Understanding Deep Web Search Interfaces: A Survey

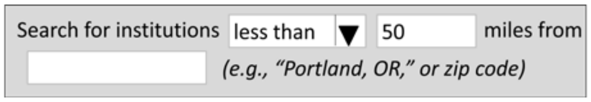
Ritu Khare Yuan An Il-Yeol Song
The iSchool at Drexel, Drexel University, Philadelphia, USA
{ritu, yan, isong}@ischool.drexel.edu

ABSTRACT
This paper presents a survey on the major approaches to *search interface understanding*. The Deep Web consists of data that exist on the Web but are inaccessible via text search engines. The traditional way to access these data, i.e., by manually filling-up HTML forms on search interfaces, is not scalable given the growing size of Deep Web. Automatic access to these data requires an automatic understanding of search interfaces. While it is easy for a human to perceive an interface, machine processing of an interface is challenging. During the last decade, several works addressed the automatic interface understanding problem while employing a variety of understanding strategies. This paper presents a survey conducted on the key works. This is the first survey in the field of search interface understanding. Through an exhaustive analysis, we organize the works on a 2-D graph based on the underlying database information extracted and based on the technique employed.

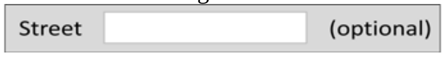
1. INTERFACE UNDERSTANDING
The Deep Web consists of data that exist on the Web but are inaccessible by text search engines through traditional crawling and indexing [17]. The most popular way to access these data is to manually fill-up HTML forms on search interfaces. This approach is not scalable given the overwhelming size of Deep Web [3]. Automatic access to these data has gathered much attention lately. This requires automatic understanding of search interfaces. This paper presents a survey on the major approaches to *search interface understanding* (SIU) and is the first work to present an assessment of this field.

The Deep Web is characterized by its growing scale, domain diversity, and numerous structured databases [9]. It is growing at such a fast pace that effectively estimating its size is a difficult problem [9, 19, 27, 18]. In the past, researchers have proposed many solutions to make the Deep Web data more useful to users. Ru and Horowitz [25] classify these solutions into 2 classes: dynamic content repository, and real-time search applications. We extend this classification scheme by defining 3 goal-based classes: (i) solutions to increase the content visibility on text search engines, such as collecting/indexing dynamic pages [21, 16] and creating repository of dynamic page contents [24, 29, 26]; (ii) solutions to increase the intra-domain searchability, such as meta-search engines [32, 8, 23, 5, 30, 11]; (iii) solutions to accomplish knowledge organization, such as ontology derivation [2].

Automatic understanding of Deep Web search interfaces is the pre-requisite to attain any of the aforementioned solutions. Deep Web contains at least 10 million high quality interfaces [20] and automatic retrieval of such interfaces has also received special attention [1]. Search interface understanding is the process of extracting semantic information from an interface. A search interface contains a sequence of interface components, i.e., text-labels and form elements (textbox, selection list, etc.). An interface is primarily designed for human understanding and querying. It does not have a standard layout of components (see Figure 1) and there is infinite number of possible layout patterns [7]. Thus, while human users easily perceive an interface based on past experiences and visual cues, machine processing of an interface is challenging.


Search for institutions less than ▼ 50 miles from
(e.g., "Portland, OR," or zip code)

a. An interface segment from Education domain


Street (optional)

b. An interface segment from Healthcare domain

Figure 1. Diversity in Interface Design

A search interface represents a subset of queries that could be performed on the underlying Deep Web database. In data-driven Web applications, a user-specified query is translated to a structured format, such as SQL, and is executed against the online database. The placement patterns of components provide information on implied queries. For instance, the text-label 'Street:' and the adjoining textbox in Figure 1b imply the WHERE clause of an SQL query, i.e., "WHERE Street='XYZ'." Additionally, the textual contents on an interface provide information on the underlying database schema [2]. For instance, the data entering instructions such as '(eg. …' and '(optional)' in Figure 1, provide insights on data and integrity constraints, respectively.

Motivated by the opportunities provided by search interfaces, several researchers address the SIU problem. This paper presents the results of a survey conducted on the key SIU approaches. While different approaches employ different understanding strategies, a sequence of activities is common to all. Figure 2 shows the four stages of the SIU process.

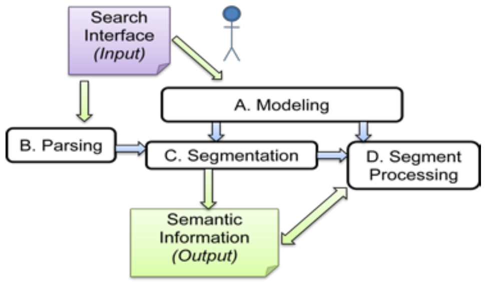


Figure 2. Search Interface Understanding Process

Input: A Search Interface
Output: Semantic Information
Stages:
(A) Modeling: First, human modelers formalize a model for the interface. Most of the works model an interface as a sequence of implied queries against the underlying database. Each query is a group of logically related components, hereafter known as a *segment.* Different approaches assign different *segment labels* to segments. For instance, Zhang et al. [33] use "conditional pattern" as the segment label and would represent Figure 3's interface as a sequence of two conditional patterns. Components in a segment might also be assigned query roles, known as *semantic labels.* Zhang et al. [33] use 3 semantic labels: "attribute-name," "operator," and "value." Furthermore, an interface might be modeled to contain constraints information about the underlying database schema. The work in [26] models information such as domain, invisible and visible values of form elements.
(B) Parsing: Then, an interface is parsed into a workable physical structure. He et al. [10] parse an interface into a string, "interface expression," with 3 constructs: 't', corresponding to any text; 'e', corresponding to any form element; and '|', corresponding to a row-delimiter. The interface in Figure 1a would be parsed as "teet|et."
(C) Segmentation: The query information, modeled in the modeling stage, is extracted in this stage. The interface is divided into segments where each segment corresponds to a query implied by the underlying database. Zhang et al. [33] use a rule-based method to group components into segments and assign semantic labels to components. The lower segment in Figure 3 is created by grouping 3 components: "Gene Name," radio button group, and textbox.
(D) Segment-Processing: This stage focuses on extracting data and integrity constraints of the underlying database. For each component, He et al. [10] extract meta-information, such as domain type and unit, using machine learning classifiers.

The semantic information identified in stages C and D is the output of the SIU process. The next section describes the settings adopted for conducting the survey.

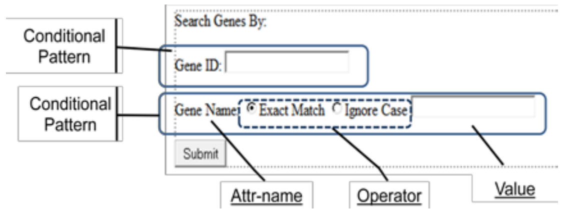


Figure 3. Segmentation by Zhang et al. [33]

2. SURVEY SETTINGS

We surveyed 10 major SIU approaches based on certain manually derived *dimensions*. A dimension is a feature of the SIU process that helps in understanding a work, and in distinguishing it with its counterparts. The survey was conducted in two phases: reductionist analysis, and holistic analysis. In the reductionist analysis phase, each work was decomposed in two ways, in terms of the four SIU stages, and in terms of the *stage-specific dimensions*. In the holistic analysis phase, each work was studied in its entirety using *composite dimensions*, created using the stage-specific dimensions. Section 3 presents the results of reductionist analysis. Section 4 presents the results of holistic analysis.

3. REDUCTIONIST ANALYSIS

We initiated the survey with the reductionist analysis phase. Each work was studied in terms of the stages of the SIU process; and in terms of the stage-specific dimensions. Each of the Sections 3.1 through 3.4 focuses on a specific stage of the process. Section 3.5 discusses the evaluation schemes employed by the surveyed approaches.

3.1 Modeling

In this stage, an interface is modeled into a formal structure suitable for machine processing. This stage was studied under the following two dimensions.

Information on Implied Queries: This dimension denotes the information related to queries implied by an interface. An interface contains multiple segments, each corresponding to an implied query. The surveyed works use a variety of segment labels to refer to a segment. The segment label adopted by *LITE* [24] and *LEX* [10] is "logical attribute." These works model an interface as a list of queries, each specific to an underlying database table attribute. Segment contents for *LITE* include a form element and a text-label. *LEX* models a segment to have a text-label, multiple form elements, and an optional text-label associated with each form element. It assigns the semantic labels "attribute-label," "domain/constraint element," and "element label," respectively, to these components.

The work on *Hidden Syntax Parser (HSP)* in [33] adopts "conditional pattern" as the segment label. Each conditional pattern represents a query capability of the underlying database. A conditional pattern consists of a text with a semantic label "attribute name," a form element with label "operator," and a form element with label "value." The model adopted in [13] is also similar in that it represents an interface as a sequence of segments and uses "attribute-name," "operator," and "operand," as the semantic labels. Along with modeling segments corresponding to a query, Benslimane et al. [2] creates groups of segments, "structural units," each corresponding to a logical entity in the database schema.

Certain works do not explicitly assign any labels to segment or segment components, but do mention the segment contents. Kaljuvee et al. [12], *LabelEx* [22], and *DEQUE* [26] model a segment to consist of a text-label and one or more form elements. Dragut et al. [6] and *ExQ*[31] present a novel way of modeling an interface as a tree structure having arbitrary number of levels. Both these works create groups and sub-groups of related form elements and text-labels and hypothesize a hierarchical structure. Each internal node of the tree represents a text-label and has a group of related form elements as its descendants.

Hereafter, the works by Kaljuvee et al. [12], Benslimane et al. [2], Khare and An [13], and Dragut et al. [6], are referred to as *CombMatch*, *FormModel*, *HMM*, and *SchemaTree*, respectively.

Information on Constraints: This dimension denotes the information related to data and integrity constraints of the underlying database. *HSP* and *LabelEx* model a form element to have a domain of values. *DEQUE* models a form element to have domain, invisible and visible values. *LEX* models a segment to have a domain type and a default value. It models a form element to have domain type (finite, infinite, Boolean), and a unit ($, grams, days, seconds). *FormModel* includes the relationship among structural units, constraints, and the underlying source information. *HMM* models miscellaneous texts which might include information on constraints.

3.2 Parsing

Parsing marks the beginning of automatic processing and brings the interface into workable physical structure. While representation provides a logical image to an interface, parsing physically reads the interface components. Parsing strategies were studied under the following 3 dimensions.

Input Mode: The input to the parsing stage can be in two modes: HTML source code of an interface, and its visual counterpart, i.e., an interface as viewed on a Web browser. *CombMatch*, *LEX*, *FormModel*, and *HMM* use HTML code as the primary input. Along with HTML code, *LabelEx*, *LITE*, *HSP*, *DEQUE*, *ExQ*, and *SchemaTree* use layout engines to extract the visual features such as pixel distances between components.

Description: This dimension refers to the tasks performed while parsing an interface. *LITE* parses an interface in the "Pruning" stage wherein the components that directly affect the layout and labels of form elements are isolated from the rest. *CombMatch*, in its "Chunk Partitioning" stage, segments an interface into chunks delimited by HTML and TABLE cell tags. *LEX* develops an "interface expression" that looks like 't|eee|te|ee|ttee|eet|'. *HSP* parses a page into a set of tokens using its module, "Tokenizer," and stores information such as name, layout position, etc. *HMM* creates a DOM tree of interface components and traverses the tree in depth-first order. *SchemaTree*, in its "Token Extraction" module, creates lists of text tokens, field tokens, and image tokens, and also stores the information about their bounding boxes.

Purgation: This dimension enlists the components that are removed while parsing to avoid information overload on subsequent stages. *LITE* discards images and text styling information. *FormModel* and *CombMatch* remove stop words and text formatting tags. *DEQUE* ignores the components that correspond to font size, typefaces and styling information. *HMM* ignores all the components except the form elements and the text-labels.

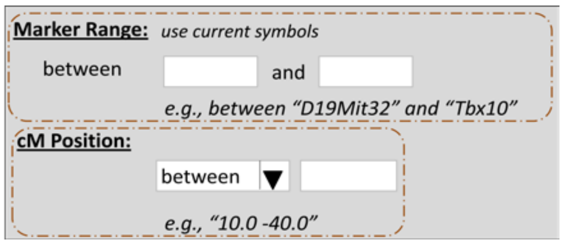


Figure 4. Segmented Search Interface

3.3 Segmentation

After a suitable logical representation and a physical structure are accomplished, the interface is segmented, i.e., the information regarding the implied queries is extracted from the interface. Figure 4 shows a segmented interface having 2 queries. This stage was studied under the following dimensions.

Segmentation Tasks: Segmentation can be visualized as a 3-task process. The first task, *text-label assignment,* involves associating a form element with a surrounding text-label, e.g., associating “cM Position:” with the form elements, selection list, and textbox, in the bottom segment of Figure 4. The second task is *grouping* where the related interface components are grouped together to form a segment. In Figure 4, the 4 components (‘cM Position:,’ selection list, textbox, and ‘e.g., “10.0-40.0”’) belong to the same atomic query and are hence grouped together. In the third task, *semantic labeling,* labels or query roles are assigned to individual components of a query. Automatic text-label assignment and grouping are difficult due to diversity in Web design. Automatic semantic labeling is difficult as Web designers usually do not assign explicit labels in the HTML source code.

A majority of the works (*LITE*, *CombMatch, DEQUE, LabelEx*) only address the text-label assignment problem. *LEX* groups related text-labels and form elements together into “logical attributes”. *HSP* finds groups of “conditional patterns.” *LEX*, *HSP,* and *HMM,* perform grouping as well as semantic labeling. *LEX* also identifies the “exclusive attributes” on an interface based on a domain-specific vocabulary. *SchemaTree* performs text-label assignment and creates segments and sub-segments resulting into a tree of interface tokens. *ExQ* extracts the grouping information of an interface into an unlabeled tree structure and then performs text-label assignment to generate a labeled tree.

Segmentation Technique: Segmentation techniques, i.e., the mechanisms to segment an interface, belong to 3 categories: heuristics, rules, and machine learning.

Heuristic-properties are of 3 kinds: textual, styling and layout. Textual properties include text length, no. of words, string similarity, element’s HTML name, etc. Styling properties include font size, font type, form element format, etc. Layout properties include position of a component, distance between two components, etc. To perform text-label assignment *LITE* exploits all 3 kinds of heuristics. *CombMatch* uses a combination of 8 different algorithms leveraging the 3 kinds of heuristics to assign text-label to a form element. *DEQUE* and *LEX* perform text-label assignment based on the textual and layout properties of components. In *LEX,* all the form elements associated with same text and the text itself are assigned to one segment. Based on heuristics, it also assigns the semantic labels, “attribute label,” “constraint element,” “domain element,” and “element label” to the components.

A rule is a formalized heuristic. Rule-based techniques employ techniques such as regular expressions, grammar, finite state methods, and create rules for associating a form element with a surrounding text. *HSP* assumes that a hidden syntax guides the presentation of interface components on a query interface. The identification of segments and semantic labels is performed using a grammar. The grammar rules are based on layout properties and are derived using pre-studied examples.

SchemaTree uses both rules and heuristics. A tree of fields is built based on the layout properties of form elements, and a tree of text tokens is built based on the layout and styling properties of the text-labels. Then, the two trees are integrated based on some common-sense rules, to generate a complete schema tree corresponding to the interface.

Recent years have seen an advent of machine learning techniques in the field of interface understanding. *LabelEx* employs supervised machine learning to assign labels to form elements. It designs a “Classifier Ensemble” using Naïve Bayes and Decision Trees classifiers and employs both textual and layout properties to perform text-label assignment. *HMM* explores another machine learning technique, Hidden Markov Models. It creates a 2-layered artificial designer having the ability to understand an interface based on the layout and textual properties of components. The first layer tags the components with semantic labels, and the second layer identifies the boundaries of segments. *ExQ* creates the interface structure tree using hierarchical agglomerative spatial clustering. Each form element is considered to be a visual attribute block. To generate the tree, spatially closer and similarly styled blocks are clustered under the same internal node. *ExQ* performs node label assignment using annotation rules and hence falls under a hybrid category.

3.4 Segment Processing

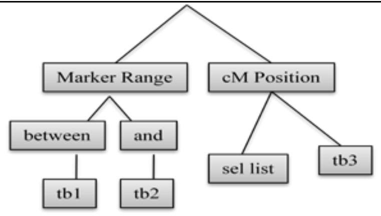
After an interface is segmented, more semantics related to segments and segment components are extracted. This includes information on data and integrity constraints of the underlying database. While several approaches enlist this information in the modeling stage, very few extract it. These approaches were studied under the following dimensions.

Technique: *LEX* uses machine learning classifiers to identify more semantics from a segment, such as type, domain type, value type, unit of form elements, relationship and semantics of domain elements, and logic relationship of attributes. *FormModel* uses another machine learning technique, learning by example, to extract relationship between two "structural units," and constraints of a form instance.

Post-processing: *LITE* and *LEX* post-process the text-labels by removing stop words such as "the," "any," etc. *LITE* also performs standard IR-style stemming on the text-labels. *HSP*'s "Merger" module reports conflicting tokens that occur in more than one query conditions, and missing tokens that they do not occur in any query condition. *LabelEx* devises heuristics for reconciliation of multiple labels assigned to an element and for handling form elements with unassigned labels.

Table 1 gives a summarized view of reductionist analysis showing the outputs generated by each work as a result of understanding the interface in Figure 4.

Table 1. Summary of Reductionist Analysis

	Modeling	*Parsing*	*Segmentation & Processing*	*Semantic Information*
CombMatch, LITE LabelEx DEQUE	Logical Attribute = <text-label, form element(s)>	Chunk Partitioning (CombMatch), Pruning (LITE).	tb1 => between, tb2 => and, select list => cM Position: , tb3 => cM Position:	4 label assignments
HSP	Pattern= <attr-name, operator, value>	Token positions.	<between, tb1>, <and, tb2>, <cM Position, sel list, tb3>	3 query conditions
LEX	Logical Attribute = <attr-label, element label, domain/ constraint element, domain type, default value>	Interface expression: tt|tete|t|t|ee|t	{Attr-label = Marker Range, Ele-label = between, Domain element = tb1, Ele-label = and, Domain element = tb2} {Attr-label = cM Position, Const. element = selection list, Domain element = tb3}	2 logical attributes
HMM	Segment= <attr-name (s), operator(s), operantor(s), misc-text(s)>	Pre-order DOM traversal: Marker…, use…, between, tb1, and, tb2, e.g. between, cM Position, sel list, tb3, e.g. "10.0-40.0"	{Attr-name = Marker Range, operator = between, operand = tb1, operator = and, operand = tb2, Misc-texts = use current …, e.g., bet …} {Attr-name = cM Position, operator = selection list, Misc-texts = e.g., "10.0 – 40.0"}	2 segments
SchemaTree ExQ	Tree Node = text-label or form element.	Text tokens: Marker Range:, use current, between, … Field tokens: tb1, tb2,… & bound. boxes for all tokens) (*SchemaTree* only)	 Marker Range cM Position between and tb1 tb2 sel list tb3	1 tree

3.5 Evaluation

Although evaluation is not a part of the core SIU process, it acts as a significant after-stage in all surveyed approaches. Here, the extracted semantic information is evaluated by comparing with either the manually extracted information or a gold standard as in the cases of *SchemaTree*, *LabelEx*, and *ExQ*.

Test Domain: The surveyed approaches are tested on several domains. The most popular choices of researchers are automobile, airfare, books, movies and real estate, followed by car rental, hotel, music, and jobs. Some of the least tested domains include biology, database technology, electronics, games, health, medical, references and education, scientific publication, semiconductors, shopping, toys, and watches. We compiled a list of various datasets at http://cluster.ischool.drexel.edu:8080/ibiosearch/datasets.html.

Metrics: *LITE, HMM,* and *LEX* report the extraction accuracy, i.e., the number of correctly identified components (segments) over the total number of manually identified components (segments). *DEQUE* reports the label extraction accuracy and the domain value extraction accuracy. *CombMatch* reports the success percentage, i.e., the number of correctly identified text-labels over the total number of elements, and the failure percentage, i.e., the number of incorrectly identified text-labels over the total number of elements. *HSP* reports precision and recall. Precision is the number of correctly identified segments over the total number of identified segments. Recall is the number of correctly identified segments over the total number of manually identified segments. *LabelEx* also reports recall, precision, and F-measure. *SchemaTree* measures text-label assignment accuracy, and the overall precision, recall and F-score. *ExQ* measures precision and recall for grouping, ordering, and node labeling.

Comparison of performance: Most of the surveyed works evaluate the performance by comparing their results with those of one or more of the contemporary works. *HSP* and *LEX* are the most widely used benchmarks for evaluation of performance. *HSP* was chosen by *LEX*, *LabelEx*, and *SchemaTree*, to compare the performances of respective works; and *LEX* was chosen by *LabelEx*, *SchemaTree*, and *HMM*. Another benchmark work is *CombMatch*, chosen by *LITE*.

4. HOLISTIC ANALYSIS

Section 3 viewed each work in the light of the stage-specific dimensions. It was found that certain dimensions, such as query and constraint information, segmentation task, and segmentation and segment processing technique, hold more potential for making significant changes in the overall process. These dimensions were used to create two composite dimensions for holistic analysis: database description and extraction technique. Based on this, the surveyed approaches can be plotted on a 2-D graph (See Figure 5) with the two axes corresponding to the two composite dimensions.

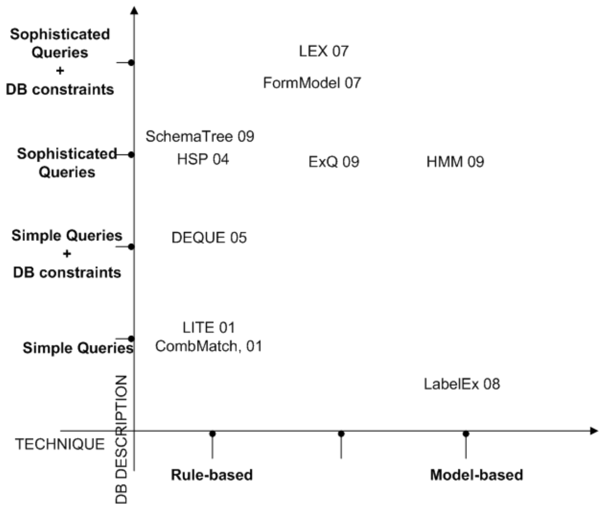
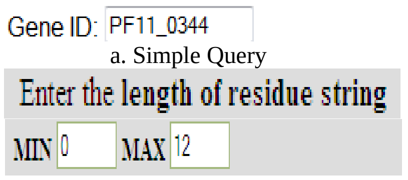


Figure 5. Holistic Analysis

Database Description: This dimension is described along the Y-axis and denotes the underlying database information extracted by a given approach. The surveyed approaches can be organized into 4 levels. The first level consists of *LITE*, *CombMatch*, and *LabelEx*. These works extract simple queries by performing text-label assignment. Figure 6a shows an example of a simple query extractible by associating “Gene ID:” with the adjoining textbox. This corresponds to the clause, “WHERE GeneID = ‘PF11_0344.’” However, text-label assignment at times results in extraction of partial query capabilities when it faces sophisticated designs like the one shown in Figure 6b. Such works might assign both textboxes to the text-label “Enter the length …,” but would fail to extract the complete implied query that corresponds to the clause, “WHERE length>=0 AND length <=12.” At the next level lies the work *DEQUE*. This approach extracts simple query capabilities along with data and integrity constraints of the underlying database.

The next level includes the works that extract sophisticated queries, like the one in Figure 6b, from an interface. *HSP*, *LEX*, and *HMM* identify such queries by grouping all related components into segments corresponding to logical attributes. *FormModel* forms a different type of segment that refers to an entity, “structural unit,” instead of an attribute. *SchemaTree* and *ExQ* are different too in that they perform hierarchical grouping and the queries extracted might be associated with both attributes and entities. Both *LEX* and *FormModel* employ strategies for extracting data and integrity constraints too, and thus, occur at the highest level.

Extraction Technique: This dimension refers to the techniques employed during the stages, segmentation and segment processing. These techniques fall under two categories: rules and models. We blend rules and heuristics into the rule-based category, and supervised and unsupervised machine learning into the model-based category. *HSP*, *LITE*, *CombMatch*, *DEQUE* and *SchemaTree* represent the rule-based approaches. *LabelEx* and *HMM* are both model-based. *LEX* and *FormModel* lie in between the two categories because they extract implied queries using rules, and extract constraint information using models. *ExQ* too lies in between as it performs grouping using a clustering model and performs text-label assignment using rules.


Gene ID: PF11_0344

a. Simple Query

Enter the length of residue string

MIN 0 MAX 12

b. Sophisticated Query

Figure 6. Types of Queries

Holistic analysis reveals two striking points regarding the journey of interface understanding in the past decade. First, a considerable progress has been made in terms of the underlying database information extracted. This is depicted by the transition from simple to sophisticated query capabilities across the Y-axis of the graph. However, the extracted information on data and integrity constraints does not appear to follow a regular timeline. Secondly, an improvement in the sophistication level of segmentation and segment-processing techniques, from rule-based to model-based techniques, is clearly visible along the X-axis.

5. OPEN QUESTIONS

The survey on the key search interface understanding approaches helped in identifying several unaddressed issues in this field. In terms of database description, we are very far from extracting the complete schema of the database that lies underneath an interface. In terms of the employed technique, previous studies [14, 15] have favored model-based over rule-based approaches for handling design heterogeneity. This is followed by a logical transition from rules to models. However, this transition did not have much effect on the degree of human intervention. While rule-based approaches, such as *HSP* and *LITE,* require manual specification of rules and human observations of heuristics, the model-based approaches, such as *LabelEx* and *HMM,* require manual annotation of the training data. *ExQ* made the first step toward unmediated understanding by employing a clustering technique to derive the initial tree structure for an interface. Such unsupervised learning techniques are much needed for developing scalable SIU solutions.

It should be noted that this survey focused on those approaches that attempt to understand an interface solely based on the information available on the interface itself. Interestingly, there is another alternative of deriving interface semantics, which is, by filling up the HTML forms using instances and analyzing the result pages. For example, [30] performs text-label assignment by "query probing," and [24] derives the domain of form element values using form submissions. Also, in the quest of surfacing, the work in [21] determines whether a form element is a "binding" or a "free" input, by generating the result pages. Another work [28] determines a list of possible "atomic queries" for an interface using form submissions. An "atomic query" is a minimal set of attributes that result in a valid result page. A combination of both interface-based and instance-based approaches of form understanding has not yet been explored. It should also be noted that certain works [23, 32] that perform SIU were not included in this survey. These works manually extract semantic information from interfaces and thus could not contribute much to the discussion of holistic and reductionist analysis.

Based on our findings in Section 3.5, a majority of the tested domains fall under the commercial Yahoo subject categories [9]. The other half of the Deep Web, containing databases from non-commercial domains [9] such as education, arts, science, reference, etc., has hardly been explored. Previous studies [13, 22, 33] have investigated the question of whether an SIU approach should be domain-specific or generic. Considering that a significant number of domains have remained unexplored and that the interface designs differ across subject domains, this question needs to be re-investigated on a balanced dataset of commercial and non-commercial domains.

Lastly, most of the SIU approaches have been designed for a specific application. While *HSP, LEX, LabelEx,* and *SchemaTree* target to increase the intra-domain usability of Deep Web contents, *DEQUE, LITE, LabelEx,* and *ExQ* target to increase content visibility on text search engines. Out of all, *SchemaTree* shines out as it has been cautiously designed to suit specific applications like interface matching and unification. This suggests the importance of aligning methodologies with intended applications. In future, a formal study of the correlation between the extraction methodologies and the potential application will be greatly beneficial.

6. REFERENCES

[1] Barbosa, L., Tandon, S., and Freire, J. 2007. Automatically constructing a directory of molecular biology databases. In Proc. of the International Workshop on Data Integration in the Life Sciences (Philadelphia, PA, Jun. 27-29, 2007) DILS' 07. Springer Berlin, Heidelberg. 6-16.

[2] Benslimane, S. M., Malki, M., Rahmouni, M. K., and Benslimane, D. 2007. Extracting personalised ontology from data-intensive web application: An HTML forms-based reverse engineering approach. Informatica, 18, 4 (Dec. 2007), 511-534.

[3] Bergman, M., K. 2001. The deep web: Surfacing hidden value. White Paper. University of Michigan.

[4] Cawsey, A. 1998. The essence of artificial intelligence Prentice Hall, Upper Saddle River, NJ.

[5] Chang, K. C., He, B., and Zhang, Z. 2005. Toward large scale integration: Building a MetaQuerier over databases on the web. In Proc. of the 2nd Conference on Innovative Data Systems Research (Asilomar, CA, Jan. 4-7, 2005) CIDR'05. ACM Press, New York, NY. 44-55.

[6] Dragut, E., C., Kabisch, T., Yu, C., and Leser, U., 2009. A Hierarchical Approach to Model Web Query Interfaces for Web Source Integration. In Proc. of the 35th International Conference on Very Large Data Bases (Lyon, France, August 24-28, 2009). VLDB'09. IEEE Computing Society, Washington, DC, 325 - 335.

[7] Halevy, A. Y. 2005. Why your data won't mix: Semantic heterogeneity. Queue, 3, 8(Oct. 2005), 50-58.

[8] He, B., and Chang, K. C. 2003. Statistical schema matching across web query interfaces. In Proc. of the ACM International Conference on Management of Data (San Diego, CA, June 9-12, 2003). SIGMOD'03. ACM Press, New York, NY. 217-228.

[9] He, B., Patel, M., Zhang, Z., and Chang, K. C. 2007. Accessing the deep web. Communications of the ACM, 50, 5 (Oct. 2008), 94-101.

[10] He, H., Meng, W., Lu, Y., Yu, C., and Wu, Z. 2007. Towards deeper understanding of the search interfaces of the deep web. World Wide Web, 10,2 (Jun. 2007), 133 - 155.

[11] He, H., Meng, W., Yu, C., and Wu, Z. 2004. Automatic integration of web search interfaces with WISE-integrator. The VLDB Journal the International Journal on very Large Data Bases, 13, 3(Sep. 2004), 256-273.

[12] Kaljuvee, O., Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. 2001. Efficient web form entry on PDAs. In Proc. of the 10th International Conference on World Wide Web (Hong Kong, China, May 1-5, 2001). WWW'01. ACM Press, New York, NY, 663 - 672.

[13] Khare, R., and An, Y. 2009. An Empirical Study On Using Hidden Markov Model for Search Interface Segmentation. In Proc. of the 18th International Conference on Information and Knowledge Management (Hong Kong, China, Nov 2-6, 2009). CIKM'09. ACM Press, New York, NY, 17 -26.

[14] Kushmerick, N. 2002. Finite-state approaches to web information extraction. 3rd Summer Convention on Information Extraction (Frascati, Italy, July 15-19, 2002) SCIE'02, Springer, Berlin, Heidelberg, 77-91.

[15] Kushmerick, N. 2003. Learning to invoke web forms. In On the move to meaningful internet systems. Springer Berlin, Heidelberg, 997-1013.

[16] Lage, J. P., da Silva, A. S., Golgher, P. B., and Laender, A. H. F. 2004. Automatic generation of agents for collecting hidden web pages for data extraction. Data and Knowledge Engineering, 49, 2(May 2004), 177 - 196.

[17] Lawrence, S., and Giles, C. L. 1998. Searching the World Wide Web. Science, 280, 5360(Apr. 1998), 98-100.

[18] Ling, Y., Meng, X., and Liu, W. 2008. An attributes correlation based approach for estimating size of web databases. Journal of Software, 19, 2(Mar/Apr. 2007), 224-236.

[19] Lu, J. (2008). Efficient estimation of the size of text deep web data source. In Proc. of the 17th ACM Conference on Information and Knowledge Management (Napa Valley, CA, Oct. 26-30, 2008) CIKM '08. ACM Press, New York, NY.

[20] Madhavan, J., Jeffery, S., R., Cohen, S. I., Dong, X., Ko, D., and Yu, C. 2007. Web-scale data integration: You can only afford to pay as you go. In Proceedings of Conference on Innovative Data Systems Research (Pacific Grove, CA, Jan. 7-10, 2007) CIDR'07. ACM Press, New York, NY. 40-48.

[21] Madhavan, J., Ko, D., Kot, L., Ganapathy, V., Rasmussen, A., and Halevy, A. Y. 2008. Google's deep web crawl. Proc. of the VLDB Endowment, 1, 2 (Aug. 2008), 1241-1252.

[22] Nguyen, H., Nguyen, T., and Freire, J. 2008. Learning to extract form labels. In Proceedings of the VLDB Endowment, Auckland, New Zealand. , 1, 1(Aug. 2008), 684-694.

[23] Pei, J., Hong, J., and Bell, D. 2006. A robust approach to schema matching over web query interfaces. In Proc. of the 22nd International Conference on Data Engineering Workshops (Atlanta, GA, April 3-7, 2006). ICDEW'06. IEEE Computer Society, Washington, DC. 46-55.

[24] Raghavan, S., and Garcia-Molina, H. 2001. Crawling the hidden web. In Proc. of the 27th International Conference on very Large Data Bases (Rome, Italy, September 11-14, 2001) VLDB '01, Morgan Kaufmann Publishers Inc, San Francisco, CA, 129-138.

[25] Ru, Y., and Horowitz, E. 2005. Indexing the invisible web: A survey. Online Information Review, 29, 3 (Apr. 2005), 249-265.

[26] Shestakov, D., Bhowmick, S., S., and Lim, E. 2005. DEQUE: Querying the deep web. Data and Knowledge Engineering, 52, 3 (Mar. 2005), 273-311.

[27] Shestakov, D., and Salakoski, T. 2007. On estimating the scale of national deep web. Database and expert systems applications, Springer Berlin, Heidelberg, 780-789.

[28] Shu, L., Meng, W., He, H., and Yu, C. 2007. Querying Capability Modeling and Construction of Deep Web Sources. In Proc. of 8th International Conference on Web Information Systems Engineering (Nancy, France, December 3-6, 2007) WISE '07. Springer Berlin, Heidelberg, 13-25.

[29] Wang, J., and Lochovsky, F. 2003. Data extraction and label assignment for web databases. In Proc. of 12th International Conference on World Wide Web (Budapest, Hungary, May 20-24, 2003) WWW '03. ACM Press, New York, NY, 187-196.

[30] Wang, J., Wen, J., Lochovsky, F., and Ma, W. 2004. Instance-based schema matching for web databases by domain-specific query probing. In Proc. of 30th International Conference on Very Large Data Bases (Toronto, Canada, August 29-30, 2004) VLDB '04, VLDB Endowment, 408 - 419.

[31] Wu, W., Doan, A., Yu, C., and Meng, W. 2009. Modeling and Extracting Deep-Web Query Interfaces. In Advances in Information and Intelligent Systems. Springer Berlin, Heidelberg, 65-90.

[32] Wu, W., Yu, C., Doan, A., and Meng, W. 2004. An interactive clustering-based approach to integrating source query interfaces on the deep web. In Proc. of the ACM International Conference on Management of Data (Paris, France, June 13-18, 2004) SIGMOD '04.ACM, New York, NY, 95 - 106.

[33] Zhang, Z., He, B., and Chang, K. C. 2004. Understanding web query interfaces: Best-effort parsing with hidden syntax. In Proc. of the ACM International Conference on Management of Data (Paris, France, June 13-18, 2004) SIGMOD '04.ACM, New York, NY, 107 – 118.