An Empirical Study on Using Hidden Markov Model for Search Interface Segmentation

Ritu Khare
The iSchool at Drexel
Drexel University
Philadelphia, PA, USA
ritu.khare@drexel.edu

Yuan An
The iSchool at Drexel
Drexel University
Philadelphia, PA, USA
yuan.an@ischool.drexel.edu

ABSTRACT
This paper describes a hidden Markov model (HMM) based approach to perform search interface segmentation. Automatic processing of an interface is a must to access the invisible contents of deep Web. This entails automatic segmentation, i.e., the task of grouping related components of an interface together. While it is easy for a human to discern the logical relationships among interface components, machine processing of an interface is difficult. In this paper, we propose an approach to segmentation that leverages the probabilistic nature of the interface design process. The design process involves choosing components based on the underlying database query requirements, and organizing them into suitable patterns. We simulate this process by creating an "artificial designer" in the form of a 2-layered HMM. The learned HMM acquires the implicit design knowledge required for segmentation. We empirically study the effectiveness of the approach across several representative domains of deep Web. In terms of segmentation accuracy, the HMM-based approach outperforms an existing state-of-the-art approach by at least 10% in most cases. Furthermore, our cross-domain investigation shows that a single HMM trained on data having varied and frequent design patterns can accurately segment interfaces from multiple domains.

Categories and Subject Descriptors
H.m **[Information Systems]:** Miscellaneous.

General Terms
Design, Experimentation, Performance.

Keywords
Design Pattern, Information Extraction, Search Interfaces, Segmentation.

1. THE PROBLEM STATEMENT
The deep Web consists of data that exist on the Web but are not returned by search engines through traditional crawling and indexing. The primary way to access these data is by filling up HTML forms on search interfaces. Deep Web is characterized by its growing scale, domain diversity, and a large number of structured databases [6]. Many solutions have been proposed to make the deep Web data more useful to users. These include designing intra-domain meta-search engines [21, 8, 15, 5, 20]; increasing content visibility on existing search engines [17, 12]; and deriving ontologies from search interfaces [1]. A pre-requisite to all these solutions is an automatic understanding of search interface semantics. This entails automatic *segmentation*, i.e., the task of grouping related components of an interface together. In this paper, we propose a *hidden Markov model* based approach to perform segmentation.

A search interface contains a sequence of interface components, such as text-labels and form elements (textboxes, selection lists, etc.). Interfaces are designed for human users to query the underlying databases. While it is easy for a human to discern the logical relationships among interface components, machine processing of an interface is difficult [4]. The key problem is to identify the boundaries for groups of logically related components. Following example illustrates the challenges involved in automatic segmentation.

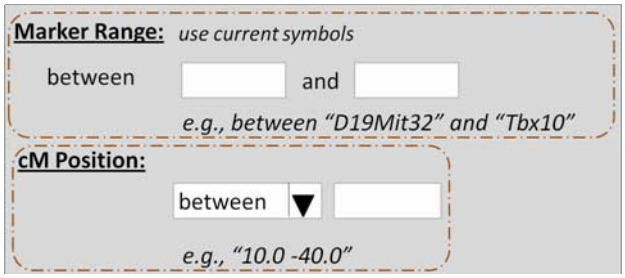

Figure 1. Segmented Search Interface

In Figure 1, consider the bottom segment having 4 components: "cM Position:," selection list, textbox, and 'e.g., "10.0-40.0".' From a user's standpoint, this segment has a very apparent semantic existence. By looking at the visual arrangement of components, and based on past experiences, the user creates a logical boundary around the 4 components as they appear to belong to the same atomic query. On the other hand, a machine is unable to "see" this segment because: (i) the components that are visually close to each other might be located very far apart in the machine-readable HTML code; (ii) a machine does not have the cognitive ability to identify a segment boundary. Motivated by this problem, in this work, we investigate whether a machine can "learn" how to segment an interface. We propose a new approach to perform segmentation. The novelty of our approach lies in the incorporation of the implicit knowledge used by a designer for designing an interface, and the use of this knowledge to perform segmentation. This is accomplished by learning an artificial designer using *hidden Markov model*.

CIKM'09, November 2–6, 2009, Hong Kong, China.

Previous solutions [17, 9, 22, 7, 13] to segmentation employed a variety of techniques including rules, heuristics, and machine learning. However, none of the solutions has explored the elegant probabilistic method, hidden Markov Model (HMM), for tagging and grouping interface components. HMM has been successfully applied to many applications including speech recognition [16], information extraction [2], and web text understanding [23]. A common feature of these applications is that there is a hidden process generating a stream of observable tokens. When it comes to tag and group the elements in the stream, reconstructing the hidden process provides an ultimate solution. In this work, we view a search interface as a stream of interface components. We believe that an HMM would be a nice way to complement previous efforts and provide an improved solution to the segmentation problem. We conduct a thorough study on the effectiveness of using HMM for interface segmentation.

To the best of our knowledge, this is the first work to apply *hidden Markov models* on deep Web search interfaces. Our first contribution is the segmentation performance achieved on using the proposed approach. Considering the subject diversity of deep Web [6], we experimented on interfaces from varied domains, and achieved more than 75% segmentation accuracy in most cases. In biology domain, our approach attained a segmentation accuracy of 87%, achieving 16% improvement over an existing approach, *LEX* [7]. In automobile domain, our approach attained an accuracy of 79%, achieving 24% improvement over *LEX*.

Our second contribution is that we assess the ability of a single model to segment interfaces from multiple domains. Previous related works [22, 13] compared the performance of *generic* (one model for all domains) methods with that of *domain-specific* (one model per domain) methods. While, a highly accurate generic model remains an open question, domain-specific models are not practicable owing to the growing domain diversity. We have a reconciling vision of designing a minimal set of learning models that reaches numerous deep Web domains. We compared the segmentation performance of models, trained in different ways, on a given domain. Initial results suggest that, if trained appropriately, a single model has the potential of accurately segmenting interfaces from multiple domains, while outperforming the domain-specific models of individual domains. The choice of the model depends on the frequency of *segment design patterns* in both the training and the test interfaces. A segment design pattern denotes the arrangement of components in a segment, e.g., the bottom segment in Figure 1 follows the pattern sequence: text, selection list, textbox, text.

The rest of this paper is organized in the following manner. Section 2 presents a related literature review. Section 3 describes hidden Markov models. Section 4 discusses an in-depth analysis of interfaces and our underlying principles of segmentation. Section 5 illustrates the 2-layered HMM approach. Section 6 presents the empirical study. Section 7 concludes the paper with contributions and future directions.

2. RELATED WORKS

The related literature is divided into two parts. Section 2.1 describes the existing approaches to search interface segmentation. Section 2.2 elaborates on the current status of the comparison between domain-specific and generic methods.

2.1 Interface Segmentation

The techniques employed for interface segmentation can be divided into two categories: rule/heuristic-based, and model-based. A large majority of existing works is based on rules and heuristics. Both rules and heuristics are crafted manually by observing design patterns on interfaces. Model-based approaches adopt a holistic approach and automatically learn the patterns into a model using training data. This learned model is then used for segmenting interfaces.

An example of a rule-based technique is the work by Zhang et al. [22]. This work assumes that a hidden syntax guides the presentation of segments (*query conditions)* on an interface. A *query condition* consists of the 3-tuple, {attribute, operator, values}. The tuples are identified using pre-specified grammar rules. An example of a heuristic-based technique is He et al. [7]'s *LEX*. The segment, *logical attribute,* is identified using a set of heuristics. Every HTML form element of an interface is associated with a surrounding text, and preference is given to the text that ends with a colon, matches with element's internal name, or is closer to the element. The form elements associated with the same text and the text itself belong to a *logical attribute*.

Certain works partially resolve the segmentation problem by focusing on the sub-problem, *label assignment*. Label assignment involves associating each form element with a surrounding text, e.g., associating the textbox with "Gene ID:" in the top segment of Figure 2, and associating both radio button group, and adjoining textbox with "Gene Name:" in the bottom segment. Both Raghavan and Garcia-Molina [17] and Kalijuvee et al. [9] perform label assignment using a combination of textual, styling, and layout properties of components.

With the growing diversity in interface design [6], it would be difficult for rule-based and heuristic-based techniques to cope with large interface collections. For every new interface, a new set of extraction rules would be required [10]. Nguyen et al. [13]'s *LabelEx* is a path-breaking model-based work based on supervised machine learning. It uses Naïve Bayes and Decision Trees classifiers to assign text labels to form elements based on textual and layout features of components.

In this paper, we explore another model-based technique, *hidden Markov models*. This helps in creating an artificial designer that has the ability to segment an interface. The proposed approach improves the process of manual rule-based and heuristic-based techniques in that it automatically learns the rules and heuristics into the model. It complements the model-based approach, *LabelEx*, in that along with label assignment, it also groups the related components together.

2.2 Domain-specific vs. Generic Methods

The deep Web has a diverse distribution of domains [6]. The interface designs differ across domains [13]. This is also supported by the results of our experiments on a large number of interfaces drawn from various domains. Previous works investigated whether a single method is sufficient for understanding all the domains by comparing domain-specific with generic methods. Consider an interface I belonging to a domain D_i. For I, a domain-specific method is designed by observing design patterns of interfaces belonging only to D_i. For the same interface, a generic method is designed by observing interfaces belonging to a mix of arbitrary domains, D_1, D_2, D_3 …

Zhang et al. [22] first investigated this arena. A rule-based grammar was designed and tested on different domains. It was concluded that a single generic grammar can generate a "reasonably good" segmentation performance (grouping & semantic tagging) for multiple domains. Later in 2008, Nguyen et al. [13] used model-based machine learning classifiers to measure the label assignment accuracy. These results too suggested that generic methods have the potential to perform reasonably well on multiple domains, and are only slightly less accurate than the domain-specific ones. The authors, however, recommend the design of domain-specific methods for applications in which accuracy is a critical factor.

We further this line of investigation in two ways. First, we design 3 kinds of segmentation methods: domain-specific, generic, and *cross-domain*. For an interface *I* belonging to a domain D_i, a *cross-domain* method is designed using interfaces belonging to another domain D_j. Secondly, we compare the performance of the 3 methods for the entire segmentation task, and not just for label assignment.

3. HIDDEN MARKOV MODEL

Hidden Markov Model (HMM) is a statistical machine learning technique, formally defined as a finite state automaton with stochastic state transitions and token emissions [16]. HMMs are based on Markov's assumption that the current state depends only on a finite history of previous states. In an HMM, states emit tokens based on a fixed state-specific distribution, and transitions among states occur based on a fixed distribution [11]. An HMM is used to model the process that produces a signal output, and to explain the signal output. Following are the main elements of an HMM:

- State Space (size N): A finite set of states $\{q_0, q_1, q_2 \ldots q_n\}$ denoted by an unobservable state variable q [18].
- Transition Model (A): The set of transitions is captured in a state transition matrix. Each cell contains the state transition probability P ($q \rightarrow q'$) of transitioning from a state q to q'.
- Prior Probability (π): The prior probability for a state q is the probability that q is the start state, i.e., the state at time t=0.
- Output Token Space (size M): an output token is denoted by an observable evidence variable σ [18]. The vocabulary of output tokens is a set $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$.
- Sensor Model (B): Output emission probability P ($q \uparrow \sigma$) denotes the state q's probability of emitting the token σ.

Rabiner [16] represents an HMM as $\lambda = (A, B, \pi)$ where A, B and π are the model parameters. $\sigma(t)$ denotes the token produced at time t, and q(t) denotes the state underwent by the model at time t. In this paper, we adopt two assumptions: (i) the HMM is of first-order, i.e., q(t) depends only on q(t-1);(ii) $\sigma(t)$ depends only on q(t). The most common problems handled by HMMs are parameter learning and state sequence prediction [18]. To learn the parameters, systematic algorithms, such as Baum Welch, Viterbi, and Maximum Likelihood are needed [2]. State sequence inference is the process of determination of the best state sequence, given an observation sequence [18]. Viterbi algorithm is the most accurate method to accomplish this [18].

4. SEARCH INTERFACE ANALYSIS

A Web search interface consists of a sequence of components that belong to different logical groups. Components in a single group have difference semantic roles. For example, in Figure 2, each component of the lower group, i.e, segment, is labeled with an underlined text, which we term as a *semantic label*. For a given component, the associated semantic label denotes the meaning of the component from user's or designer's standpoint. Web designers do not usually embed explicit component labels in the HTML source code. This makes automatic semantic labeling a difficult task.

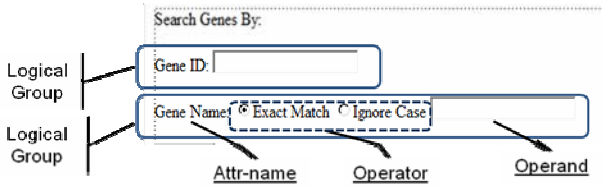

Figure 2. Assigning Semantic Labels to Components

In this study, we use a fixed set of 3 semantic labels, {*attribute-name, operator, operand*}. For data-driven Web applications, the user-entered values and the text-labels of a search interface are often translated into structured query expressions against the underlying databases. A typical structured query statement, i.e., SQL, contains 3 types of clauses: a clause indicating output (SELECT), a clause pointing to the database (FROM), and a clause specifying query conditions (WHERE). For example, assuming the underlying database table name is "Gene," following SQL queries might be generated for the interface in Figure 2:

1. SELECT * FROM Gene WHERE Gene_ID= 'PF11_0344';
2. SELECT * FROM Gene WHERE Gene_Name LIKE 'maggie';

For a typical Web application, although the underlying database name and schema are invisible, some clauses in a structured query are observable. In particular, a response page presenting query results corresponds to a SELECT clause, and a search interface collecting values for query conditions corresponds to a WHERE clause.

A WHERE clause consists of a set of predicates, e.g., Gene_ID = 'PF11_0344'. Such a predicate often specifies a query condition, using a built-in operator, for a particular attribute in the underlying database schema. Based on this observation, we use semantic labels, *attribute-name, operator,* and *operand,* for tagging interface components.

Before we proceed to present the proposed approach, we want to point out certain previous efforts that use a similar set of semantic labels. The work in [22] uses a very similar representation scheme {attribute name, operator, values}. We also share few similarities with *LEX* [7]; an operand is similar to the *domain element*, an operator is similar to the *constraint element*; and a text-based operator, such as "between" or "and" (Figure 1), is similar to the *element label*. Our work differs from the previous ones in that we view the interface design process as a stochastic process for laying out a stream of components with different semantic roles. When an HMM is learned to simulate the process, it can be used to automatically tag the observable components, as opposed to manually crafted and constantly changing rules or heuristics.

5. HMM: THE ARTIFICIAL DESIGNER

HMMs are used to model processes that are probabilistic in nature. Interface design process represents one such case. This

process involves statistically choosing an element from the universal set of components, and placing it on the interface based on its semantic label (see Figure 3). This organizes the components into certain design patterns. While a component is observable, its semantic label appears hidden to a machine. The sequencing of semantic labels is similar to the transitioning of HMM states. In Figure 4, oval represents a state, and rectangle represents an emitted token. Attribute-name "emits" text, operator "emits" a radio button group; and operand "emits" a textbox.

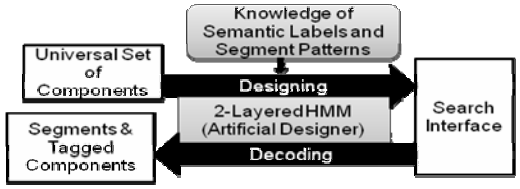

Figure 3. Simulation of Human Designer

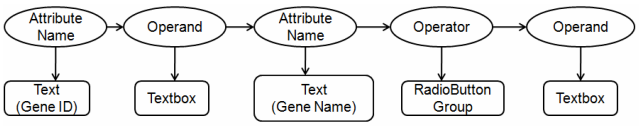

Figure 4. Decoding Figure 2's Interface

We simulate this design process by creating an "artificial designer" in form of a 2-layered HMM. The HMM learns the designing ability using suitable algorithms and data. The learned HMM acquires implicit design knowledge required for the decoding process, i.e., for determining semantic labels and segment boundaries. In the context of segmentation, the problem of decoding is two-folded involving semantic labeling as well as grouping of components. Hence, we employ a layered HMM [14] with 2 layers: the layer *T-HMM* (T stands for Tagging) tags each component with a semantic label (attribute-name, operator, or operand); the layer *S-HMM* (S stands for Segmentation) creates groups of related components. A single layer HMM is another design alternative to this approach. We choose the 2-layered HMM as it offers conceptual clarity and simplicity in preparing training data. The following subsections illustrate the procedure of designing the 2-layered HMM for interface segmentation.

5.1 Model Specification

We compiled a dataset, *spec_dataset,* containing 50 interfaces from 4 data sources (Table 7) to create model specifications. There are two different ways of sequencing the 2 layers: <T-HMM, S-HMM> and <S-HMM, T-HMM>. Experiments on *spec_dataset* suggest that the sequence <T-HMM, S-HMM> accomplishes higher accuracy than the sequence <S-HMM, T-HMM>. Thus, we position T-HMM before S-HMM. We use the superscripts, T and S, to differentiate the variables for T-HMM and S-HMM, respectively. Prior to training the 2-layered HMM, it is required to specify state space, token space, and model topology. Our adoptions of specifications are described next.

5.1.1 Layer 1: T-HMM

Table 1. State Space for T-HMM

Code	State Name	Semantic Role
$\mathbf{q_0}^T$	Operand	User- entered value matched against a database field
$\mathbf{q_1}^T$	Text-misc	Other text such as description, constraints, examples, etc.
$\mathbf{q_2}^T$	Attribute-name	Field name
$\mathbf{q_3}^T$	Operator	Search query operator

Choosing a suitable state space (size N^T , where $N^T=|q^T|$) was tricky. The time complexity of the employed state recovery algorithm is $O(L*N^2)$, where L is the length of observation sequence. We experimented with a varying number of states and decided on 4 states to manage this time complexity. Table 1 shows the states and their corresponding semantic roles. In many examples in *spec_dataset*, we found some texts that represent neither an attribute-name, nor an operator, e.g., the texts, 'use current symbols,' and 'e.g."10.0-40.0",' in Figure 1. We consider such miscellaneous texts to be emitted by a separate state, *text-misc,* denoted by q_1^T. The component emitted by this state may or may not belong to a segment. The state transition probabilities were automatically learnt from the *spec_dataset* using the Maximum Likelihood algorithm. Resulting topology is shown in Figure 5.

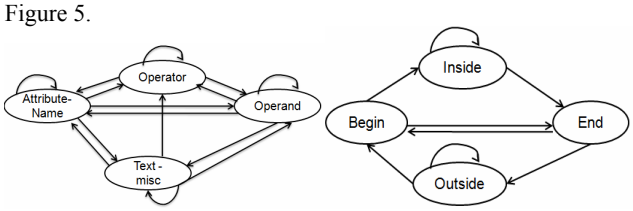

Figure 5. Topology (left: T-HMM, right: S-HMM)

Intuitively, the semantic labels would transition in the following order: <attribute-name, operator, operand>, i.e., $<q_2^T q_3^T q_0^T>$. The bottom segment of Figure 2 exemplifies this sequence. However, transitions in *spec_dataset* were counterintuitive in several ways. For instance, a self-transition from an attribute-name to itself, shown in Figure 5, might look questionable. In certain examples, a general attribute-name such as "ID," was followed by a specific one such as "Gene ID" as shown in the top segment of Figure 6, or an attribute-name was repeated as shown in the bottom segment of Figure 6.

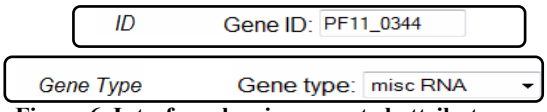

Figure 6. Interface showing repeated attribute-name

Table 2. Observation Token Space for T-HMM

Code	Observation Token
$\mathbf{\sigma_0}^T$	Any String (other than codes σ_1^T and σ_9^T)
$\mathbf{\sigma_1}^T$	String ending with colon
$\mathbf{\sigma_2}^T$	Textbox, test, text?
$\mathbf{\sigma_3}^T$	Textarea
$\mathbf{\sigma_4}^T$	File Input
$\mathbf{\sigma_5}^T$	CheckBox group
$\mathbf{\sigma_6}^T$	RadioButton group
$\mathbf{\sigma_7}^T$	Selection list
$\mathbf{\sigma_8}^T$	"between"
$\mathbf{\sigma_9}^T$	Parenthesized string
$\mathbf{\sigma_{10}}^T$	"min", "lower limit", "from", "from:", endsWith("from")
$\mathbf{\sigma_{11}}^T$	"max", "To:", "To", "less than or equal to", endsWith("less than equal to")

An HMM learns transition probabilities for different transitions using suitable training algorithms and data. If more training examples having the transition from an attribute-name to an operand, $q_2^T \rightarrow q_0^T$, are observed, a high values for $P(q_2^T \rightarrow q_0^T)$,

such as 0.7, is learnt. If fewer examples have this transition, a low value, such as 0.15 is learnt.

Table 2 shows the population of observation tokens (M^T=12, where $M^T=|\sigma^T|$). Token σ_1^T is added to the list because an attribute-name is likely to end with a colon; token σ_9^1 is added because a parenthesized string is not likely to be an attribute-name [7]. Codes σ_8^T, σ_{10}^T, and σ_{11}^T, are commonly used text-based operators found in *spec_dataset*. The styling and presentation components, such as <B>, <I>, <TR>, are ignored while parsing an HTML file.

5.1.2 Layer 2: S-HMM

Semantic labeling by T-HMM solves only half the segmentation problem. In *spec_dataset,* it was found that a segment does not have a standard structure. It does not always begin with an attribute-name (q_2^T), and end with an operand (q_0^T). The length of a segment is also not fixed. A segment can take varied patterns, such as $<q_2^T q_3^T q_0^T>$, $<q_1^T q_2^T q_2^T q_0^T q_0^T q_3^T q_1^T>$, or $<q_3^T q_3^T q_2^T q_1^T q_0^T>$. Therefore, identifying the boundaries of segments also requires learned knowledge. S-HMM completes the segmentation task by further tagging the T-HMM tagged components with respect to their sequential position in a segment. The state space for S-HMM is shown in Table 3. The observation token space for this layer is same as the T-HMM state space shown in Table 1. The topology is shown in Figure 5.

Table 3. State Space for S-HMM

Code	State Name	Semantic Role
q_0^S	Begin	begins a segment
q_1^S	End	ends a segment
q_2^S	Inside	lies inside a segment, other than q_0^S & q_2^S
q_3^S	Outside	lies outside a segment

Table 4 summarizes the tags assigned by different layers to the interface shown in Figure 2. A machine would parse the interface as a sequence of raw components, i.e., colonized string, colonized string, textbox, and so on. T-HMM would tag this sequence as text-misc, attribute-name, operand, and so on. S-HMM would further tag this as outside, begin, end, and so on.

Table 4. Tags assigned to Figure 2 Interface

Agent	Tagged Sequence	Description
Machine	$\sigma_1^T \sigma_1^T \sigma_2^T \sigma_1^T \sigma_6^T \sigma_2^T$	Components
T-HMM	$q_1^T q_2^T q_0^T q_2^T q_3^T q_0^T$	Semantic Labels
S-HMM	$q_3^S q_0^S q_1^S q_0^S q_2^S q_1^S$	Segment Positions

5.2 Learning HMMs

After the model specifications for both layers are decided, next step is to learn the model parameters, i.e., transition probabilities A^T and A^S, emission probabilities B^T and B^S, and prior probabilities π^T and π^S. We collected several hundred search interface examples from a variety of domains. We turn these examples into training data in order to learn the model parameters. Figure 7 describes the key components in our 2-layered HMM approach. To test the effectiveness of learned HMM for interface segmentation, we conduct cross-validations among the collected interfaces. We report the results of the empirical study in next section.

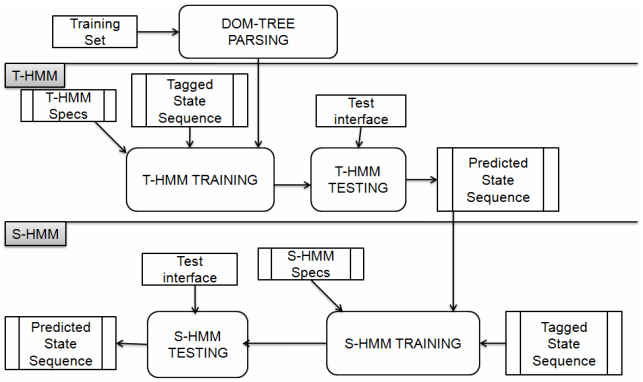

Figure 7. 2-Layered HMM Approach

6. EMPIRICAL STUDY

The goals of this paper are to study a more generic and accurate method for search interface segmentation and to investigate the factors affecting its performance. Theoretical analysis on designing interfaces suggests that HMM would be an effective technique to realize these goals. To verify this, we conduct a comprehensive empirical study. The study consists of two parts. In the first part, we measure the segmentation accuracy achieved on using HMMs. We compare the HMM based approach to a previous heuristic-based solution, *LEX* [7]. We choose *LEX* because the details of the heuristic-based algorithm are available in the literature and it explicitly studies the segmentation problem. Moreover, both *LEX* and our approach take HTML files as the primary input. This makes the two approaches comparable. There are other model-based solutions such as *LabelEx* [13]. However, in addition to the HTML file, *LabelEx* also takes the pixel positions on Web browser's window as an input. We thus consider *LabelEx* as a complement to our approach.

In the second part of the study, we compare the performance of HMMs, trained in different ways, on a given domain. We investigate whether a single HMM could demonstrate a high-quality performance across multiple domains. This would help in drawing some conclusions on how to design an HMM that caters to a given domain, as well as to multiple subject domains.

Based on the results, the proposed approach delivered a promising performance on interfaces from various domains. Biology domain was observed to have the most varied patterned search interfaces among all tested domains. The HMM, when trained on interfaces from biology domain, significantly outperformed *LEX* by 5% to 18%. The study concluded that the HMM, trained on a domain with most varied and frequent design patterns, returns a high segmentation accuracy in multiple domains, and even outperforms the domain-specific models from individual domains. This provides useful insights in designing model-based segmentation methods. In the following subsections, we report the details of our study.

6.1 Accuracy and Comparison

For preliminary evaluation, we collected 200 examples from biology domain (Table 7). A portion of an example interface described in its original HTML code is shown below.

<I><STRONG>Neighbors or Motifs file:</STRONG></I>
<P><SELECT NAME="file"><OPTION SELECTED
value="gbnr_neighbors">Genbank Non-redundant Protein DB

<OPTION value="dbEST_neighbors">dbEST Summary
</SELECT></P>
*<p>You may choose to place a lower limit on what you consider to be a valid neighbor:
"None" indicates 1e-3
*
<select name="p_value"><option selected>None<option>1e-50
</select>
<P><HR ALIGN=LEFT>Enter Query...The search is case insensitive!
<A HREF="http://www.cbil.upenn.edu/ParaDBs/syntax.html">Help with syntax</A></P><P><TEXTAREA NAME="search" ROWS="4" COLS="60"></TEXTAREA></P>

Parsing: For each interface, the source code was parsed into a DOM-tree [3] of components. These trees were traversed in the depth-first search order as the components belonging to the same segment are likely to be close to each other on the depth-first traversal path. They were then translated into corresponding observation sequences, e.g., the above example would be parsed as "$\sigma_1{}^T\sigma_7{}^T\sigma_1{}^T\sigma_0{}^T\sigma_7{}^T\sigma_0{}^T\sigma_0{}^T\sigma_3{}^T$."

Training and Testing Sets: The examples were trained and tested using 20-fold cross-validation method. The training examples were first tagged with the corresponding state information (Table 1 for T-HMM, Table 3 for S-HMM). In both layers, training and testing were performed using Maximum Likelihood and Viterbi algorithms, respectively. For evaluation, the predicted state sequence was compared with the correctly tagged sequence. Table 5 shows a summary of the results.

Table 5. Segmentation and Tagging Accuracy

S. No.	Semantic label	Accuracy (%)
1	Segment	86.05
2	Operator	85.10
3	Operand	98.60
4	Attribute-name-strict	90.11
5	Attribute-name-flexible	99.75
6	Text-misc	62.94

S-HMM Accuracy: The segmentation accuracy is listed in the 1st row of Table 5. Out of the total number of segments present in all the interfaces, 86.05% were correctly identified. A misidentification occurred due to one or more misidentified semantic label(s) by T-HMM. *Over-segmentation* was considered a success. For instance, in Figure 2, the top segment has 2 components and the component "Search Genes By:" occurs outside this segment. If a method includes the component "Search Genes By:" in the top segment, this is considered a case of over-segmentation. We consider this as success since the additional component is not a part of any other segment.

T-HMM Accuracy: This was calculated only for the segments correctly identified by S-HMM. The tagging accuracies for the operators and operands are shown in rows 2 and 3 of Table 5, respectively. Overall, 85.1% of the operands were correctly identified. A misidentification occurred because an operator state was mistaken as one of the other 3 states of T-HMM. Overall, 98.6% of the operands were correctly identified. In all the incorrect cases, the operand state was misidentified as an operator state. Overall, 90.11% of attribute-names were correctly identified (row 4). In all the incorrect cases, they were misidentified as text-misc. Attribute-name extraction was challenging as many segments tend to have multiple instances of attribute-name in the form of synonyms, hyponyms, etc. Therefore, we created another metrics, attribute-name-flexible accuracy. To calculate this, every segment with at least one correctly identified attribute-name instance was considered a success. Our approach, thus, achieved an accuracy of 99.75% (row 5). There are many existing approaches to handle multiple attribute-names. For instance, the work in [13] provides a term frequency based mechanism, *Mapping Reconciliation,* to choose a single text-label when multiple labels seem to be correct. The accuracy of the text-misc identification (row 6) is low as several instances of text-misc states were misidentified to be attribute-names. In future, we intend to improve this accuracy.

Comparison with LEX: We implemented the heuristic-based *LEX* algorithm based on its description in [7]. We used simple string similarity function to measure textual similarity, and weighed all heuristics equally. The 2nd column of Table 6 shows the accuracy attained by *LEX* on interfaces from 4 domains collected from sources shown in Table 7. The 3rd column of Table 6 shows the improvement in accuracy attained on using the proposed approach. The 4th column shows the improvement attained on using the "smartest" variation (HMM^{bio}) of the approach, described in Section 6.2.

Table 6. Comparison of Segmentation Accuracy (%)

Domain	LEX	2-layered HMM	2-layered HMM^{bio}
Biology	70.94	+16.66	+16.66
Health	66.85	+5.39	+13.74
Automobile	54.34	+24.66	+18.01
Movie	70	+0	+5.9

The proposed approach helps accomplish up to 24% improvement in accuracy over *LEX*. *LEX* does not model miscellaneous texts such as 'e.g., "10.0 – 40.0",' or "Enter up to two decimal places." It thus suffered from *under-segmentation* in a large number of cases. A heuristic in *LEX* considers only those texts as attribute-names that are located within 2-top-row distance from the form element. This led *LEX* to miss instances where attribute-name and operand were located far apart in the source code. This was the primary reason of a poor performance in automobile domain; interfaces in this domain had numerous row delimiters (<P>,
, </TR>). Also, in many cases, *LEX* misidentified a text operator (referred to as *element label* by *LEX*), such as "MIN" and "MAX" in Figure 10, as the attribute-name. In these cases, the text operator occurred closer to the form element and had an HTML name similar to the operator's text, and thus, weighed more than the correct attribute-name.

6.2 HMM Variations

In this section, we report the performance of the 2-layered HMM approach on altering the models. An advantage of using machine learning techniques is that a new model can be designed by merely replacing the training data. We tested the method on interfaces belonging to 5 test domains, using 3 variations of HMM: domain-specific, generic, and cross-domain.

For a test domain D, a domain specific model is a model trained on the interfaces belonging to the domain D. A cross-domain model with respect to D is a model trained on interfaces drawn from a domain other than D. A generic model is a model trained on interfaces drawn from a mixed set of domains.

Table 7. Sources of Datasets

Domain	Data Source	Size
Movie	Tel-8[1], Completeplanet[2], Beaucoup[3]	100
Health	Beaucoup	100
Ref. & Edu.	Beaucoup, Completeplanet	100
Automobiles	Tel-8, Completeplanet	100
Biology	NAR[4]	100
Mixed	All the above	100

1: http://metaquerier.cs.uiuc.edu/repository/datasets/tel-8/
2: http://www.completeplanet.com/
3: http://www.beaucoup.com/
4: http://www3.oup.co.uk/nar/database/c/

The experiments were carried out on 5 representative domains (Table 7). Interfaces from both commerce (movie, automobile) and non-commerce (biology, health, reference & education) categories were selected considering the balanced domain distribution of deep Web [6]. The "mixed" dataset was prepared by mixing 20% of interfaces from each domain.

Table 8. S-HMM Results (%)

Test Domain	*Training Data*					
	HMM *auto*	HMM *bio*	HMM *health*	HMM *movie*	HMM *ref_edu*	HMM *mixed*
Auto	**79**	72.35	73.63	68.81	*67.52*	70.7
Bio	48.7	**87.6**	48.72	*45.29*	52.56	51.2
Health	70.35	**80.59**	72.24	*69*	74.12	73.05
Movie	72.96	**75.9**	73.33	*70*	74.81	74
Ref. & Edu.	44.44	**62.3**	43.25	*38.88*	51	44

Table 8 shows the results. Each row corresponds to a test domain. Each column represents a version of HMM. Interfaces from each domain were segmented using 6 different models: 1 domain-specific, 4 cross-domain, and 1 generic. The models are different from each other as shown in Figure 8, where the T-HMM state transition topology and probabilities for all the models are different. The probabilities, $P(q_i^T \rightarrow q_j^T)$, were calculated using the Maximum Likelihood algorithm. Very low transition probabilities (<=0.05) are omitted. Due to limited space, we do not list the learnt values for emission probabilities of T-HMM, $P(q^T \uparrow \sigma^T)$, and emission and transition probabilities for S-HMM, $P(q^S \rightarrow q^S)$, and $P(q^S \uparrow q^T)$.

Overall, 30 experiments were conducted. Each domain-specific experiment used 10-fold cross-validation across 100 examples. For the rest, testing and training data sizes were 100 each. In Table 8, the numbers in bold represent the result by the winner model, i.e., the HMM that delivered the best performance in a domain. The italicized numbers represent the weakest performance in a domain. Next, we analyze the results and discuss implications.

6.3 Analysis

The results in Table 8 are counter-intuitive and suggest that domain-specific models do not always result in higher accuracy than generic or cross-domain models. In Sections 6.3.1 through 6.3.5, we analyze the results for each test domain. In Section 6.3.6, we present a summary of the analysis.

Hereafter, we denote a model trained on interfaces belonging to a domain *D* by HMM^D. For a test domain *D*, we represent a domain-specific model by HMM^D; a generic model by HMM^M,

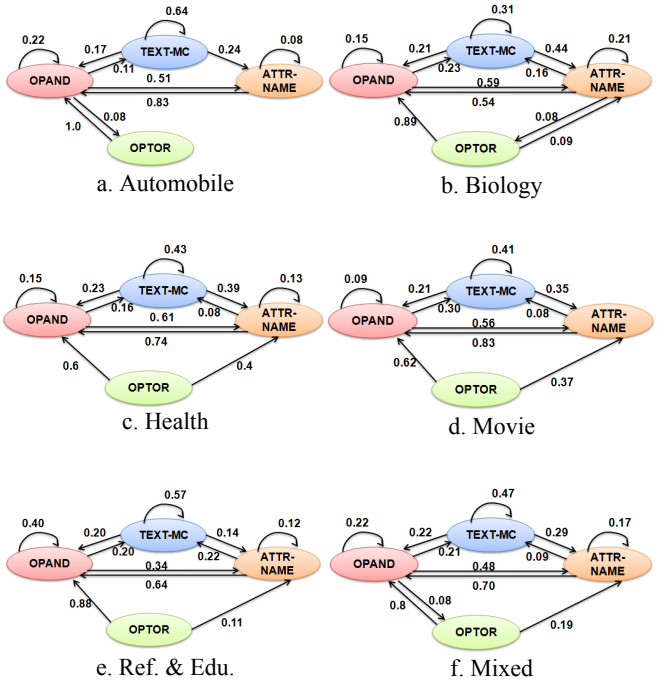

Figure 8. T-HMM Topology
OPAND=operand, OPTOR=operator, TEXT-MC= text misc, ATTR-NAME=attribute-name

where M is a mixed set of domains; and a cross-domain model by HMM^B, where *B* is a domain other than *D*. For segmenting interfaces from automobile domain, examples of domain-specific, generic, and cross-domain models are HMM^{auto}, HMM^{mixed}, and HMM^{bio}, respectively.

6.3.1 *Automobile Domain*

In automobile domain, the domain-specific model, HMM^{auto}, resulted in the highest accuracy (79%). For all other domains, HMM^{bio} delivered the best performance. To find the peculiarities in automobile domain, we made a note of the specific cases, correctly segmented by HMM^{auto} and misidentified by HMM^{bio}.

Figure 9 shows a frequent segment pattern <attribute-name, operand, operator, operand>, rewritten as $<q_2^T q_0^T q_3^T q_0^T>$ using Table 1 codes, found in automobile domain. In biology domain, it is not common to find this pattern, and operators occur in the settings $<q_2^T \mathbf{q_3^T} q_0^T \mathbf{q_3^T} q_0^T>$ and $<q_2^T \mathbf{q_3^T} q_0^T>$ as shown in Figures 10 and 2, respectively. Hence, HMM^{bio} was unable to identify the pattern as a complete segment, and instead decomposes the pattern into two segments. This difference is also apparent in the learnt values of transition and emission probabilities. The probability of the sub-pattern, having first 3 components of Figure 3, emitting the token sequence "text, selection-list, text" is higher in automobile domain than that in biology domain. Using HMM notation, the probabilities of occurrence of the sequence of events, $<q_2^T \uparrow \sigma_0^T, q_2^T \rightarrow q_0^T, q_0^T \uparrow \sigma_7^T, q_0^T \rightarrow q_3^T, q_3^T \uparrow \sigma_{11}^T>$, is higher in automobile domain.

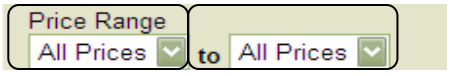

Figure 9. Under-segmentation by HMM^{bio} in auto domain

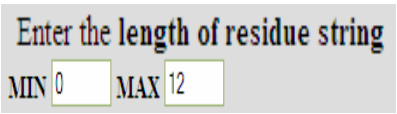

Figure 10. Typical Pattern in biology domain

The automobile domain has a limited set of patterns and very less outlier patterns. It thus resulted in higher accuracy on being trained with HMM^{auto}.

6.3.2 *Biology Domain*

In biology domain, the domain-specific model, HMM^{bio}, outshined the other models by a wide margin. Figure 11 presents a common pattern incorrectly tagged by the T-HMM layers of cross-domain and generic models. This resulted in an incorrect segmentation by respective S-HMM layers. While the tokens, textarea (σ_3^T) and file input (σ_4^T), are not found in other domains, they do occur in biology domain. The emission probabilities, P ($q_0^T \uparrow \sigma_3^T$) and P ($q_0^T \uparrow \sigma_4^T$) are 0.07, and 0.03, respectively, for $T\text{-}HMM^{bio}$ and are both 0.0 for other models. Thus, other models failed to recognize these segments.

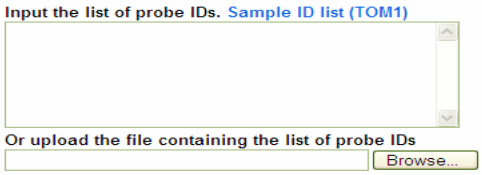

Figure 11. Common form elements in biology domain

Figure 10 shows yet another pattern, which was misidentified by all other models. In other domains, the transition from attribute-name ("Enter the length …") to operator ("MIN") is not very frequent. This is apparent in Figure 8, wherein $T\text{-}HMM^{bio}$ is the only model with a transition arrow from attribute-name to operator, i.e. with a $P(q_2^T \rightarrow q_3^T)$ value higher than 0.05. Biology domain contains many frequent segment patterns that are rarely found in other domains. Hence, no other model was able to keep up with its variety of patterns.

6.3.3 *Health Domain*

In health domain, the cross-domain model, HMM^{bio}, resulted in 8.35% higher accuracy than the domain-specific model, HMM^{health}. We noted the specific segments, misidentified by HMM^{health}, but correctly identified by HMM^{bio}. One such case is shown in Figure 12 where an operand (textbox) is followed by a text-misc ("(optional)"). As shown in Figures 8b and 8c, the value for this transition, $P(q_0^T \rightarrow q_1^T)$, in biology domain (0.23) is higher than that in health domain (0.16). Also, the probabilities of emission of a parenthesized text(σ_9^T) by a text-misc state (q_1^T), $P(q_1^T \uparrow \sigma_9^T)$, were found to be 0.34 and 0.13 in biology and health domain, respectively. In other words, constraints, in form of the text-misc state, occur after the operand state, more often in biology domain (Figure 1) than in health domain. Thus, HMM^{bio} is able to identify such pattern in the health domain.

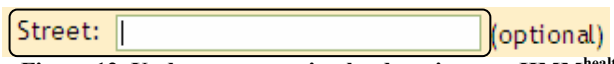

Figure 12. Under-segmentation by domain-spec. HMM^{health}

Another sub-pattern misidentified by $T\text{-}HMM^{health}$ is an operator state emitting a text token ("Dentists who …,") as shown in Figure 13. $S\text{-}HMM^{health}$ misidentified this operator as an attribute-name and retuned incomplete segment.

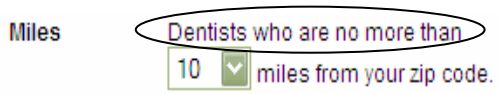

Figure 13. Rare text "operator" in health domain

6.3.4 *Movie Domain*

Similar to health domain, the cross-domain model, HMM^{bio}, resulted in 5.9% higher accuracy than the domain-specific model, HMM^{movie}. We noted the specific segments, misidentified by HMM^{movie}, and correctly identified by HMM^{bio}. Figures 14 and 15 show 2 such cases. In Figure 14, the inner shape is the one extracted using HMM^{movie}. In the movie domain, it is not very common to find a text-misc ("Hold down …") in between the states, attribute-name and operand. This is apparent in Figure 8, where the probabilities of transition from attribute-name to text-misc, $P(q_2^T \rightarrow q_1^T)$ are 0.16 and 0.08, in biology and movie domains, respectively. HMM^{ref_edu} was also able to capture this segment pattern as it has a relatively high value (0.22) for $P(q_2^T \rightarrow q_1^T)$.

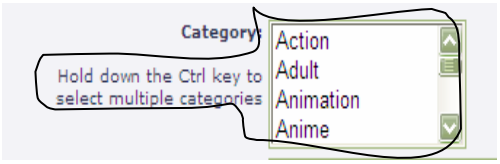

Figure 14. Under-segmentation by domain-spec. HMM^{movie}

Figure 15. Rare "operator" in movie domain

As shown in Figure 8d, there is no incoming transition to the operator state. This indicates the rarity of this state in movie interfaces. Figure 15 shows a pattern with an operator emitting a selection list, not identified by HMM^{movie}, but captured by HMM^{bio}. In general, the design patterns, which are not very common in movie domain but are common in biology domain, attribute to the 5.9% higher accuracy provided by HMM^{bio}.

6.3.5 *Reference & Education Domain*

In the reference and education domain too, HMM^{bio} resulted in higher accuracy than the domain-specific model, HMM^{ref_edu}. As compared to all other domains, the segmentation accuracy in this domain is low. We found this domain very different from others in terms of the frequency of segment patterns. This domain contains several rare patterns that are rare in other domains too.

Similar to the health and movie domains, the operator state is rare, as depicted in Figure 8e. A peculiar feature found in this domain is that it has multiple operands within a segment as shown in Figure 16. This can also be deduced from Figure 8e where the self-transition to operand state, P ($q_0^T \rightarrow q_0^T$), is the highest (0.4) among all the models. This is one case where the non-domain-specific models failed.

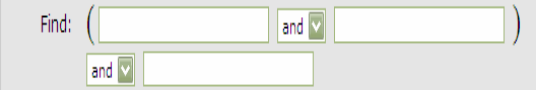

Figure 16. Segment with 3 Operands in Ref. & Edu domain

Figure 17 shows another rare pattern with an operand emitting the textarea token. T-HMM^{ref_edu} is not used to seeing a textarea as an operand state and thus, S-HMM^{ref_edu} did not acknowledge this pattern. This pattern was however captured by HMM^{bio}.

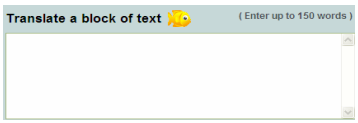

Figure 17. Textarea, a rare component in Ref. & Edu

Experiments in this domain also exposed some limitations of the 2-layered HMM approach. Consider Figure 18, the components in the HTML code do not occur in a sequential pattern. The segments are distributed in that the components, "Begin" and textbox located below, do not occur sequentially in the source code, and a component "End" that belongs to another segment lies in between. Currently, we are unable to represent such cases while preparing the training data. Figure 19 shows another rare pattern which we found difficult to represent.

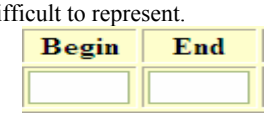

Figure 18. Distributed Segment in Ref. & Edu. domain

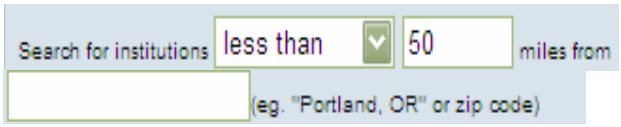

Figure 19. Complex Segment in Ref & Edu. domain

6.3.6 Implications

The analysis presented in previous sections brings up interesting points. Although the number of domains tested is limited; certain approximate generalizations can be made.

If a domain *D* has a peculiar and frequent pattern *P*, then *P* can be recovered by HMM^{D}. The example domains include biology and automobile. Figure 9 shows an example of one such pattern in automobile domain. The more the number of peculiar patterns in a domain, the more is the difference in accuracies between domain-specific and non-domain-specific models. This explains why the accuracy attained by HMM^{bio} on biology domain is much higher than that attained by any other model.

In *D,* if *P* is rare*,* then *P* can best be recovered using HMM^{B}, where *B* is a domain other than *D* that has *P* as a frequent pattern. In the domains, movie and health, most of the rare patterns (Figures 12, 14, and 15) are recovered by HMM^{bio}, generating 75.9% and 80.59% segmentation accuracies, respectively. The reference and education domain also has many rare patterns, which were rare in other domains too. Thus, most of the models were unable to identify those patterns.

Contrary to one's intuition, domain-specific models do not always lead to the highest possible accuracy. For instance, in movie domain, all non-domain-specific models resulted in higher accuracy than HMM^{movie}. It can, thus, be concluded that an artificial designer trained by more appropriate interfaces would lead to more accurate results. The appropriateness of a training dataset depends on the frequency of segment design patterns in the test domain as well as in the training dataset. In future, this might give directions in designing a minimal set of models, where each model returns higher accuracy than its domain-specific counterpart in individual domains, while reaching as many domains as possible. An example of such a model is HMM^{bio} that performed better than other models on 80% of the tested domains. The variety and frequency of patterns in biology domain helps the model contain more design knowledge, and thus, be a smarter designer.

7. CONTRIBUTIONS AND PROSPECTS

In this paper, we introduced the 2-layered HMM approach for automatic search interface segmentation. This approach is motivated by the probabilistic nature of interface design process. We encoded the implicit knowledge required for interface segmentation in an HMM-based artificial designer. This is the first work to apply HMMs on deep Web search interfaces. It improves and complements previous solutions in many aspects. Through this work, we have made the following contributions.

We tested the effectiveness of this approach across several representative domains of deep Web. The approach achieved high segmentation accuracy in most domains. We also compared the performance with a contemporary approach, *LEX* [7]. Our method outperformed *LEX* by at least 10% in most cases. We designed different kinds of learning models and compared their performances in each domain. It was deduced that: a single model has the potential of accurately segmenting interfaces from multiple domains, provided it is trained on the data having appropriate variety and frequency of design patterns. In future, we want to involve more domains, and derive a minimal set of models that reaches as many deep Web domains as possible.

The proposed approach can be expanded in several ways. One of our long term goals is to recover the schema of deep Web databases by extracting finer details, such as data types and constraints. Also, we intend to transition to a representation scheme that is able to address distributed and complex segments (Figures 18, 19). Using HMMs poses certain challenges too. Currently, the training data for both T-HMM and S-HMM is prepared by manual tagging. Our experience of tagging 600 examples suggests that the training samples can be constructed quickly and easily, as compared to the construction of exhaustive set of rules or heuristics. However, to minimize human intervention, we want to explore the use of unsupervised training methods such as Baum Welch algorithm. We also noticed the lengthy time taken by Viterbi algorithm for state recovery. We want to find optimization techniques to improve efficiency. Another alternative is to use the HMM-based method as an off-line pre-processing module to other applications such as meta-search engines and deep Web crawlers.

8. ACKNOWLEDGEMENTS

We sincerely thank the anonymous reviewers for providing valuable suggestions for revising this paper.

9. REFERENCES

[1] 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.

[2] Freitag, D. and Mccallum, A. K. 1999. Information extraction with HMMs and shrinkage. AAAI-99 Workshop on Machine Learning for Information Extraction (Orlando, Florida, July 18-19, 1999).31-36.

[3] Gupta, S., Kaiser, G. E., Grimm , P., Chiang, M. F., and Starren, J. 2005. Automating content extraction of HTML documents. World Wide Web, 8, 2(Jun. 2005), 179-224.

[4] Halevy, A. Y. 2005. Why your data won't mix: Semantic heterogeneity. Queue, 3, 8(Oct. 2005),50-58. DOI=http://doi.acm.org/10.1145/1103822.1103836.

[5] 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, California, June 9-12, 2003).SIGMOD '03. ACM Press, New York, NY, 217-228. DOI=http://doi.acm.org/10.1145/872757.872784.

[6] 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. DOI=http://doi.acm.org/10.1145/1230819.1241670.

[7] 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.

[8] 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.

[9] Kalijuvee, 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. DOI=http://doi.acm.org/10.1145/371920.372180.

[10] 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.

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

[12] 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. DOI= http://doi.acm.org/10.1145/1454159.1454163.

[13] 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. DOI= http://doi.acm.org/10.1145/1453856.1453931.

[14] Oliver, N., Garg, A., and Horvitz, E. 2004. Layered representations for learning and inferring office activity from multiple sensory channels. Computer Vision and Image Understanding, 96, 2(Nov. 2004), 163-180.

[15] 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, Georgia, April 3-7, 2006). ICDEW'06. IEEE Computer Society, Washington, DC, 46-55.

[16] Rabiner, L., R. 1990. A tutorial on hidden markov models and selected applications in speech recognition. Readings in Speech Recognition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 267–296.

[17] 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.

[18] Russell, S., J., and Norvig, P. 2002. Artificial intelligence: Modern approach, Prentice Hall, Upper Saddle River, NJ, USA.

[19] Seymore, K., Mccallum, A. K., and Rosenfeld, R. 1999. Learning hidden markov model structure for information extraction. AAAI 99 Workshop on Machine Learning for Information Extraction (Orlando, Florida, July 18-19, 1999). 37-42.

[20] 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.

[21] 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. DOI= http://doi.acm.org/10.1145/1007568.1007582.

[22] 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. DOI= http://doi.acm.org/10.1145/1007568.1007583.

[23] Zhong, P., & Chen, J. 2006. A generalized hidden markov model approach for web information extraction. In Proc. of ACM International Conference on Web Intelligence (Hong, Kong, China, Dec 18-22, 2006) WI '06, ACM, New York, NY. 709-718.