly* associated with a set of meta-information such as the domain type (e.g., finite, range) and value type (e.g., date, currency), which is essential for enhancing attribute matching [1].

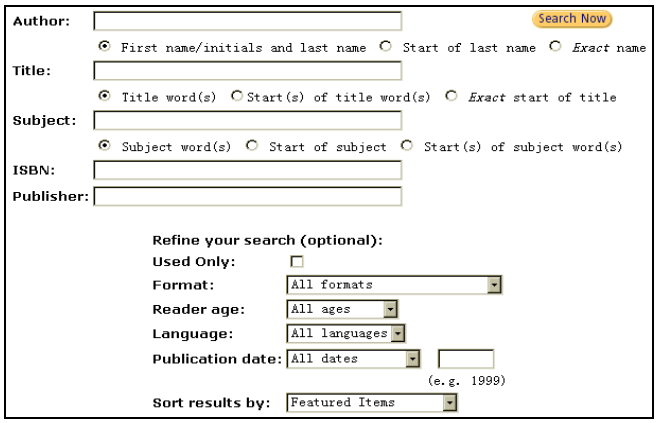

Figure 1. The book search interface of amazon.com.

2. INTERFACE EXTRACTION
In our work, interface extraction consists of two major steps: (1) *Attribute extraction*: given all search interfaces of a domain, extract labels and elements appearing in each form and then group them into logic attributes; (2) *Attribute analysis*: analyze the labels and elements of each attribute to derive meta-information of the attribute such as domain type and value type.

2.1 Attribute Extraction
We observe that labels and elements of the same attribute have a certain layout and are usually close to each other, and that in most cases they share some similar information. The layout of labels and elements can be captured as an ***i**nterface **exp**ression*(**IEXP**). For a given search interface, its IEXP is a *string* consisting of three basic items 't', 'e' and '|', where 't' denotes a label/text, 'e' denotes an element, and '|' denotes a row delimiter which represents a physical row border in the search interface. IEXP provides a high-level description of the layout of different labels and elements on the interface while ignoring the details like the values of the elements and the actual implementations of laying out labels and elements. For example, the IEXP of the search

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

interface in Figure 1 is "te|eee|te|eee|te|eee|te|te|t|te|te|te|te|tee|t|te", where the first 't' denotes the label "author", the first 'e' denotes the textbox following the label "author", the first '|' is the first row delimiter, the following three 'e's denote the three radio buttons below the textbox (the text on a radio button or checkbox is treated as the value of the element, thus the text and its radio button/checkbox together are a whole entity).

We employ a two-step approach to perform automatic extraction of attributes. In the first step, the IEXP of a given interface is constructed. Starting from HTML tag "<form>" of the search engine form, when a label or an element or a row delimiter is encountered, we append a 't' or 'e' or '|' to the IEXP (initially it is empty) accordingly. The delimiter is identified by "<p>", "
" and "<tr>" tags in the HTML source code. This process continues until tag "</form>" is reached. The IEXP organizes labels and elements into multiple rows. In the second step, based on the IEXP, labels and elements are grouped such that each group corresponds to a separate attribute. For each element *e* in a row, we need to find the text either in the same row or some rows above the current row that is most likely to be the attribute label for *e*. To this end, some special features of search forms are applied such as: (a) texts ending with a colon are likely to be attribute labels; (b) an element is likely to appear close to its attribute label.

2.2 Attribute Analysis

When attributes are extracted, attribute analysis is to analyze each attribute to derive its meta-information. In our interface representation, four types of meta-information for each attribute are defined and they are domain type, value type, default value and unit. Deriving meta-information is fairly straightforward. In the following, we sketch how to automatically extract each type of meta-information.

Domain type: Four attribute domain types are defined: *range, finite*, *infinite*, and *Boolean*. The domain type of an attribute can be derived from its label(s) and associated element(s). If an attribute has element(s) with range semantics such as "between-and" and "less than" patterns, its domain type is *range*. If it has a list of pre-defined values for users to select and they have no range semantics, its domain is of *finite* type. If it has just a single checkbox, the attribute is considered to have a *Boolean* domain type. An attribute with *infinite* domain type usually consists of *textbox*(es) with no range semantics.

Value type: Value types defined in our model include *date, time, datetime, currency, id, number* and *char*. To identify *date, time, datetime, currency* and *id*, we provide a thesaurus for each type, which just contains domain independent information such as keywords and patterns. If the labels and element values contain relevant keywords and patterns, the attribute's value type is determined. A pattern can be defined by a regular expression. For example, keywords "date" or regular pattern "[0-1]?[0-9]/[0-3]?[0-9]/([0-9]{2}|[0-9]{4})" imply a *date* value type. If an attribute does not belong to one of these five value types, then the value type is declared to be *number* if the values of each element are numeric; otherwise the value type is *char*.

Default value: Not all attributes have its default value. If an attribute just contains textboxes, then the attribute has no default value. The default value may occur in a selection list, a group of radio buttons/checkboxes. It is always marked as "checked" or "selected" in the HTML source code of forms.

Unit: The unit defines the meaning of an attribute value (e.g., *kilogram* is a unit for *weight*). To identify the unit of an attribute, we construct a unit library that contains the most popular units in e-commerce sites, such as "currency", "weight", "age" and "date". From the labels and values of an attribute, we may get some information about its unit. Then we use the information and the library to derive the appropriate unit for the attribute. For example, if a label containing "US$" implies that the unit is "USD"; "age" implies that the unit is "year".

If an attribute has multiple *domain elements*, we also identify their relationship and their semantics. In our model, three types of relationships for multiple domain elements are defined and they are *group*, *range* and *part*. If an attribute contains just multiple checkboxes/radio buttons, the group type is recognized for these elements. Since *range* is a special type of *part*, we consider range type first. A range domain type implies that the elements are of range related. Then the elements that are not of range type would be treated as part type by default.

2.3 Experimental Results

We evaluated our interface extraction technique using 184 search interfaces from 7 application domains. Grouping extracted labels and elements into separate attributes is the most complex problem in the interface extraction. In this task, we need to group the labels and elements that conceptually represent the same concept into a single attribute. Meanwhile, an attribute label should be identified for each attribute. To evaluate our method, we manually identified the attributes of each search interface, and then manually compare them with the results of our method. We evaluate the accuracy in two levels of granularity: **element level** and **attribute level**.

Element level: A label is correctly extracted for an element if it matches the manually identified label. This criterion is also used in [3] to evaluate the LITE method for label extraction. The overall accuracy of our method based on **1582** elements from **184** forms is **97.66**%.

Attribute level: An attribute consists of up to three aspects of information: the name/label of the attribute, the set of domain elements and the set of constraint elements. An extracted attribute matches a manually extracted attribute if they match on all three aspects. No results on attribute-level accuracy were reported in [3]. The overall attribute-level accuracy of our method is **95.61**%.

Our approach also has good accuracy in obtaining meta-information of extracted attributes, such as domain type, value type, unit, default value and the relationships of elements. The average accuracy of our method for each of the above category of meta-information is above 97%.

3. RELATED WORK

The works reported in [2, 3] are the most relevant to our work. Our approach is different from the above works in many ways and the major difference is that our approach is *attribute-oriented* (i.e., labels and elements are grouped into attributes), more comprehensive, and more suitable for search interface integration.

4. ACKNOWLEDGEMENT

This work is supported in part by the following grants from NSF: IIS-0208574 and IIS-0208434.

5. REFERENCES

[1] H. He, W. Meng, C. Yu, and Z. Wu. WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-commerce. VLDB Conference, Berlin, 2003.

[2] O. Kaljuvee, O. Buyukkokten, H. Garcia-Molina, and A. Paepcke. Efficient Web Form Entry on PDAs. 10th WWW conference, 2000.

[3] S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. VLDB Conference, Italy, 2001.