Internet Business Intelligence

Hao Tan
Service Oriented Computing (SOC)
LIRIS, University Lyon 1
LYON, FRANCE
Email: ferdinandfly@gmail.com

Parisa Ghodous
SOC
LIRIS, University Lyon 1
LYON, FRANCE
Email: parisa.ghodous@recherche.univ-lyon1.fr

Jacky Montiel
ALTERNANCE Soft
LYON, FRANCE
Email: jmontiel@ntsys.fr

***Abstract*—Business Intelligence (BI) refers to computer-based techniques used in spotting, digging-out, and analyzing business data. It is mainly focused on how to dig out business data. This type of business data is a on-line web database which can be searched through their Web query interfaces. Deep Web (often called hidden web or invisible web) is composed of all the web databases. With the evolution of the "deep web", more and more researchers pay attention to the "integration" of the web database. However, to achieve this goal, it needs a complex system and many applications to work together. We are interested in an automatic extracting system to get the formulas or the lists of the results from those websites in specific domain of government procurement. To tackle this challenge, we propose a solution to create a unified interface and to inquire resources in a predefined domain. In this paper, we will discuss the automatic extracting system in several steps. First of all, the web query interfaces crawler which can execute JavaScript guarantees the coverage of the web database. Secondly, the query interface extractor and the interface integrator can allow us query all these founded web databases through a global query interface. Thirdly, the result page extractor and the result integrator can give a unified presentation. Lastly, a feedback method is developed to gather the result accuracy. A statistical model is built to improve the performance of the step 2 and 3. We assume our system is a dynamic system, which means the more we use it, the more precise results we will get.**

Keywords-schema matching; web-database integration;

I. INTRODUCTION

Deep web (often called invisible or hidden Web) is made up of massive web databases. The traditional search engines which use static URL based crawler are not able to access most of the data on the Internet. The reason is that this kind of information is hidden behind the query interfaces and does not contain a unique URL link. The recent study [20] shows that the top 3 famous search engine: Google, Yahoo and MSN only cover respectively 32%,32% and 11% of the result pages of the sample Web databases. More importantly, even if we combine the three search engines, we can only get 37% coverage. Because of this, users often have difficulties to find the sources of web database and query them to get the results. On the other hand,the interfaces are built to be queried and generate the dynamic result pages. As traditional access methods cannot accomplish the work of searching the deep web, it is imperative to find a new way to index the hidden result pages.

Deploying an automatic system to understand and extract the deep web information from the entire Internet is still impracticable based on the existing technologies. Recent research [10], [13] and [17] decomposed this into different domain-based web database integration problems. For each domain, a global interface can be built to integrate all the web databases. To enable effective access to web databases, our works focus on building a domain based, automatic web database integration system that is independent of the domain style. We choose the domain $Government\ Procurement$ to analyze and to test the performance of this system. The main purpose of such technology is to be more efficient to get the business information from the Internet as we called $Internet\ Business\ Intelligence$. This application covers all the sites found in the domain predefined and makes a unified query form to list all the analyzed data. According to this objective, we need to develop a system that has the following three features. First, it should dynamically find the data sources in a specific domain. The only input is some keywords of this domain. Second, it needs to integrate these data sources automatically, including query interfaces and the query results. This integration should not have human intervention such as predefined training samples, and query interface extractor interpreted by programmer, etc. Third, the performance should be measured and it needs a real-time feedback system to gather information to adjust the integrator. That means this system is a self-training system.

The remainder of the paper is organized as follows. In Section 2, the related works are introduced. In Section 3, the whole web-database integration system is explained and the relationships between the subsystems are clarified by diagrams. In Section 4, the Interface Extraction subsystem is well developed and in Section 5, we focused on the Interfaces Integration subsystem. A feedback method is introduced in Section 6 and we conclude in Section 7.

II. RELATED WORK

The purpose of Internet Business Intelligence is not limited to domain-based automatic web database integration. It can be easily extended to a large scaled integration system by supplying semantic ontology for additional domains. This system considers the new problems come with the developments of technology, such as JavaScript embedded form extraction. With the minimum query conditions matching and the feedback modules, we can build a high accuracy, high efficiency matching system.

The Information Integration has been studied for a long time. The early works focused on the traditional Database Integration. Li and Clifton [21] developed a semi-automatic semantic integration procedure (SEMINT) which can find the corresponding attributes. Batini and Lenzerini [2] introduced the conceptual foundation to the problem of schema integration, which integrates the different individual methodologies and gave a general guideline for future improvement. The

 ISBN: 978-1-61208-110-6

traditional database integration is often divided into two steps, the View Integration, which produces a global conceptual description of a database, and the Database Integration, which produces a global schema of the databases.

Since the late nineties, the $deep\ web$ information integration(or Web Database Integration) is coming into the view of the researchers. Early stage research focused on small-scale, reconfigured, and manual intervened systems. The recent works focus on dynamic, large-scaled systems. As the web databases usually provide a integrated query interface to let users query the databases, we may consider that a global conceptual description for each database already existed. We do not need to make such "View Integration" for Web Databases. On the other hand, in comparison with the traditional databases integration problem, we do need to "understand" or "get" the attribute descriptions from the query interfaces and query result pages. Suppose a global schema is built correctly, the search results should be also correct. Therefore the Web database Integration can be viewed in two parts, the Query Interface Integration(QII) and the Query Result Integration(QRI). Rahm and Bernstein [6] gave a survey of the approaches of traditional automatic schema matching.

The research QII is mainly based on the visual elements identification. Golin and Reissa [8] gave a specification of visual language. Zhang ([19] and [22]) gave us an approach to understand query interfaces: Best-Effort parsing with Hidden Syntax. Liu and Meng [20] presented another approach $ViDE$ based on calculating the $distance$ among visual elements. Chuang and Chang [17] introduced a context aware wrapping system which contains the peer sources to facilitate the subsequent matching and to improve the extraction accuracy. The matching between different interfaces is often determined by calculating the semantic matching times (in results of search). He and Chang [13] introduced an approach named DCM framework. This approach tried to build a complex matching which matches a set of m attributes to another set of n attributes. Madhavan and Bernstein [7] introduced a new algorithm, Cupid Matchers in comparison with the traditional schema matching, such as Linguistic based Matchers, Constraint based Matchers, Individual matchers and Combinational matchers. He and Meng [10] concerned with the E-commerce interfaces and proposed a weight based matcher. His approach tried to automatically construct a global interface and the global attribute from the query interfaces. Wang and Wen [12] proposed an approach based on query probing and instance-based schema matching techniques. He separated the schema-matching into two areas: intra-site and inter-site. In other words, the matching between results and query interfaces in the same source and the matching between different sources.

The Web data extraction is a little bit different from interface extraction. Laender and Silva [1] gave a survey of traditional web data extraction tools. These tools are often based on declarative languages, HTML structure analysis, natural languages process, machine learning, data modeling and ontology. All of these tools focused on the treatment of HTML script. Chang and Hsu [4] introduced a method to trait the HTML code and to find the repeated patterns in the query result pages. Hu and Meng [5] introduced a semantic blocks, section and data items identification system. The data items which have the same role were mapped.

Our work has several main differences from the other works. First, the new web-page parser can execute Javascript and analyze the $true$ web-page by visual relationships. Second, we are not trying to match every attributes or query conditions in every query-interface. Only the most closed query conditions will be matched. In other words, through all the query-interfaces, there exists a minimum query-condition group which permits to construct a query and to get all the records from the back-end databases. At last, a feedback system based on automation theory of Nonlinear Discrete Systems was never discussed before.

III. Automatic Web-database Integration System Architecture

In order to build such a domain based, automatic Web-database integration system, we decompose it into several parts:

1) Web database crawler: finds the web database of specific domain.
2) Interface Extraction: parses the query form to the group or element trees.
3) Interface Integration: applies to a domain specific semantic ontology and builds a mapping from the query form contents to the semantic model.
4) Query result integration: parses and matches the different result lists from the different web query interfaces.
5) Collection system: collects the responses of end-users by detecting the records rank of the search results. A click is a user action when he clicks the hyper link in the result page. We construct a table to save the times of clicks for each record of search result. The record rank can be built from this table. The mismatching query conditions and query results normally should have a record rank much lower than the correct ones.
6) Feedback and Ranking System: includes the search results ranking, element group matches ranking, interface parsers ranking, and the web database sites ranking. The accuracy and the integrity will be gathered from the clients and send to the Ranking system. The ranking system will then $FeedBack$ to the formal subsystems, like $Interfaces\ Extraction$ subsystem, $Interfaces\ Integration$ subsystem and $Query\ result\ Integration$ subsystem.

Web database crawler may be viewed as a traditional search engine with some predefined characters. This crawler can travel through the web and identify the query interfaces. With the study of [14], the depth of a web database is often very limited. According to their work, 94% Web database have a depth within 3, these database are found from 1,000,000 randomly selected IPs. Our approach to this subsection is divided into 2 steps: First, a traditional crawler finds the site root pages which contain the web database. Secondly, a JavaScript concerned crawler find out all the query interfaces in every site. $JavaScript\ concerned$ is a conception in comparison with the traditional site crawler. The traditional crawler is designed to get the static HTML code from the static links of pages. With the evolution of JavaScript and the new technology, the original HTML code does not contain all

 ISBN: 978-1-61208-110-6

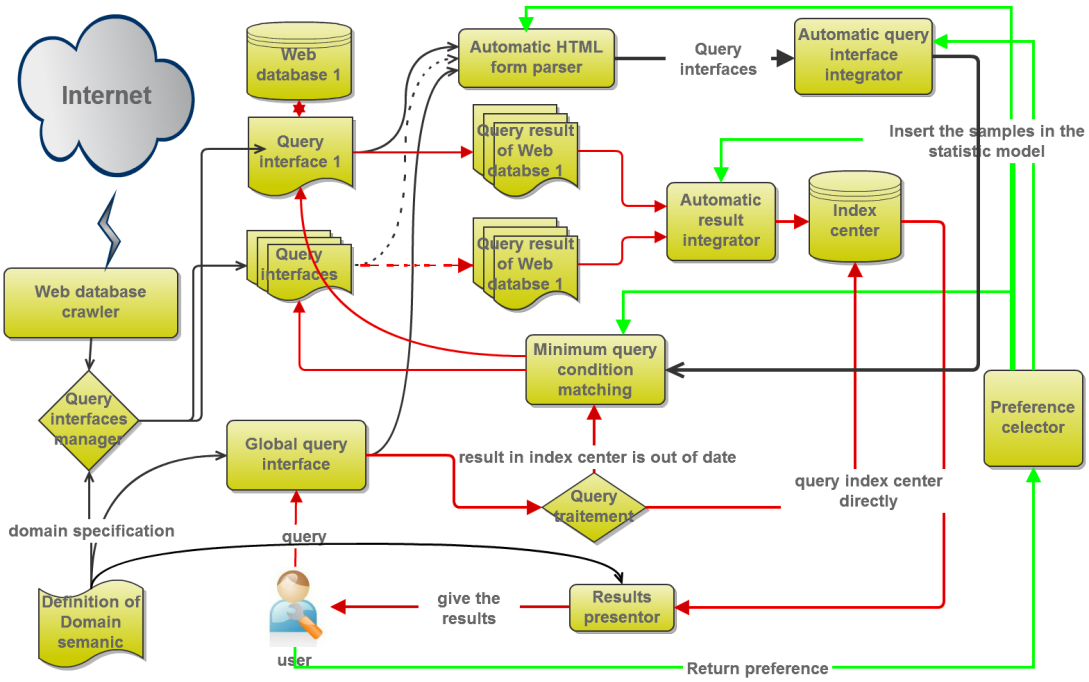

Figure 1. Web Query Interface Integration System

the information that we really see from the navigator. The final HTML code is often treated by the initialization Javascript and could be modified by the end-user's action.

A layout engine, which can execute JavaScript from a URL and simulate the end-user's action, is necessary. The most popular layout engines are Trident, WebKit and Gecko. Trident is used by Internet Explorer, WebKit is the core of the Safari and Chrome and Gecko is used in Mozilla Firefox. All the layout engines share the same idea:

- Get resources from a URL include CSS, HTML, JavaScript, image, video, etc.
- Construct the DOM tree and Render Tree. DOM tree contains all the nodes of which should be showed in the navigator. Render tree contains the visual definition of the nodes in DOM tree.
- Draw the elements of DOM Tree and take the description from Render Tree.

The new crawler embeds a layout engine to get resources from a URL and to generate the DOM tree and the Render tree. The last step, which draws the DOM elements, and consumes most of the memory and time for a layout engine, will not be executed. The DOM tree will be gathered by our crawler and query interfaces could be detected after the analysis with the semantic ontology. The render tree could also be used for further analysis, like the position of elements, size, etc.

The Interface Extraction system will focus on *understanding* the web query interfaces. The Interfaces Integration system will focus on matching the groups of query conditions from the different query interfaces. The traditional schema matching process has been focusing on identifying semantic relationships between two attributes. Why do we talk about the query conditions matching instead of the elements/attributes matching? We should take into consideration the purpose of interface integration, which is to build a mapping between query interfaces, to enable a query condition pass through all of them. Not only the query capabilities of query interfaces may not be equivalent, but also the matched attributes/elements may play different roles in the different web databases. Therefore the objective is to find equivalent query conditions but not equivalent *attributes*. Furthermore, how could we guarantee the precision and integrity of the matching, not only for the query conditions in all the interfaces, but also for the query records from the different result pages? How could we associate the responses of end-users to the interface extraction system, interfaces integration, and query-result integration?

Figure 1 describes the relationships between the function modules. The advantage of this structure is if we change the domain, the only thing that we need to modify is the manually collected Semantic Ontology. The process in Figure 1 can be simply denoted as four parts:

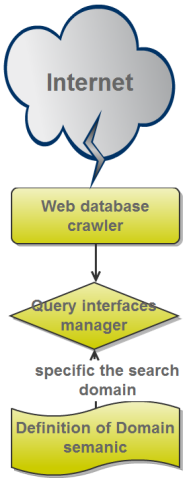

Figure 2. Web interface crawler

- Query interface crawler find the site root that contains web databases. It is showed in Figure 2
- Analyzes the structure of a known formula and then realize a semantics association between the parameters of the semantics model and the interface elements to simulate the client's query request. This part is indicated

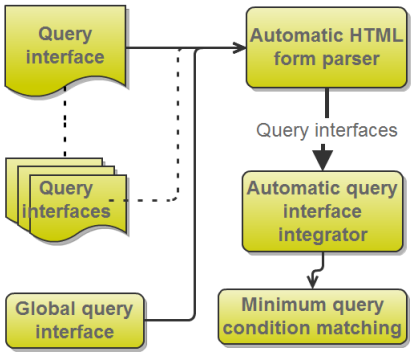

Figure 3. Interface integration

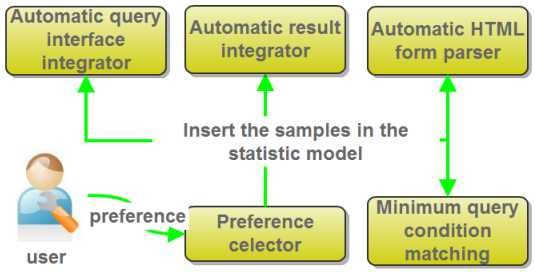

Figure 4. Feed-back system

by black arrow line and showed in Figure 3. It shows how we treat and merge the interfaces with the global interface by the definition of semantic clusters. This interfaces integration system focus on a so called "Minimum Query Condition Matching". This matching does not attempt to match every attribute pair found. It tries to find a *minimum* query set across all the interfaces, which has the highest accuracy and allows our system to get all information from the back-end databases by the web query interfaces.

- Analyzes and parse the results page: it defines the structure of a result page and then identifies the relevant result fields and their associations with the semantics model parameters. This part is indicated by red arrow line.
- Analyzes the feedback from end-users and modifies the preferences of query interface matchers and query result matchers. It involves users into the matching system. Such kind of feedback mechanism can correct and improve the likelihood of successful mapping. This part is indicated by green arrow line and is showed in Figure 4.

IV. Interface Extraction

Query interfaces hide the data behind them from direct access. A form extractor is a prerequisite for the mapping works. Traditionally, we use a wrapper induction program which is supplemented by a GUI for users to click and highlight strings on a rendered Web page to produce a training example. These training examples help the program build up the patterns of the forms. If we can build all the patterns of the forms and give them relationships correspondingly, we can translate the queries from one interface to another. This kind of work is time consuming. With the development of the network, the "manual" analysis is become impracticable. An *automatic* extractor is thus required to a true web database query system. As a general automatic extraction (in all different domains) is almost impossible for an all-in-one integration, the recent works of automatic extraction is always "domain-based". We enrich the study of Zhang [19] which proposed a "Best-effort" parsing with "Hidden Syntax". It treats the web interface in a "visual" way and introduces a "Hidden Syntax" between the positions and the relationships. The rules, named "patterns specification", are given and the way of extraction, named "pattern recognition", is specified. From the human point of view, this approach is much better than HTML pattern extractors because all the interfaces are designed for visual effect. The HTML code often contains the clues for programmers. As a result, the related tags may be placed scattered or the attributes and the corresponding elements may not have the same name. However, the visual presentation is always the main purpose of form design. So if we can make a system which extracts the visual elements, the accuracy will be guaranteed. In fact, the syntax or grammar of visual language is not a huge set and can be defined properly. On the other hand, the "pattern recognition" is more flexible. A more general and precise method will be proposed in the following paragraph.

A. Hidden syntax

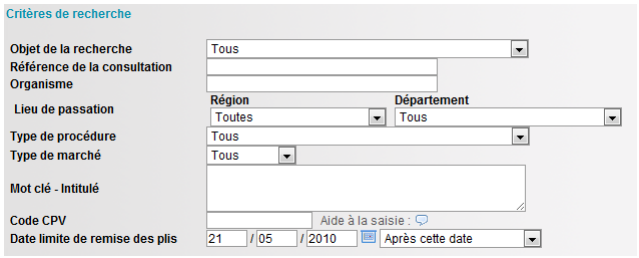

Figure 5. Query interface:Achatpublic.net

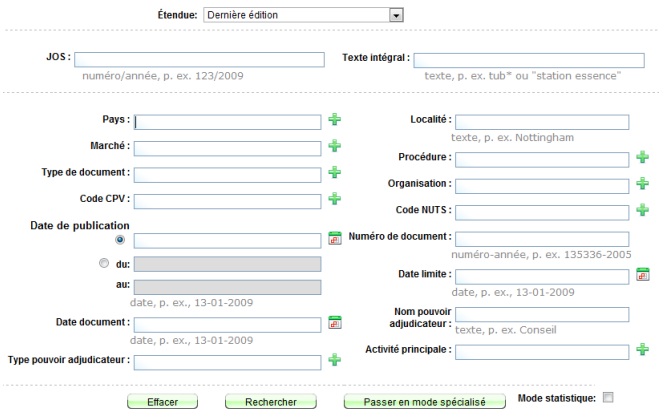

Figure 6. Query interface:TED

Suppose we have two different query interfaces as Figure 5 and Figure 6 in the domain of *Government Procurement*. We could easily find out some simple rules: the attribute is always on the left or above the input element. They are often in the same line. Normally, this kind of formula is built for human to read and understand. There exist many thorough studies in the domain of diagram analyzing. As Zhang [22] discussed, it assumed that there is a relationship between the semantics and the presentations behind the query-form creation, which is guided by some hidden syntax based on topology and proximity. So we can easily get a set of *pattern specification*,see [8]. On the other side, the form design also has some hidden syntax in the style of the HTML code for the query-form. He and Meng [9] has discussed some of the

 ISBN: 978-1-61208-110-6

rules of the HTML code. In fact, the $pattern\ discovery$ has been extensively studied, like [4] and [18]. But the main idea of their works is to find out the maximum repeated patterns which are mostly used in the query result. Normally a query-form does not contain any $repeated\ patterns$, so the method of $find\ repeat$ cannot be used directly. However, on the other hand, the definition of the pattern is useful for us if we consider the visual analysis at the same time. Therefore the **Hidden Syntax** of us is defined by a combination of the $pattern\ specification$ of HTML container and semantic relationships of the visual elements. The definition of hidden syntax could be defined as following:

The raw map of production rules: A production P in V/S grammar Σ, N, s, P_d, P_f is a four-tuple H, M, C, F. Head $H \epsilon N$ is a non-terminal symbol. Components $M \subseteq \Sigma \bigcup N$ is a multi set of symbols. Constraint C is a Boolean function defined on M. Constructor F is a function defined on M, returning an instance of H.

The rules definition are well studied in the work of [19] and [11]. The differences of the V/S grammar are that:

- P_d does not only contains the rules of visual elements, but also contains some rules of HTML patterns.
- P_f does not only contains the preference beyond the grammar, but also contains the preference by a method of the result of searching and matching.

The definition of symbol: The smallest visual elements $\bigcap$ the smallest HTML tag elements. For example, Figure 7 and Figure 8 represent the two different symbol groups for the same query condition, i.e., date range. For the 7, we find the first three input elements that are connected by '/'. The fourth input element does not close to any attribute. So we can associate the first three input elements as a date input and the fourth is a selection. These four elements with the attribute build a group.

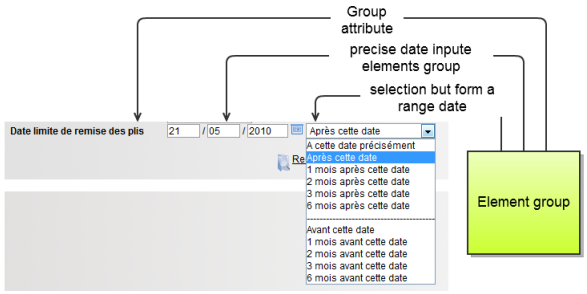

Figure 7. Date analysis: Achat public.net

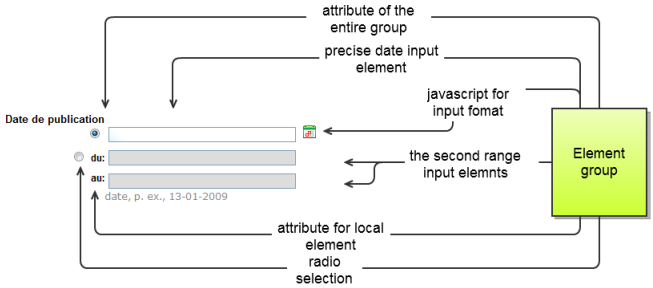

Figure 8. Date analysis: TED

B. Preferences

A preference of Σ, N, s, P_d, P_f is defined in I, U, W

- conflicting instances $I = A, B$ where $A, B \epsilon M$
- conflicting condition: a Boolean expression on A, B
- probability criteria W: a probability function on A, B that specifies the winner or both of them.

W should be considered as two branches. A threshold τ should be considered:

- $W > \tau$ and $W < (1 - \tau)$: A and B should be taken as 2 branches.
- $W < \tau$ or $W > \tau$: A and B should be taken as a confect and take A if $W < \tau$, take B if $W > \tau$

For example, in the Figure 8, the attribute "du" is very close to the ratio button, while it is also very close to the input element on its right hand. Which one should be chosen? Considering the semantic relations, it should connect to the attribute "au", and next to "au" there is only one input element on its right hand. So we prefer to link the "du" attribute with the input element on its right hand. Furthermore, we could even link the attribute "du" with the ratio button. When we try to match a range date, the attribute "du" will associate with a value 1/0, which cannot make any valuable query condition in date search. Formally, the preferences are constructed by three parts.

1) preference from the topology
2) preference from the constraint of HTML TAG
3) probability function from the responses of end-users

With the preferences, we need a parser to generate an original parse forest which eliminates all the obvious conflicts and ambiguities. In this parser, a probability function, which gathers the information from end-users, will influence the parse tree. Suppose we get $n\ pair$ different parse trees after the parse process. Each pair is decided by the three constraints. If the parse tree still contains some different parse results after we have pruned all the obvious conflicts and ambiguities, how could we decide the preferences? To answer this question, we need to know only the end-users can decide which query result is what they want. Hence, the question is how can we allow the end-user influence the preferences of the Parser. We build a subsystem to gather the feedback from end-users and build preferences for each combination of nodes. If we can choose a proper threshold of the preference, then we can choose the parse condition and resolve the conflicts or ambiguities.

C. Dynamic page

There is a problem that has never been well considered in Interface Extraction, i.e., the dynamic query interfaces which contain java script. The query interfaces often have some java script or even Ajax to modify the form structures and the form contents. For example, Figure 10 is the result of a selection from 9. Before we select a value in the selection tag, we only have a form which contains one element: select tag. After we chose something in the selection tag, we get a totally different form to extract.

Figure 9. Before choosing the option

Take account of the JavaScript, we need a layout engine which allows us to execute of JavaScript. Also, we need this layout engine to simulate the actions of a navigator such as click, drag and drop etc.. After each step of the action that we simulate, the whole parse tree should be rebuilt with grammar

 ISBN: 978-1-61208-110-6

Figure 10. After choosing the option

Σ, N, s, P_d, P_f. By comparing with the original parse tree, we got a new one. The new nods construct a new sub tree which can be pasted to the main tree. If there are several different results of JavaScript, we consider that the query interface has several different equivalent parse tree. The selection value should be considered as a search condition such as in Figure 9 and Figure 10.

V. Interfaces Integration

Query interfaces integration is a fundamental problem of Web Database Integration system. After the studies of related works, the types of matching can be described as: Query interfaces matching, Query results matching, Multi interfaces complex matching [13], Query interfaces and Query results matching. We can consider these kinds of matching is a type of $flat$ matching because they do not pay attention to the feedback of end-users. Meanwhile, if we also consider the research presentation, there is a big misunderstanding in Query Interfaces matching. The Web database integration does not really need to **integrate** or **match** every attribute in every Web database Interface. Logically it is impossible if we concern about the differences of the query abilities among the Query interfaces. Due to the purpose of our system, forming a unified result to end-users, we only need to integrate/match the main attributes/query conditions or the $clue$ of elements/attributes groups which enable our query system to query through all the Web databases and get all the information behind the interface. The purpose of interface integration is not to find the entire attribute pairs or group pairs and match them. On the contrary, the purpose is to find a minimum matching set to query them. Hence our Web database integration system should solve the following two problems:

- how can we find the minimum set of attributes which allow us to query all the information of the database. This attribute set could be the most easily matched set.
- How to match them.

To answer these two questions, we assume every pair of matching between global interface and web query interfaces have preferences, and the groups of the matching of all interfaces have preferences. A very high threshold of preference is defined as default and a small subset of matches can be selected. The selected matches are used to be the minimum matching set to query the web databases. The results-queries matching will be executed. For instance, if the results do not have matching with the queries for one of the Web Database, we consider our minimum subset is not suitable for it. The system will cut down the threshold and rebuild the minimum subset of matches.

A. Interface Analysis

First, elements of the query interface have generally the following types of format: **text box, ratio button, check box, and selection list**. Text box allows an input ”infinite”. A selection list provides a pre-determined value which is ”finite”. Radio box is like a single-select list. Check box is like a multi-select list. So we can say there are only two types of format: String S/ array()K, where S is infinite and array K is finite. Secondly, the elements have some types of relationships that need to be grouped together:

- Range type: it refers to the situation where two or more elements are used to specify the range.
- constraint type: like radio box: last name / first name should be grouped with element which has an attribute ”name”. An element could be used as a constraint for another element. The constraint could be ”or” if they are radio box, and could be ”and” if they are check box.

B. Semantic relationships for elements matching

We identify three semantic relationships among elements or the groups of elements. Suppose they have some kind of $matching$. If they do not have any relationships or exclusive relationships, it is not discussed here.

- **Synonymy** $T1$ is a synonym of $T2$
- **Hypernymy** $T1$ is more generic than $T2$,denote $H(T1, T2)$
- **Meronymy** $T1$ is a part of $T2$, denote $M(T1, T2)$

There is a case that should be noticed: T1 is a group of two text boxes, attribute is ”price”. T2 is an element of select list, attribute is price, the list is (0-100,100-200,200-400,¿ 400) we can say they are Synonymy because they play the same role in the query system. But all the option of T2 is a Meronymy of T1. For example, T1 condition is 0-300, the correspondence of T2 should be 3 query combined together: (0-100) $\bigcup$ (100-200) $\bigcup$ (200-400). More generally, this phenomenon deduces another important problem: Query translation.

C. Query translation

To translate Q_s from T1 to T2 in the above example, there are 3 query heterogeneities defined in [22]:

- **Attribute level**: Two sources may not support query the same concept or may query the same concept using different attribute names.
- **Predicate level**: Two sources may use different predicate templates for the same concept. For instance, price predicate template in T2 is a different set of value range from T1.
- **Query level**: Two sources may have different capabilities of querying valid combinations of predicate templates. In this case, the query translation is almost $impossible$.

According to the three different query heterogeneities, we can consider our minimum attribute set problem as: a group of attribute which can be matched from all the web database interfaces with no query heterogeneities, e.g., Figure 7 and Figure 8. The capability of Figure 7 can only define a period

 ISBN: 978-1-61208-110-6

of several months before or after a certain date which is given by a formal input value. If we need a period from 01-01-2008 to 15-02-2008, the closest condition is 2 month after 01-01-2008. For the interface of Figure 8, we need to choose the second option of the radio and just give the exact time interval. Moreover, in the query interface of Figure 5 we do not have the condition $pays$ but in the second interface of Figure 6, it has. So the query condition of $pays$ can never be matched between the two interfaces. There are two possibilities: Figure 5 contains only the information of one country, or Figure 5 do not distinguish the property $pays$. We cannot solve the matching problem by just analyzing the semantic means, a statically method based on the feedback system will be introduced in sector 5.

D. Matching discovery

1) Preparation of matching discovery: We take query interfaces as flat schema with sets of attribute entities. Due to domain-specific integration, we need to define a semantic ontology for the domain. Relying on the semantic ontology, a set of groups of attribute can be built from the web query interfaces. According to our study, the projects which are being carried out from other's studies always have a global interface and some kind of groups of attribute. The matching discovery problem is to find out the attribute entities matching to the global interface.

The matching discovery needs another precondition, which is data processing step. The data processing for our system here is attributes normalization. We consider the text of attributes like the natural languages. There are many studies about natural language normalization or attribute normalization such as [13]. The data preprocessing step consists of

- standard normalization [16] is a process of removing the commoner morphological and the flexional endings from words in English. It is mainly used as part of term normalization process that usually done when setting up Information Retrieval systems.
- normalize irregular nouns and verbs
- removes common stop words

2) Merging: All kinds of merging intends to construct the semantic relationship groups. This process will reduce the quantity of the final semantic relationship groups. The less we have the semantic relationships groups, the easier we establish the matching.

Type recognition Sometimes the same attribute name could have different meanings if they are in different types.

Syntactic Merging

- $name\ base\ merging$: It is based on the occurrence frequency of attributes. The statistics result of occurrence will give some preferences of merging rules.
- $domain-based\ merging$: It is based on the occurrence frequency of attribute groups. If some groups are similar and have a high occurrence, even they denote to different meanings, we can merge them together.

E. Matching construction

The approach of [13] is much more convincing than the works of [3] and [10]. He and Chang [13] introduce a ranking system who considers the samples of $N : M$ matching among the query interfaces. It does not consider any influence of the feedback from end-users. The accuracy and the integrity of matching can only be judged by the end-users. We will introduce the matching construction system which includes two steps, $Rank\ and\ selection$.

Given a set of discovered matching candidates, $R = \{M_1, M_2, ..., M_V\}$, the ranked matching is $R_C = \{M_{t_1}, ..., M_{t_V}\}$. For a n-array complex matching M_j in $R : G_{j_1} = G_{j_2} = ... = G_{j_\omega}$, C_{max} the maximal m_n value among pairs of groups in a matching is : $C_{max}(M_j, m_n) = max m_n(G_{j_r}, G_{j_t}), \forall G_{j_r}, G_{j_t}, j_r \neq j_t$.

We rank matching with the following rules:

1) if $s(M_j, m_n) > s(M_k, m_n), M_j$ is ranked higher than M_k
2) if $s(M_j, m_n) = s(M_k, m_n)$, and $M_j \succeq M_k, M_j$ is ranked higher than M_k
3) the $\succeq$ is a semantically subsumption which is described as a "top-k" approach.

In the theory of [13], the ranked matching R_C only considers the relationship between interfaces, the match ratio does not depend on any feedback of the end-users. Our consideration is based on this fact: in the statistical point of view, a correct matching will get more click ratio than a mismatching. Now the problem for us is how to decide the feedback coefficient.

F. Matching Selection

The selection rules are:

1) among the remaining matching in RC, choose the highest ranked matching Mt.
2) Remove matching conflicting with Mt in RC.
3) If RC is empty, stop; otherwise, go to step 1.

The main purpose of these rules is to remove all the conflicts and keep the highest ranked items.

VI. Feedback ranking integration

As the Figure 4 shows, the click ratio of end-users will be gathered by the system. We take the two examples of Figure 5 and Figure 6. The query condition pay is not contained in interface 1 but in interface 2. Suppose end-users execute queries by our global interface of government procurement. They want to find out all the procurement contracts. Suppose Figure 5 contains only the information of French government and Figure 6 contains the information of Europe. The matching between the two interfaces does not contain the condition "pay". Our system will not find any difference between the two web databases at the beginning. After this system has ran for a period of time, the feedback from the end-users will be distinguished. When condition "pay=French" is selected in the global interface, suppose the condition "pay" exist in our global interface, interface of Figure 5 and Figure 6 will give the same result. The feedback system will get the same click ratio of the results from the two databases. When condition "pay!=French" is selected, the click ratio of the result from query interface of Figure 5 get 0 and the click ratio for Figure 6 will be very high. The feedback system will then give the query interface in Figure 5 a precondition "pay=French".

Proposition: Consider every web database is an input, denote $x_1, x_2, ..., x_n$, suppose that we have find n web database sources need to be searched. The request query is an input denote u. After the interface extraction and interface

 ISBN: 978-1-61208-110-6

matcher, each database will give us a search result, denote $y_1, y_2, ..., y_n$. The process of extraction, match denote as $k_{11}, k_{21}, ..., k_{nn}$. The transfer function could be noted as:

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = u \times \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{1n} \\ \dots & \dots & \dots & \dots \\ k_{n1} & k_{n2} & \dots & k_{nn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

This is the transfer function of open-loop. For the close-loop function, it denotes:

$$\begin{bmatrix} y_1^n \\ y_2^n \\ \dots \\ y_n^n \end{bmatrix} = u \times \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{1n} \\ \dots & \dots & \dots & \dots \\ k_{n1} & k_{n2} & \dots & k_{nn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$
$$+ \sum_{i=1}^{n} (\begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{1n} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{bmatrix} \times \begin{bmatrix} y_1^i \\ y_2^i \\ \dots \\ y_n^i \end{bmatrix})$$

Where $c_{11}, c_{12}, ..., c_{nn}$ are the coefficients of feedback function. Specially, $\forall n \epsilon N, \forall j \epsilon n, \sum_{i=1}^{n}(C_{ij}) = 1$. It means the sum of the probability distribution is 1. The difficulty here is how to define the presentation of x and y. If we can find the presentation function of x and y, the stability and the convergence speed can be calculated by the theory of automatic control.

VII. Conclusion

In this article, we have shown some associated mapping approaches for understanding the web interfaces and their limitations. The proposed method needs improvements and tests of performance in real conditions. Several elements are to be assessed:

- Improve the quality of the model: the user interaction allows it to quickly converge to a reliable mapping, which is the speed of convergence and stability.
- Quality of mapping techniques, configuration and optimization.

In addition, unsolved problems have been identified as:

- how can we assure the minimum matching attributes groups are sufficient for querying the web database?
- how can we find out the best parameters to assure the convergence speed and stability of the feedback system?
- how can we specify the transfer function with the known attribute matcher theory?

Further work is to evaluate this approach through a prototype of our case study in the domain of government procurement.

References

[1] A.H.F. Laender, A.S. da Silva, B.A. Ribeiro-Neto, and J.S. Teixeira *A brief Survey of Web Data Extraction Tools* ACM SIGMOD Record, vol. 31, pp.80-93,Jun 2002

[2] C. Batini ,M. Lenzerini, and S.B. Navathe *A Comparative Analysis of Methodologies for Database Schema Integration* ACM Computing Surveys, vol. 18, pp.323-364, Dec 1986

[3] B. He and K.C. Chang *Statistical Schema Matching across Web Query Interfaces* Proc. ACM SIGMOD international conference on Management of data, pp.217-228, 2003

[4] C. Chang, C. Hsu, and S. Lui *Automatic information extraction from semi-structured Web pages by pattern discovery*Decision Support Systems,vol. 35, pp.129-147, Apr 2003

[5] D. Hu and X. Meng *Automatic Data Extraction from Data-rich Web Pages* DASFAA , LNCS 3453, pp.828-839, 2005

[6] E. Rahm and P. Bernstein *A survey of approaches to automatic schema matching* VLDB, vol. 10, pp.334-350, Dec 2001

[7] J. Madhavan ,P. Bernstein, and E. Rahm *Generic Schema Matching with Cupid* Proc. of the 27th International Conference on VLDB, pp.49-58, 2001

[8] E.J. Golin and S.P. Reissa *The specification of Visual Language Syntax* Journal of Visual Languages and Computing, vol. 1, pp.141-157, Jun 1990

[9] H. He, W. Meng, C. Yu and Z. Wu *Constructing Interface schemas for Search Interfaces of Web Databases* WISE, LNCS 3806, pp.29-42, 2005

[10] H. He, W. Meng, C. Yu and Z. Wu *WISE-Integrator: An Automatic Integrator of Web Search Interfaces for E-Commerce* Proc. of the 29th international conference on VLDB, vol. 29, pp.357-368, 2003

[11] G. Costagliola and V. Deufemia *Visual language editors based on lr parsing techniques* 8Th International Workshop On Parsing Technologies, IWPT'03, pp.79-90, 2003

[12] J. Wang, J. Wen, F. Lochovsky and W. Ma *Instance-based Schema Matching for web databases by Domain-specific Query Probing* Proc. of the 13th international conference on VLDB, vol. 30, pp.408-419, 2004

[13] B. He and K.C.C. Chang *Automatic complex Schema Matching across web Query Interfaces: A Correlation Mining Approach* ACM Transactions on Database Systems, vol. 31, pp.346-395, Mar 2006

[14] K.C.C Chang, B. He, C. Li, M. Patel and Z. Zhang *Structured databases on the web: Observations and Implications* ACM SIGMOD Record, vol. 33, pp.61-70, Sep 2004

[15] M.R. Genesereth and A.M. Keller *Infomaster: An Information Integration System* Proceedings of the 1997 ACM SIGMOD international Conference on Management of Data, pp.539-542, May 1997

[16] Martin Poter *The porter stemming algorithm* http://tartarus.org/\~martin/PorterStemmer/ unpublished

[17] S. Chuang and K.C.C Chang *Context-Aware Wrapping: Synchronized Data Extraction* Proc. of the 33rd international Conference on VLDB, pp.699-710, Sep 2007

[18] Y. Yang and H.J. Zhang *HTML Page Analysis Based on Visual Clues* Proc. 6th International Conference on Document Analysis and Recognition, pp.859-864, Sept 2001

[19] Z. Zhang, B. He and K.C.C Chang *Understanding Web Query Interfaces: Best-Effort Parsing with Hidden Syntax* Proc. of the 2004 ACM SIGMOD, pp.107-118, June 2004

[20] W. Liu, X.F. Meng, and W. Meng *ViDE: A Vision-based Approach for Deep Web Data Extraction* IEEE Trans. on Knowl. and Data Eng, vol. 22, pp.447-460, Mar 2010

[21] W.S. Li and C. Clifton *SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks* Data & Knowledge Engineering, vol. 33, pp.49-84, 2000

[22] Z. Zhang *Large scale information integration on the web: finding, understanding and querying web databases* unpublished, PhD thesis, 2007

 ISBN: 978-1-61208-110-6