Querying Capability Modeling and Construction of Deep Web Sources

Liangcai Shu[1], Weiyi Meng[1], Hai He[1], and Clement Yu[2]

[1] Department of Computer Science, SUNY at Binghamton,
Binghamton, NY 13902, U.S.A
{lshu, meng, haihe}@cs.binghamton.edu
[2] Department of Computer Science, University of Illinois at Chicago,
Chicago, IL 60607, U.S.A
yu@cs.uic.edu

Abstract. Information in a deep Web source can be accessed through queries submitted on its query interface. Many Web applications need to interact with the query interfaces of deep Web sources such as deep Web crawling and comparison-shopping. Analyzing the querying capability of a query interface is critical in supporting such interactions automatically and effectively. In this paper, we propose a querying capability model based on the concept of atomic query which is a valid query with a minimal attribute set. We also provide an approach to construct the querying capability model automatically by identifying atomic queries for any given query interface. Our experimental results show that the accuracy of our algorithm is good.

Keywords: Deep Web, query interface, querying capability modeling.

1 Introduction

It is known that public information in the deep Web is 400 to 550 times larger than the so-called surface Web [1]. A large portion of the deep Web consists of structured data stored in database systems accessible via complex query interfaces [3] (also called search interfaces). Many Web applications such as comparison-shopping and deep Web crawling [2] require interaction with these form-based query interfaces by program. However, automatic interaction with complex query interfaces is challenging because of the diversity and heterogeneity of deep Web sources and their query interfaces. Automatic interaction includes automatically identifying search forms, submitting queries, and receiving and processing result pages. Some related work that has been carried out by different researchers includes: source extraction and description [4, 8] and deep Web crawling [2, 7, 11, 12].

The focus of this paper is on the automatic analysis of the *querying capability* of complex query interfaces, i.e., we want to find out what kinds of queries are *valid* (acceptable) by any given interface. A typical complex query interface consists of a series of attributes, each of which has one or more control elements like textbox, selection list, radio button and checkbox. A query is a combination of pairs, each consisting of an attribute and its assigned value. Those combinations that are accepted by the query interface syntax are *valid queries*; others are *invalid queries*.

B. Benatallah et al. (Eds.): WISE 2007, LNCS 4831, pp. 13–25, 2007.

Determining whether a query is valid is an important aspect of querying capability analysis. Consider the query interface in Fig. 1. Query {<*Author*, "John Smith">} is valid but query {<*Published date*, (2000, 2006)>} is invalid because this interface requires that at least one of the attributes with "*" must have values. The query with both conditions {<*Author*, "John Smith">, <*Published date*, (2000, 2006)>} is valid.

In this paper, we define a *query* as a set of attributes that appear in the query conditions. Traditional query conditions that include both attributes and values are defined as *query instances*. We propose the concept of *atomic query* to help describe querying capability. An atomic query is a set of attributes that represents a minimum valid query, i.e., any proper subset of an atomic query is an invalid query. For the query interface in Fig. 1, query {*Author*} is an atomic query because any query formed by filling out a value in the *Author* textbox is accepted by the search engine (note that a valid query does not guarantee any results will be returned). We can determine if a given query is valid if we identify all atomic queries for a query interface in advance. We propose a method to identify atomic queries automatically.

The work that is most closely related to our work is [8]. In [8], source descriptions are discussed. The focuses are on what kind of data is available from a source and how to map queries from the global schema to each local schema. In contrast, we are interested in describing the full querying capability of a query interface (i.e., what queries are valid) and constructing the querying capability model automatically. Both the issues studied and the solutions provided in these two works are different.

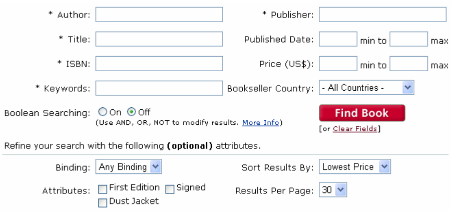
* Author:
* Publisher:
* Title:
Published Date: min to max
* ISBN:
Price (US$): min to max
* Keywords:
Bookseller Country: - All Countries -
Boolean Searching: ○ On ● Off
(Use AND, OR, NOT to modify results. More Info)
Find Book
[or Clear Fields]
Refine your search with the following (optional) attributes.
Binding: Any Binding
Sort Results By: Lowest Price
Attributes: □ First Edition □ Signed □ Dust Jacket
Results Per Page: 30

Fig. 1. Query interface of abebooks.com

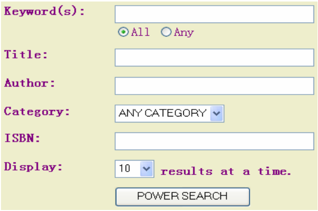

Fig. 2. Query interface of aaronbooks.com

The paper makes the following contributions:

(1). Make a classification of attributes. The attributes are classified into four types: functional attribute, range attribute, categorical attribute and value-infinite attribute.

(2). Propose the concept of atomic query (AQ) and a querying capability model.

(3). Present an algorithm to construct the AQ set that represents querying capability for a given query interface.

(4). Compare different classifiers' performance on result page classification.

The rest of this paper is organized as follows. Section 2 introduces attribute types. Section 3 presents a querying capability model of query interfaces and introduces the concept of atomic query. Section 4 discusses constructing querying capability of a query interface. Section 5 reports experimental results. Section 6 concludes the paper.