

# Knocking the Door to the Deep Web: Integrating Web Query Interfaces\*

Bin He, Zhen Zhang, Kevin Chen-Chuan Chang  
Computer Science Department  
University of Illinois at Urbana-Champaign  
{binhe, zhang2}@uiuc.edu, kcchang@cs.uiuc.edu

## 1. INTRODUCTION

Recently, we witness the rapid growth and thus the prevalence of databases on the Web. Our recent survey [2] in December 2002 estimated between 127,000 to 330,000 deep Web sources. On this deep Web, myriad online databases provide dynamic query-based data access through their *query interfaces*, instead of static URL links. As the “door” to the deep Web, it is essential to integrate these query interfaces for integrating the deep Web.

The overall goal of the MetaQuerier project (<http://metaquerier.cs.uiuc.edu>) aims at opening up the deep Web to users, by building a system to help users exploring and integrating deep Web sources. In particular, to start with, we focus on the integration of deep Web sources in the same domain (e.g., Books, Airfares), which is itself an important integration task. The typical scenarios include purchasing a book with lowest price among book sources and a flight ticket with the best trade-off between price and number of connections among airline sources. The deep Web presents challenges for such *large-scale* integration: With plenty of databases in one domain, how can we integrate them to facilitate user queries?

To automate this integration scenario, for each domain, the MetaQuerier constructs a mediator, which provides users a *unified interface* to query. For instance, in Books domain, the unified interface may contain title, author, subject and ISBN. Users send their queries through the unified interface and the mediator translates the queries to each specific online source (e.g., *amazon.com* and *bn.com*) and then returns the integrated query result to the users.

In particular, the scope of this demo is to automate the integration of Web query interfaces (i.e., construct the unified interface). Specifically, this demo addresses several important problems. First, *interface extraction* – automatically extracting the attributes, given a query interface in HTML format. Second, *schema matching* – discovering the semantic correspondences (e.g., synonyms) among attributes. Third, *interface unification* – constructing a unified interface based on the discovered matchings.

As a new attempt, our solutions are leveraging the large scale “regularity” of Web query interfaces to explore their hidden “se-

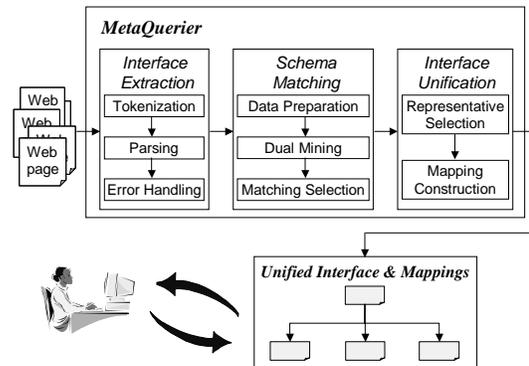


Figure 1: The framework of MetaQuerier system.

manics.” Specifically, to solve the interface extraction problem, we introduce a *parsing* paradigm by hypothesizing the existence of *hidden syntax* which describes the layout and semantics of Web interfaces [9]. Also, unlike traditional pairwise schema matching, we propose a holistic matching approach, which matches all the schemas at the same time with the hypothesis of a *hidden schema model* [4]. This demo integrates both of these new techniques into a coherent system.

To the best of our knowledge, MetaQuerier is the first fully integrated streamlined system to automatically unify Web interfaces. In the literature, large-scale integration or *meta-search* of structured information sources has largely been left unexplored. Existing research effort [1, 8, 7] has mostly focused on meta-search over *text* sources, with simple keyword search, where the problems of interface extraction, schema matching and interface unification do not exist. Also, some recent work such as [6] also aims at integrating Web interfaces. However, they all focus on part of the integration work (e.g., schema matching) and are not really fully integrated. In particular, none of them solves the problem of interface extraction. In contrast, the MetaQuerier system is the first fully integrated system that starts from raw HTML pages.

## 2. OVERVIEW OF METAQUERIER

The MetaQuerier system consists of three subsystems in a sequence: interface extraction, schema matching and interface unification, as Figure 1 illustrates. The input of the MetaQuerier system is a set of Web pages (containing query interfaces) in the same domain. In practice, crawling (e.g., our random IP based crawling in [2]) and clustering (e.g., our clustering algorithm for Web interfaces [5]) techniques can be used to get such input. In this demo, for the purpose of simplifying the illustration, we show the MetaQuerier system on several domains selected from the TEL-8 dataset of the UIUC Web Integration Repository [3].

Specifically, the interface extraction step accepts Web pages as input and outputs extracted attribute information. Next, the schema

\*This material is based upon work partially supported by NSF Grants IIS-0133199 and IIS-0313260. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMOD 2004 June 13-18, 2004, Paris, France.

Copyright 2004 ACM 1-58113-859-8/04/06...\$5.00.

matching step cleans and then matches these extracted attributes. Last, the interface unification step constructs a unified interface and corresponding mappings based on the discovered matchings. In this section, we briefly present the techniques developed in these three steps.

**Interface Extraction:** Interface extraction extracts the attribute information (e.g., names and domain values) from the Web query interfaces in HTML format. We introduce a *parsing* paradigm by hypothesizing that there exists *hidden syntax* to describe the layout and semantic of Web interfaces [9]. To our knowledge, this parsing approach is the first systematic effort to automatically understand complex Web interfaces.

We observe that, in Web interfaces, the *conditions patterns* are quite regular in layout. For instance, in Web interfaces, the most frequently used condition pattern is a text (as the attribute name) followed by an input box (as the attribute value). Our survey [2] in Books, Automobiles and Airfares domains shows that they share a small set of condition patterns.

We thus hypothesize the existence of the *hidden syntax* of these interfaces. The hidden syntax describes the composition from condition patterns to Web interfaces based on the layout information. With this hypothesis, we pursue a *parsing* approach for understanding Web interfaces, considering the hidden syntax as the grammar to parse. As Figure 1 shows, the interface extraction consists of: 1) *tokenization* – accepts Web pages as input and outputs a set of *tokens* as instances of terminal symbols (e.g., text, inputbox) in the grammar, 2) *parsing* – accepts tokens generated by the tokenizer and derives best-effort parse trees based on the grammar, and 3) *error handling* – handling conflicts and errors in the parsing result such as the situation of multiple parse trees. Note that a parse tree corresponds to a specific interpretation of the query interface and it is possible to get multiple parse trees because of the inherent ambiguities of the derived grammar.

**Schema Matching:** Schema matching discovers the semantic correspondences (i.e., matchings) among the attributes in Web interfaces. We introduce a *holistic schema matching* approach by exploiting the *context* information across all the interfaces [4].

The integration of the deep Web needs to discover *complex matchings*. In complex matching, a set of  $m$  attributes is matched to another set of  $n$  attributes and it is thus also called  *$m:n$  matching*. We observe that, in deep Web sources, complex matchings do exist and are actually quite frequent. For instance, in Books domain, `author` is the synonym of the grouping of `last name` and `first name`, i.e.,  $\{\text{author}\} = \{\text{first name}, \text{last name}\}$ ; in Airfares domain,  $\{\text{passengers}\} = \{\text{adults}, \text{seniors}, \text{children}, \text{infants}\}$ .

Unlike traditional pairwise schema matching, we explore a new dimension, the *context*, to match all the schemas at the same time [4]. Traditionally, schema matching relies on matchings between pairwise schemas before integrating multiple ones. In contrast, we propose to exploit statistical analysis to holistically match many schemas by discovering their co-occurrence patterns.

Specifically, we abstract the problem of complex matching as *correlation mining*, motivated by our observations on the schema data. We observe that *grouping attributes* (i.e., attributes in one group of a matching e.g.,  $\{\text{last name}, \text{first name}\}$ ) tend to be co-present and thus positively correlated across sources in the same domain. In contrast, *synonym attributes* (i.e., attribute groups in a matching) are negatively correlated because they rarely co-occur or even mutually exclusive in schemas. These observations motivate us a correlation mining abstraction (and essentially a data mining spirit) of the schema matching problem.

Motivated by our observation, we develop an automatic process, given the extracted attributes from interface extraction, to discover

complex matchings, consisting of: 1) *data preparation* – as preprocessing, data preparation accepts the extracted attributes as input and outputs cleaned data by merging syntactically similar attributes (e.g., “title” and “title of books” denote the same attributes and thus are merged), 2) *dual mining* – discovering complex matchings with a dual mining of positive correlation (i.e., potential groups) and negative correlation (i.e., potential matchings), 3) *matching selection* – choosing the most convincing and consistent matchings from the mining result.

**Interface Unification:** Interface unification constructs a unified interface and the mappings between the unified interface and local interfaces (i.e., the specific interfaces of sources).

Given the complex matchings (that reveal synonym relationship) among attributes, we construct the unified interface by: 1) *representative selection* – choosing a representative among the synonyms (e.g., choose `author` from the matching  $\{\text{author}\} = \{\text{last name}, \text{first name}\}$ ), and 2) *mapping construction* – building up the mappings from the unified interface to local interfaces. For instance, the `author` in the unified interface is mapped to any local interface with attributes `last name` and `first name`.

### 3. SYSTEM DEMONSTRATION

The implementation of the MetaQuerier system adopts Python 2.3, wxPython 2.4 and Javascript on Windows XP. Javascript is used to implement the tokenization of interface extraction by exploiting the DOM API. Python is used to implement other parts. In particular, we use wxPython 2.4 to implement the graph user interface of the system.

The MetaQuerier system is a fully automatic system, meaning that it processes all the tasks in streamline and outputs the unified interface and mappings without the interaction of human experts. For demonstration purpose, the MetaQuerier system also supports “single-step” operation to explicitly illustrate the function of each task. Also, we demonstrate the MetaQuerier system on several domains selected from the TEL-8 dataset of the UIUC Web Integration Repository [3]. The complete TEL-8 dataset contains the original query interfaces of 447 deep Web sources from 8 representative domains.

### 4. REFERENCES

- [1] J. P. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *SIGMOD Conference*, 1999.
- [2] K. C.-C. Chang, B. He, C. Li, and Z. Zhang. Structured databases on the web: Observations and implications. Technical Report UIUCDCS-R-2003-2321, Department of Computer Science, UIUC, Feb. 2003.
- [3] K. C.-C. Chang, B. He, C. Li, and Z. Zhang. The UIUC web integration repository. Computer Science Department, University of Illinois at Urbana-Champaign. <http://metaquerier.cs.uiuc.edu/repository>, 2003.
- [4] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *SIGMOD Conference*, 2003.
- [5] B. He, T. Tao, and K. C.-C. Chang. Clustering structured web sources: A schema-based, model-differentiation approach. In *EDBT'04 ClustWeb Workshop*, 2004.
- [6] H. He, W. Meng, C. Yu, and Z. Wu. Wise-integrator: An automatic integrator of web search interfaces for e-commerce. In *VLDB Conference*, 2003.
- [7] W. Meng, K.-L. Liu, C. T. Yu, X. Wang, Y. Chang, and N. Rishe. Determining text databases to search in the internet. In *VLDB Conference*, 1998.
- [8] M. S. Panagiotis G. Ipeirotis, Luis Gravano. Probe, count, and classify: Categorizing hidden web databases. In *SIGMOD Conference*, 2001.
- [9] Z. Zheng, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best effort parsing with hidden syntax. In *SIGMOD Conference*, 2004.