Deep Web Integration with VisQI

Thomas Kabisch
Humboldt-Universität zu Berlin
Berlin, Germany
kabisch@informatik.hu-berlin.de

Eduard C. Dragut, Clement Yu
University of Illinois at Chicago
Chicago, USA
{edragut, yu}@cs.uic.edu

Ulf Leser
Humboldt-Universität zu Berlin
Berlin, Germany
leser@informatik.hu-berlin.de

ABSTRACT

In this paper, we present VisQI (VISual Query interface Integration system), a Deep Web integration system. VisQI is capable of (1) transforming Web query interfaces into hierarchically structured representations, (2) of classifying them into application domains and (3) of matching the elements of different interfaces. Thus VisQI contains solutions for the major challenges in building Deep Web integration systems.

The system comes along with a full-fledged evaluation system that automatically compares generated data structures against a gold standard. VisQI has a framework-like architecture such that other developers can reuse its components easily.

1. INTRODUCTION

There are millions of Deep Web sources. Building systems which would be able to automatically use all or a large fraction of all Deep Web sources of a given domain, such as airline reservation in the USA, would offer great benefit, but also poses serious challenges to its developers.

The most advocated approach to integrate Deep Web sources is to perform integration *domain-wise.* First, query interfaces are extracted from relevant Web pages [4]. Second, they are clustered on application domains [1]. Third, fields of different interfaces in the same domain are matched [6]. Query interfaces are integrated for further applications such as merging to form a unified interface [2] or constructing federated information systems. These systems allow a user to query a number of underlying data sources at once. Returned data of the individual sources is extracted and the results ranked in descending order of desirability (e.g., price).

In this demonstration, we describe our system VisQI (VISual Query interface Integration system) that implements the steps enumerated above. First, the system extracts HTML query interfaces into hierarchical representations [4]. Second, the system classifies previously unseen query interfaces to their application domains. Finally, it matches and clusters elements of different interfaces according to their intended meaning [6]. For example, the field `Departure city` of interface `finnair.com` matches the field `Leaving From` in Fig. 1 and thus they are placed into the same cluster. VisQI has a sophisticated testing component which is accompanied by large gold standard sets for both query interface extraction and matching. An intuitive visualization component eases the work of the integrator. VisQI is a cornerstone in our effort to provide a comprehensive solution to the problem of integrating Deep Web sources (see [2, 3]).

Proceedings of the VLDB Endowment, Vol. 3, No. 2
Copyright 2010 VLDB Endowment 2150-8097/10/09... $ 10.00.

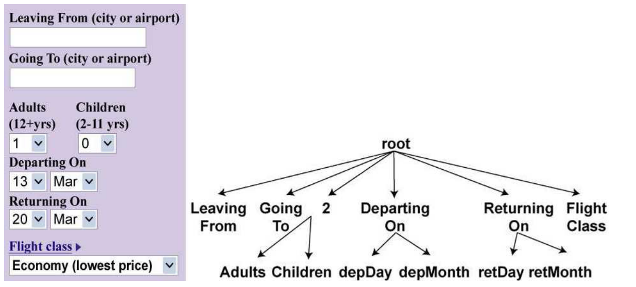


Figure 1: A Web query interface in the airline domain along with its extracted interface tree.

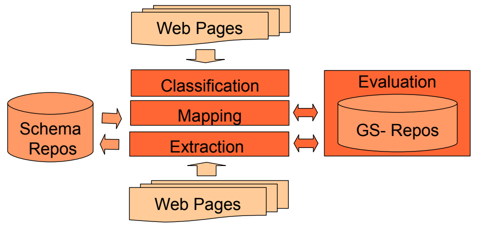
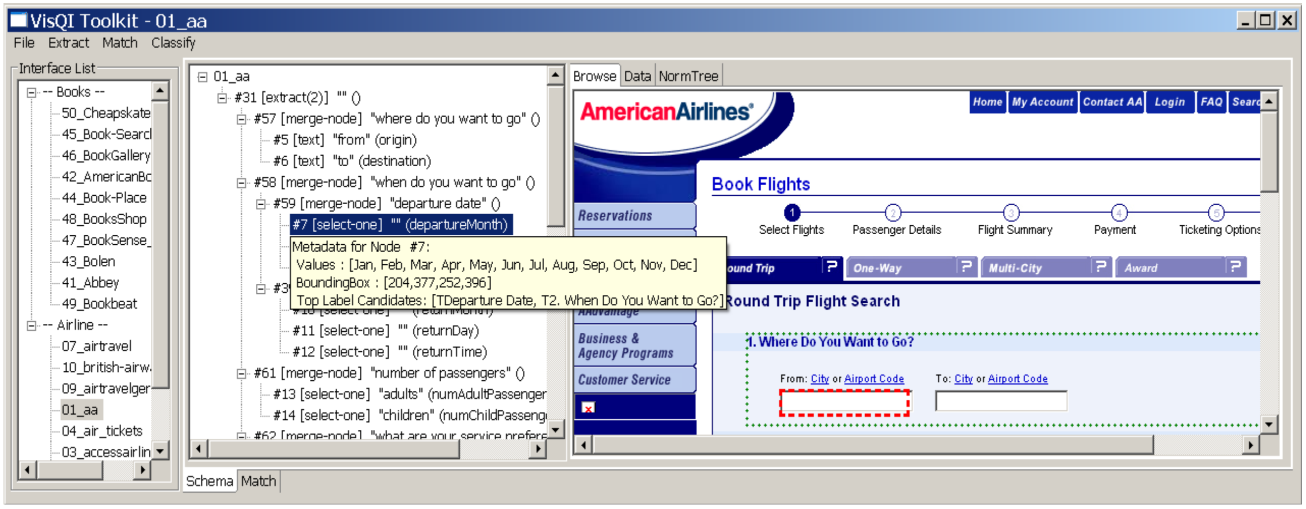


Figure 2: Architecture of VisQI

VisQI does *not* address end users, but developers of Deep Web integration systems. Such developers may use VisQI as a standalone application to support their own Deep Web integration systems. They may use the intermediate results of VisQI (e.g., extracted schema trees, domain-wise mappings) as input to their own projects. Due to the modular architecture of the system, they may also reuse the components (e.g., the extraction component) to build their own systems. Finally, developers may use VisQI to evaluate their extraction and matching algorithms.

To this end, VisQI offers several important features. It is capable of rendering arbitrary Web query interfaces together with the extracted structures. If extraction errors are detected, the system can help to debug the algorithm by using the graphical representation. Furthermore, it offers a built-in feature to automatically perform large-scale evaluations against a gold standard data set. We built the system as a framework of loosely-coupled components (rendering, extraction, gold standard management, etc.), each having a



```
-- Books --
   50_Cheapskate
   45_Book-Searcl
   46_BookGallery
   42_AmericanBc
   44_Book-Place
   48_BooksShop
   47_BookSense_
   43_Bolen
   41_Abbey
   49_Bookbeat
-- Airline --
   07_airtravel
   10_british-airw.
   09_airtravelger
   01_aa
   04_air_tickets
   03_accessairlin
```

```
01_aa
  #31 [extract(2)]  "" ()
    #57 [merge-node]  "where do you want to go" ()
      #5 [text]  "from" (origin)
      #6 [text]  "to" (destination)
    #58 [merge-node]  "when do you want to go" ()
      #59 [merge-node]  "departure date" ()
        #7 [select-one]  "" (departureMonth)
      #3
        #11 [select-one]  "" (returnDay)
        #12 [select-one]  "" (returnTime)
    #61 [merge-node]  "number of passengers" ()
      #13 [select-one]  "adults" (numAdultPassenger
      #14 [select-one]  "children" (numChildPasseng
    #62 [merge-node]  "what are your service prefere
```

```
Metadata for Node  #7:
 Values : [Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec]
 BoundingBox : [204,377,252,396]
 Top Label Candidates: [TDeparture Date, T2. When Do You Want to Go?]
```

Figure 3: A snapshot of the system showing the schema view.

clean interface. Thus, other developers may take advantage of our development by simply replacing the extraction algorithm, getting all other components such as the visualization and evaluation for free.

Though there has been a considerable number of proposals to Deep Web integration (see, among others, WISE-Integrator [5] and MetaQuerier [7]), very few of these are available as running and extensible systems. Furthermore, none of them supports all three steps of integration, none is equipped with a data set for testing as large as that of VisQI. In addition, as shown in [4], our system outperforms previous methods [5, 7] in terms of quality of the extracted data structure.

2. SYSTEM ARCHITECTURE

The system is designed as a framework consisting of loosely-coupled components. The architecture of VisQI is depicted in Figure 2. The key functionalities of the system are:

A rendering and extraction component that first converts a Web page into a list of *tokens*. Briefly, a token corresponds to a visible element of the page, e.g., a label or a field. Each token has an associated list of properties. One such property is a rectangular box describing the visual placement of the token (element) in the page, another property is the horizontal/vertical alignment of the tokens. Second, this component uses the various properties of the tokens to infer a tree structure for the elements of the query interface (called *schema tree*). An example of a schema tree is depicted in Figure 1.

A classification component. This component determines the application domain (e.g., car rental, real estate) of a query interface. There are two scenarios for classification. On the one hand, a set of interfaces of different domains is already known and a new unseen interface needs to be classified by considering this knowledge. VisQI finds the best matchings between the nodes of the new schema tree and the clusters (to be defined below) of interface nodes of the known interfaces. The algorithm classifies the new interface to the domain that best matches its schema tree nodes.

On the other hand, VisQI clusters a set of query interfaces of unknown domains by finding alignments of their schema trees. Properly aligned trees are clustered together and therefore form a domain.

A matching component that identifies semantically equivalent nodes among a number of interfaces. The matching component takes as input a set of schema trees and outputs a set of clusters. A *cluster* contains a number of schema tree nodes from different interfaces that all denote the same semantic concept (e.g., departure city).

The clustering is based on a novel multi-phase algorithm which combines several evidences for semantic equivalence, such as label similarity, value overlaps, and the neighborhood of nodes in their trees. The algorithm also uses transitivity of the equivalence relation to infer novel mappings.

We evaluated the mapping on 150 interfaces from 7 domains. In contrast to [6] our algorithm fully exploits the hierarchical representation of interfaces and also computes mappings between internal nodes; thus, both systems cannot be directly compared.

Figure 4 depicts the mapping between the nodes of two query interfaces. Query interfaces are shown in the hierarchical representation. An arrow between the nodes of two distinct schema trees represents that the two nodes may be semantically equivalent. A user can delete incorrect mappings, add missed mappings and edit a mapping. If a matching golden standard is present, the system highlights the incorrect and the missed mappings to a user. The system can also be used to manually define the matching between two query interfaces from scratch. To our knowledge, none of the existing Deep Web integration systems (e.g. [5, 7]) has such a component.

An evaluation component which allows for the assessment of the results of the other components. The evaluation utilizes manually defined gold standards. Currently, the system is equipped with schema trees of more than 500 interfaces in 15 application domains (see [4] for details on this data set). For about 200 interfaces of the data set, the gold standard provides manually defined mappings among the elements of the interfaces.

Users may run the system in one of two modes: the extraction mode, and the mapping and classification mode.

In the extraction mode users interactively select a list of Web pages (URLs or files) which are then analyzed by the extraction algorithm. Analyzing a Web page consists of identifying the query interface, if present, and extracting the schema tree from the form. The result of the extraction is displayed graphically. The extraction takes on the average 5 seconds per interface, where a significant portion of this time is spent on loading and rendering the Web page (about 4 seconds). If a gold standard schema tree for the user inter-

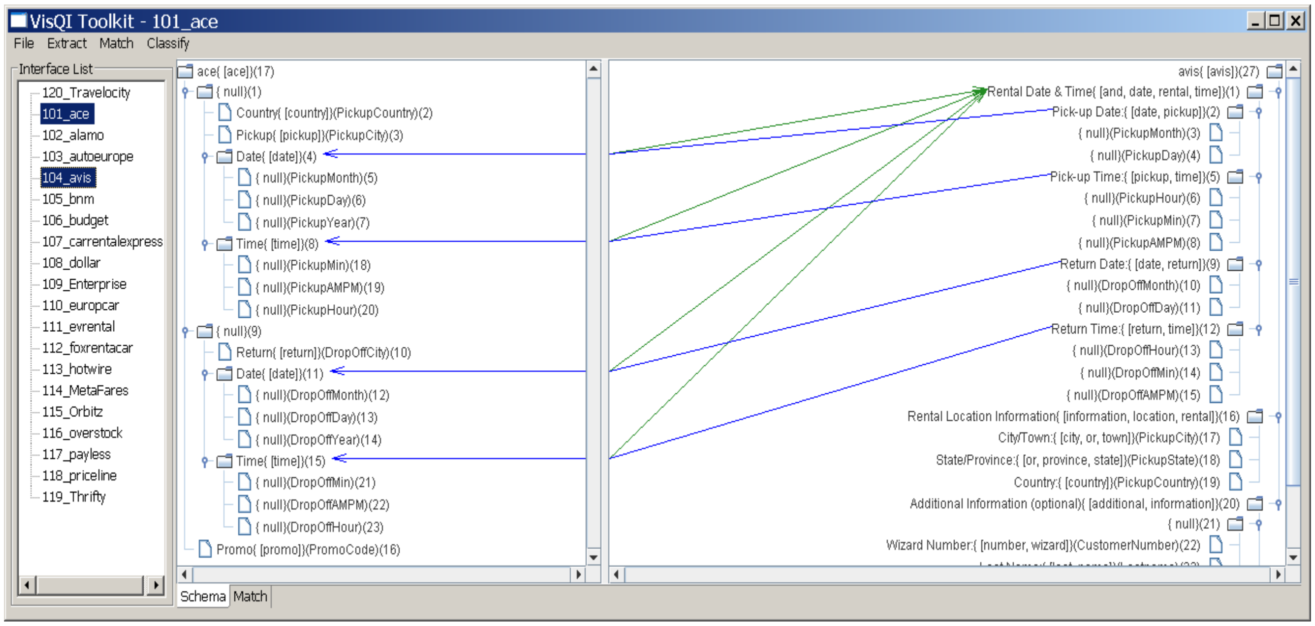


Figure 4: A snapshot of the system showing the mappings between two query interfaces. Blue arrows depict mappings computed by the algorithm whereas green arrows denote mappings added by the user.

face is available, the evaluation component can be invoked. The evaluation component calculates various statistics such as precision and recall of the extracted elements or the overall accuracy of the tree structure.

In the mapping and classification mode the system first extracts the vocabulary of the loaded set of interfaces based on labels and names of query interface elements (see also [3]). Second, the system estimates similarities for all pairs of schema tree nodes of the loaded query interfaces based on their labels, names, values and tree positions. It clusters semantically similar nodes together. In contrast to other works, in this work the presence of schema trees allows to simplify the mapping problem. For instance, 1:m mappings between fields (leaves) can be represented as 1:1 mappings between fields and internal nodes (internal nodes represent a group of leaf nodes). For example, the field `Passengers` in the query interface of `delta.com` is mapped 1:1 to the internal node `Passengers` in the query interface of `aa.com`, which is the parent of the fields `Adults, Seniors` and `Children`. This mapping is equivalent to a 1:3 mapping between the field `Passengers` in `delta.com` and the fields `Adults`, `Seniors` and `Children` in `aa.com`. The user may invoke the evaluation component to request precision, recall and F-measure of the identified mappings utilizing the mapping portion of the gold standard.

Finally, a user may utilize the system in the mapping and classification mode to check whether a new interface matches a learned domain. The system utilizes the estimated node clusters for this purpose. It estimates a best matching cluster for each node of the interface tree. If the number of possible matches meets a threshold, the system assumes that the new interface belongs to the learned application domain.

3. USAGE OF VISQI

The main window of the application has two vertical panels. The panel on the left shows the list of currently loaded interfaces. The content of the panel on the right changes depending on usage scenarios. For example, if the user selects a particular query interface from the list, the panel on the right shows the schema view. In the following, we describe the user interface of VisQI along several typical usage scenarios.

Rendering Web Pages. The visual coordinates (e.g. semantic scope, bounding box) of the objects within Web query interfaces are quite difficult to obtain; e.g., the bounded rectangle of the internal node denoted by `Where Do You Want to Go?` is shown in Figure 3. Many applications, especially Web extraction and integration tools, rely on the graphical rendering information to understand the input Web pages. Those applications may reuse our rendering component as an easy-to-use module that delivers this information for HTML pages.

Extracting Interfaces. The system may be used to extract schema trees interactively or in the background. The result, primarily an XML file containing the tree, can be directly used as input to other systems/components, such as query interface matching and merging [2]. It can also be used as an analysis/debugging extraction tool. When used in this fashion for a given web page, the middle panel shows the hierarchical representation of the currently active query interface as inferred by our algorithm and the right panel contains a tab-folder that holds different views of the current interface such as the *rendered version* of the Web page (see Figure 3) enriched with other visual clues. For example, the extracted elements are highlighted, the bounding boxes (shown in dotted lines) of the fields and labels are emphasized, etc. During extraction the user can specify either a set of URLs (batch mode) or a single URL of a page to be analyzed on-line.

Domain Classification of Interfaces. Many applications, such as the construction of vertical meta search engines require the identification of the application domain (e.g., hotel) of each Deep Web search engine. VisQI provides a classification algorithm which automatically infers the domain of a search engine using the query interface of the search engine. It has an easy to use user interface to check whether a given Web page that contains a Web query interface belongs to a given application domain. As alluded in the previous section, the intuition of the classification algorithm is that an interface that has the "best" matchings with the interfaces in an application domain A than with the interfaces in any other domain is likely to belong to A. For example, in Figure 3 the panel on the left shows the interfaces grouped by their application domains.

Matching Query Interfaces. For integration purposes, such as the generation of an integrated interface or query translation, a user might be interested in determining the equivalent fields in different query interfaces. When the system is used for matching, the match view is shown. It displays information about nodes and pairs of nodes. Figure 4 shows the match view. There is an alternative matching view which displays the mappings in a tabular format. A row in the table of mappings represents a pair of candidate matching nodes from two schema trees. One of the columns in the table of mappings shows the aggregate similarities for each pair. To ease the analysis of automatically retrieved matches VisQI classifies the matches into several classes such as all predicted matches, missed matches and gold standard matches.

Managing Deep Web Repository. The system allows to manage a repository of Deep Web sources. This contains the source files, the schema trees and interface mapping information. A user may store these pieces of information in files, edit and reload them. She may add the schema tree of a new interface to the gold standard. Furthermore, she can easily update the mapping to reflect the addition of the new interface to the repository.

Testing Extraction Algorithms. Due to its modular architecture, it is fairly easy to replace the extraction algorithm with another one. This feature can be used by every developer to make use of the other components of our software. She simply plugs-in her extraction algorithm into our system, which then displays the extracted interface graphically. By comparing the extracted interface against a gold standard (possibly produced by herself as described above), she can analyze differences as highlighted by our system. This makes testing a lot easier. Alternatively, a developer may partition the gold standard of interfaces into two subsets, one for training and the other for testing.

Performing Batch Evaluations. Although VisQI can perform one interface at a time experiments, this is not practical when the goal is to integrate hundreds of Deep Web source. The large gold standard set that accompanies the system permits large-scale experiments. A batch mode is implemented to support large experiments, which generates detailed statistics. A user can evaluate either the extraction of a set of query interfaces or the matching between a set of query interfaces.

User interfaces were developed to support this step, too. For example, for a given query interface, it displays the gold standard and the extracted schema trees side-by-side and it highlights the differences between the two.

During evaluation of mappings the system shows the correct matches using a checkbox in the first column of the table. Additionally, the evaluation component adds separate tables of mappings to the user interface that visualize the gold standard mappings and the missed mappings.

Performing Analytical Studies. We provide a way for a tester or developer to devise better evaluation scenarios. This is helpful to understand the weaknesses and strengths of a given algorithm as well as to perform comparative studies of competing algorithms. For example, we compare WISE-Extractor and our system on using several scenarios. In one of them only "flat" query interfaces were considered while in another scenario complex interfaces (trees with depth larger than three) were used. The former scenario showed that WISE-Extractor and our tool have comparable accuracies on "flat" interfaces whereas the latter scenario showed that our tool is significantly better than WISE-Extractor for complex interfaces. Testing along certain properties (such as "flatness") of the gold standard draws better conclusions about the results. Note that this is also valuable for other problems, such as schema matching. For example, one can devise a study about the quality (deterioration) of matching over interfaces whose fields lack instances.

4. DEMO PLAN

We shall illustrate all components of VisQI, which together are sufficient as algorithmic basis for the building of Deep Web integration systems. We shall demonstrate the entire range of functionalities of VisQI using URL's of real Deep Web sources. The visitor will be encouraged to participate in an interactive mode. For instance, the visitor can supply Deep Web sources of his/her choice. We shall analyze these sources with VisQI. First, the result of the extraction will be inspected together with the visitor. Then we will illustrate the classification component. We will ask the system to classify the interface chosen by the user. Then the matching is demonstrated. The new interface is matched against existing interfaces. We will guide the visitor through the process of verifying the matching candidates. To demonstrate the usage of the batch mode, the visitor may select a number of interfaces, new or existing ones.

If the gold standard for the Deep Web sources chosen by the visitor is present in our system, then the evaluation component is exhibited. We will illustrate how the results of the extraction, classification and matching algorithms are evaluated. If no gold standard is present, we will guide the visitor to the creation of her own (small) gold standard.

We also plan to provide example sets from different application domains. This will demonstrated that our solution is generic and applicable across many application domains on the Deep Web.

5. REFERENCES

[1] L. Barbosa, J. Freire, and A. S. da Silva. Organizing hidden-web databases by clustering visible web documents. In *ICDE*, 2007.

[2] E. Dragut, W. Wu, A. P. Sistla, C. T. Yu, and W. Meng. Merging source query interfaces on web databases. In *ICDE*, 2006.

[3] E. C. Dragut, F. Fang, A. P. Sistla, C. T. Yu, and W. Meng. Stop word and related problems in web interface integration. In *VLDB*, 2009.

[4] E. C. Dragut, T. Kabisch, C. Yu, and U. Leser. A hierarchical approach to model web query interfaces for web source integration. In *VLDB*, 2009.

[5] H. He, W. Meng, C. T. Yu, and Z. Wu. Constructing interface schemas for search interfaces of web databases. In *WISE*, 2005.

[6] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *SIGMOD*, 2004.

[7] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: best-effort parsing with hidden syntax. In *SIGMOD*, 2004.