Learning to Discover Complex Mappings from Web Forms to Ontologies

Yuan An
College of Information Science and Technology
Drexel University
Philadelphia, USA
yan@ischool.drexel.edu

Xiaohua Hu
College of Information Science and Technology
Drexel University
Philadelphia, USA
thu@ischool.drexel.edu

Il-Yeol Song
College of Information Science and Technology
Drexel University
Philadelphia, USA
isong@ischool.drexel.edu

ABSTRACT

In order to realize the Semantic Web, various structures on the Web including Web forms need to be annotated with and mapped to domain ontologies. We present a machine learning-based automatic approach for discovering *complex* mappings from Web forms to ontologies. A complex mapping associates a set of semantically related elements on a form to a set of semantically related elements in an ontology. Existing schema mapping solutions mainly rely on integrity constraints to infer complex schema mappings. However, it is difficult to extract rich integrity constraints from forms. We show how machine learning techniques can be used to automatically discover complex mappings between Web forms and ontologies. The challenge is how to capture and learn the complicated knowledge encoded in existing complex mappings. We develop an initial solution that takes a naive Bayesian approach. We evaluated the performance of the solution on various domains. Our experimental results show that the solution returns the expected mappings as the top-1 results usually among several hundreds candidate mappings for more than 80% of the test cases. Furthermore, the expected mappings are always returned as the top-k results with $k \leq 4$. The experiments have demonstrated that the approach is effective and has the potential to save significant human efforts.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases—*Data Translation*

General Terms

Algorithms, Design, Experimentation

Keywords

Semantic Mapping, Web Forms, Ontologies, Mapping Discovery

1. INTRODUCTION

In order to realize the Semantic Web, various structures on the Web including Web forms need to be annotated with and mapped to domain ontologies. Mappings from Web forms to ontologies could be as simple as value correspondences between atomic elements or as complex as logic formulas associating *complex* structures in the two models. In most applications on the Semantic Web, complex mappings are needed, for example, automatically extracting, translating, and loading data from Web databases to a triple store. Until now, it has been assumed that humans specify these complex mapping formulas, a highly complex, time-consuming and error-prone task when the number of forms and the sizes of ontologies become large. In this paper, we propose an automatic tool that assists users in discovering complex mapping from Web forms to ontologies.

EXAMPLE 1.1. *Figure 1 on the left-hand side shows a Web form entitled Vehicle Search. The form is a query interface on the Web for searching new and used vehicles. As a user interface, the form consists of a collection of form elements organized in a particular way. For example, this form contains elements such as text labels, text boxes, radio buttons, and drop-down lists. Other elements that are often used on forms for user input include check boxes and calendar widgets.*

To extract the data from the underlying Web database and load them into a knowledge base or triple store described by an ontology, it is necessary to map the form to the ontology. Figure 1 on the right-hand side shows a portion of the Vehicle Sale Ontology[1] *(VSO). The ontology is described in the standard UML (Unified Model Language) notation. A relationship may have cardinality constraints enforced on the participating concepts. Here, we use a "*" to indicate unlimited cardinality constraints (many), and a number, for example, 1, for restricted constraints. The VSO ontology (namespace: vso) is an existing e-business ontology compliant with the GoodRelations ontology (namespace: gr) [14] on the Web.*

The dashed-arrows from the form elements to the ontology elements are **simple** *correspondences. For example, the 'Price:' label on the form corresponds to the gr:PriceSpecification concept in the ontology. A* **complex** *mapping is indicated by the implicit relationship between the semantic association among the labels 'Vehicle Search', 'Price:', and 'Make:' on the form and the thick-lined connection among the concepts in the ontology. In particular, the 'Price:' label together with 'Vehicle Search' on the form correspond to the path in the ontology from the concept node gr:Offering to the concept gr:PriceSpecification through* `--gr:hasPriceSpecification-->-`*, where we use the notation* `--r-->-` *to represent a many-to-one (functional) relationship r in an ontology, and* `--r--` *to represent a many-to-many relationship.*

The 'Make:' label together with 'Vehicle Search' on the form correspond to the ontology path from the concept node gr:Offering

[1] http://www.heppnetz.de/ontologies/vso/ns.owl

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

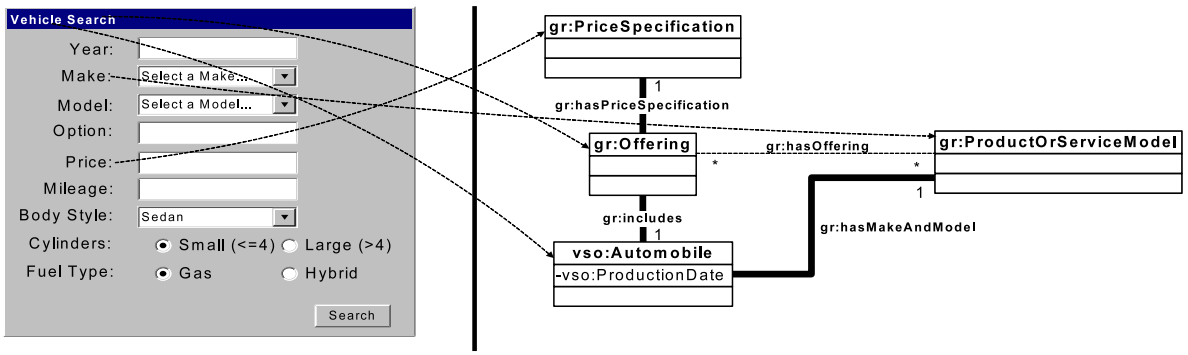

Figure 1: A Vehicle Search Form and the Simple and Complex Mappings to the Vehicle Sale Ontology (VSO)

to the concept node gr:ProductOrServiceModel through the relationship `--gr:includes-->-`*, the concept vso:Automobile, and the relationship* `--gr:hasMakeAndModel-->-`*. Notice that although there is a direct edge* `--gr:hasOffering--` *between the concepts gr:Offering and gr:ProductOrServiceModel, the relationship does not match the meaning of the concept represented by the 'Make:' label together with Vehicle Search on the form. Because the precise semantics specifies that each vehicle under search has only one make, while the direct edge is a many-to-many relationship.*

The thick-lined connection in the ontology gives rise to a complex structure that is mapped by the semantic association among the labels 'Vehicle Search', 'Price:', and 'Make:' on the form. In this paper, we aim to develop a solution for automatically discovering such complex mappings. ■

A variety of techniques have been proposed in the literature for discovering complex mappings between different data representations. For example, the solution for schema mapping (e.g., Clio [22, 26]) infers declarative logic expressions – *schema mappings*– for data exchange between two databases. The solution takes as input a source and target schema and a set of simple correspondences between schema elements. It then computes all the *meaningful* join paths in both schemas through a *chase* algorithm. In particular, the *chase* algorithm exploits the integrity constraints in schemas such as functional dependencies for computing meaningful join paths. At the end, each pair of meaningful join paths from the source and target schemas is presented to the user as a candidate schema mapping. The user needs to select the desired mapping from a list of candidate mappings.

Another related technique is the solution for discovering complex mappings from database schemas (either relational or XML) to ontologies (e.g., MAPONTO [2, 1]). The solution takes as input a database schema and an ontology, and infers a set of complex mappings representing the semantics of the schema in terms of the ontology. Once again, the solution takes as part of the input a set of simple correspondences between elements in the schema and elements in the ontology. It then analyzes the patterns of the constraints in the schema, and recovers a set of potential conceptual structures in the ontology. The underlying recovery principle is that the schema could be generated from the conceptual structures by the standard database design methodology.

There are several noticeable features shared by the existing solutions for discovering complex mappings. The first is that the solutions assume that the input includes a set of simple correspondences between elements in the input models. For example, in Figure 1, $\mathcal{F}$: Price:$\rightsquigarrow\mathcal{O}$: gr:PriceSpecification is such a simple correspondence, where, the symbol '$\rightsquigarrow$' represents a corresponding relationship, and we use the prefix $\mathcal{F}$ and $\mathcal{O}$ to distinguish terms in the form and ontology. There are techniques for *schema matching* (e.g., [27]) that are able to discover simple correspondences between *atomic* elements in a source and target model. The second feature is that the solutions return a set of candidate mappings for users to select. There is often no ranking among the candidates. The user has to sift through the entire list to find the expected one. Third, the core of the solutions is to capture and analyze some specific types of knowledge carried by the models being mapped, for example, integrity constraints or database design patterns. This requirement hinders the application of the existing solutions to the problem we study in this paper.

EXAMPLE 1.2. *Suppose we have discovered several simple correspondences between the elements on the form and in the ontology in Figure 1, for example,* $\mathcal{F}$*: Vehicle Search*$\rightsquigarrow\mathcal{O}$*: gr:Offering and* $\mathcal{F}$*: Make:*$\rightsquigarrow\mathcal{O}$*: gr:ProductOrServiceModel. Suppose we knew that there were an integrity constraint on the form specifying the uniqueness of the Make for Vehicle Search. Then we could link the ontology concepts gr:Offering and gr:ProductOrServiceModel through the functional (many-to-one) path consisting of the edges* `--gr:includes-->-` *and* `--gr:hasMakeAndModel-->-`*. Unfortunately, the extraction of such integrity constraints is difficult, if not impossible, from the form. Alternatively, a mapping discovery solution could use shortest paths to connect the ontology concepts. In this case, the correct functional path would be excluded as a candidate mapping.* ■

Here, we are faced with a specific knowledge discovery problem: Given a Web form and an ontology; suppose there is a solution to specify simple correspondences between elements on the form and elements in the ontology; the problem is to discover the most likely ontology connections that correspond to the semantics of the form which consists of a collection of user input elements. The challenge in the problem is that there is little information about integrity constraints among form elements that can be exploited. In addition, there is no standard design methodology for designing forms from conceptual models. To address the challenge and solve the problem, we resort to machine learning methods that can learn knowledge from examples.

Machine learning has been applied previously to the problem of discovering simple correspondences between atomic elements in different models, for example, schema matching [8, 9, 6] and form matching [11, 30, 25]. The learning problem was often characterized as a classification problem. For example, given a schema

element c in one schema and some instances associated with c, train different classifiers to classify c using the elements of another schema as labels. For the problem we study in this paper, we will make use of the techniques for discovering simple correspondences between form elements and ontology elements. However, we need to develop machine learning models to capture the complicated knowledge about semantic correspondences between complex structures among elements on the form and complex structures among elements in the ontology.

We consider the learning and discovery problem as a probabilistic inference problem among a set of potential ontology structures given a form structure. Specifically, for a form $\mathcal{F}$, we can apply a schema matching technique to discover simple correspondences from the elements on the form to elements in the ontology. With a set of identified elements in the ontology, we can discover a set $\mathcal{G}$ of potential structures/connections among the ontology elements. The learning task is to learn a probabilistic model h such that for each potential structure $G \in \mathcal{G}$, the probabilistic model computes the probability of G given $\mathcal{F}$, $P(G|\mathcal{F})$. Hopefully, the structure with the highest probability given $\mathcal{F}$ would be the expected result.

In the rest of this paper, we flesh out the idea outlined above and develop a naive Bayesian approach. To learn the knowledge encoded in complex structures such as ontologies, we are faced with two major challenges: The first is the need to collect large amounts of samples in order to cover every possible structure; the second is an astronomical number of joint probabilities that need to be estimated. To tackle the challenges, we adopt the strategy of the naive Bayes classifier, which has worked quite well in many complex real-world situations. The key assumption in the approach is that *the paths in the ontology are conditionally independent given the tree structure extracted from a form.* With the growth of the Semantic Web, there is an increasing number of forms annotated with ontologies. Consequently, we believe a sufficient number of training examples will be available for the learning task.

In summary, we make the following contributions in this paper:

- We explore the opportunity of applying machine learning techniques for discovering complex semantic mappings between different data representations. Our methods complement the existing solutions.
- We develop a naive Bayesian approach for discovering semantic mappings between forms and ontologies. The method greatly reduces the complexity of learning process. Its probabilistic feature naturally ranks candidate mappings.
- We evaluated the performance of the method on various domains. Our experimental results show that the method returns the expected mapping usually among several hundreds candidate mappings as the top-1 result for more than 80% of the test cases. Furthermore, the expected mappings are always returned as the top-k results with k≤4. The experiments demonstrate that the approach is effective and has the potential to save significant human efforts.

The remainder of this paper is organized as follows. Section 2 presents the formal preliminaries. Section 3 describes the machine learning task. Section 4 presents the proposed learning algorithm. Section 5 discusses the experimental results. Section 6 describes the related work. Finally, Section 7 concludes the paper.

2. PRELIMINARIES

The first step in discovering complex mappings between a Web form and an ontology is to *automatically* extract a formal structure from the form. The collection of elements on a form are often laid out in a particular way in accordance to the semantic relationships among corresponding real-world objects. A common pattern is that a relationship between two entities is represented by a hierarchical parent-child relationship on a form. Essentially, the organization of form elements is a hierarchical containment tree structure. Such a tree structure can be easily obtained by parsing the source code of a form into a tree representation such as a DOM tree.

The tree-like organization of form elements gives rise to a hierarchical structure for modeling a form. In particular, a form can be modeled as an ordered tree of elements called *form tree* which captures the semantic relationships among the real-world objects represented by the form. A leaf element in the tree corresponds to an input field on the interface. An internal element corresponds to a group of fields. Formally, we define a *form tree* as follows.

DEFINITION 2.1 (**Form Tree**). *A form tree is defined as a labeled, directed and ordered tree, $F_T = (N, E, <_{sib}, root)$, where $N = \mathcal{I} \cup \mathcal{E} \cup \mathcal{V}$ is a finite set of nodes, E is a finite set of edges, $<_{sib}$ is the next-sibling relation between children of a tree node, and $root \in N$ is the root of the tree. Moreover,*

- *$\mathcal{I}$ is a finite set of input fields (or inputs), where items of $\mathcal{I}$ are drawn from the following set of input fields: {text box, text area, radio buttons, check boxes, drop-down list, calendar} ;*
- *$\mathcal{E}$ is a finite set of logical elements. Each $e \in \mathcal{E}$ has a label $l = \lambda(e)$ and a data type $t = \tau(e)$, where the function $\lambda(e)$ returns the label l of e, and the function $\tau(n)$ returns the data type t of e.*
- *$\mathcal{V}$ is a finite set of values;*
- *For an edge $(n_i \longrightarrow n_j) \in E$, $n_i, n_j \in N$, n_i is called* **parent** *and n_j is called* **child**.

EXAMPLE 2.1. *On the top of Figure 2 shows the form tree extracted from the form Vehicle Search in Figure 1. Graphically, we use different shapes to represent different types of nodes. The root of the tree is a logical element Vehicle Search representing the entire form. The set of parallel text labels that are contained in the form are extracted as the children of the root. Each text label corresponds to a logical element on the next level in the tree. A text box next to a text label on the form is represented as an input element/field node in the tree which is a child of the logical element corresponding to the text label on the form. For example, the tb_price field node as a child of the logical element node Price in the tree corresponds to the text box next to the text label 'Price:' on the form. A value node is created for each value on the form. For example, the value 'Small' next to the radio button on the form is represented as a value node Small in the tree as a child of the input field node rb_cylinder corresponding to the radio button on the form.*

On the bottom of Figure 2 shows an expanded portion of the VSO ontology corresponding to the form tree on the top. The shaded concepts highlight the differences between the ontology portion in Figure 1 and the ontology portion in Figure 2. ■

Form Tree Extraction. Given a Web form, we want to automatically extract a form tree from the form. Form tree extraction is the process of clustering the elements on a form into semantically related groups, and organizing the groups into parent-child hierarchical relationships. In the previous work [17, 3], we developed a machine learning technique that leverages Hidden Markov Models

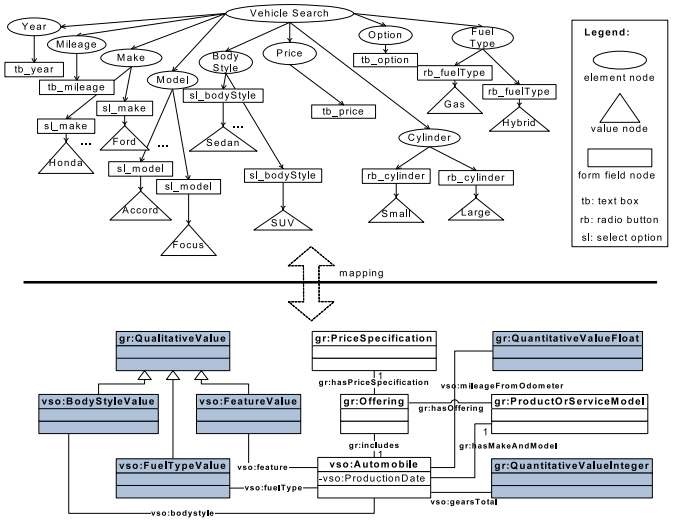

Figure 2: The Form Tree (top) for the Vehicle Search Form

(HMM) for automatically extracting a *form tree* from a form. In this paper, we will assume that a form has been converted to a form tree through the existing solution. Having a form tree, we focus on *discovering complex semantic mappings from the form tree to a given ontology.*

Ontology. An ontology is a conceptual model that describes an agreed-upon conceptualization of a domain. In this paper, we do not restrict ourselves to any particular language for describing ontologies. Instead, we use a generic conceptual modeling language (CML), which contains common aspects of most semantic data models, UML, ontology languages such as OWL, and description logics. Specifically, the language allows the representation of concepts, relationships, and attributes. Concepts are organized in the familiar is-a hierarchy. Relationships are subject to cardinality constraints, which here allow 1 as lower bounds (called total relationships), and 1 as upper bounds (called functional relationships). We use $\mathcal{O} = (C, R, A)$ to represent an ontology $\mathcal{O}$ with a set of concepts C, a set of relationships R, and a set of attributes A. An ontology can be represented as a graph whose nodes are concepts and attributes, and whose edges are relationships and links from concepts to attributes. For the sake of simplicity, we use UML notations, as in Figure 2 (bottom), to represent the ontology graph. Note that in such a diagram, instead of drawing separate attribute nodes, we place the attributes inside the concept nodes. Readers should not be confused by this compact representation. If r is a relationship between concepts C and D, we propose to write in text as C `--r--` D (If the relationship r is functional, we write C `--r-->-` D .)

Simple Correspondence. Given a form tree $F_T = (N, E, <_{sib}, root)$ and an ontology $\mathcal{O} = (C, R, A)$, a correspondence F_T:v $\rightsquigarrow$ $\mathcal{O}$:u relates a node v$\in N$ of the tree F_T to an element u$\in C \cup R \cup A$ in the ontology $\mathcal{O}$.

Complex Mapping. In general, a complex mapping relates a subtree of a form tree with a subgraph of an ontology graph. For the purpose of learning and discovering such a complex mapping, we decompose a complex mapping into a set of relationships each of which is between an edge in the form tree and a path in the ontology graph. In particular, a complex mapping is a set of relationships $\{F_T$:e $\xrightarrow{m}\mathcal{O}$:p$\}$, each of which relates an edge e$\in E$ of the tree to a path p$\in \mathcal{O}$ in the ontology graph. A path is a sequence of edges where two adjacent edges share an end node.

The Mapping Discovery Problem. We define the problem of discovering complex semantic mappings between a form tree and an existing ontology as follows:

- **Input:** an form tree $F_T = (N, E, <_{sib}, root)$ and an ontology $\mathcal{O} = (C, R, A)$
- **Output:** a set of expressions $\{F_T$:e $\xrightarrow{m}\mathcal{O}$:p$\}$
- **Goal:** to discover a subgraph in the ontology graph such that the subgraph captures the underlying meaning of the form tree. Each edge in the form tree is mapped to a path in the subgraph as F_T:e $\xrightarrow{m}\mathcal{O}$:p.

3. THE MACHINE LEARNING TASK

Given a form tree and an ontology, we now specify the machine learning task for discovering complex semantic mappings from the form tree to the ontology. Before proceeding, we pre-process form trees by removing input fields and value nodes. Specifically, a single text box under a text label is removed from the tree, because the text box is an input field which is used to collect the values of the object represented by the text label. Similarly, value nodes such as the values associated with radio buttons, check boxes, or select list are pre-processed as well. In particular, we can classify whether a value node corresponds to a concept or a value in the ontology. Value nodes corresponding to concepts are converted to element nodes, while values nodes corresponding to values are removed from the form tree.

EXAMPLE 3.1. *Figure 4 shows the form tree corresponding to that in Figure 2 after being pre-processed. In particular, all the leaf text boxes under text labels are removed. For example, the text box* `tb_year` *is removed from the form tree. All the value nodes along with their input field nodes are removed if the values correspond to some value instances in the ontology. For example, the 'Honda' value node along with the select field node* `sl_make` *is removed. However, the value nodes 'Small' and 'Large' are combined to a new element node Size under the element node Cylinder, since 'Small' and 'Large' correspond to the concept Size.* ■

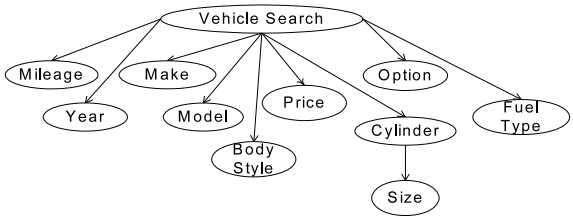

Figure 4: A Pre-Processed Form Tree

After pre-processing a form tree, we approach the problem of discovering complex semantic mapping from the form tree to an ontology in two steps as illustrated in Figure 3. First, we discover *simple correspondences* between the elements of the form tree and the elements in the ontology. Second, we derive semantic connections among the ontology elements that are identified by the form tree elements through the correspondences. Such a two-step paradigm has been employed by various existing solutions for

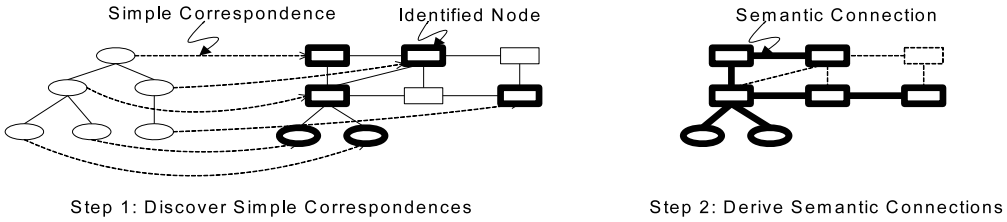

Figure 3: The Two-Step Paradigm for Mapping Discovery

schema and ontology mapping problems. Example solutions include TranSem [23], Clio [22, 26], HePToX [5], MQG [16], and MAPONTO [2]. As indicated before, we employ existing schema matching techniques (e.g., [27]) to discover candidate simple correspondences with ranked similarity values. In this paper, we then focus on the second step to *infer the most likely connections among ontology elements.*

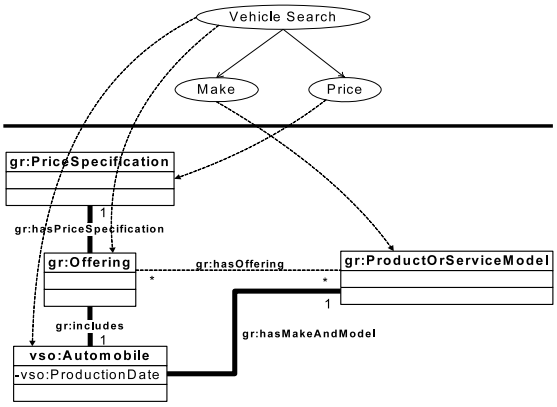

Figure 5: Simple Correspondences between a Form Tree and an Ontology

EXAMPLE 3.2. *Figure 5 shows a portion of a form tree and a portion of an ontology as well as the discovered simple correspondences (dashed arrows) linking them. Given the four concept nodes in the ontology identified by the simple correspondences, we want to discover the most likely connections among these nodes given the form tree. Figure 6 shows two candidate connections. If we assign equal weights to the edges of the ontology graph, both candidate connections are minimum spanning trees among the nodes.* ■

We aim to develop a machine learning method to infer the most likely connections in an ontology given a form tree. In general, we can define the learning task as follows:
Given:

- A set of form trees $\{F_T^i = (N^i, E^i, <_{sib}^i, root^i)\}$ and an ontology $\mathcal{O} = (C, R, A)$;
- A set of training examples $\{M^i\}$, where $M^i = \{F_T^i : \mathsf{e}^i \xrightarrow{m} \mathcal{O} : \mathsf{p}\}$ is a set of mapping expressions capturing the semantic mappings between a form tree F_T^i and the ontology $\mathcal{O}$.

Task:

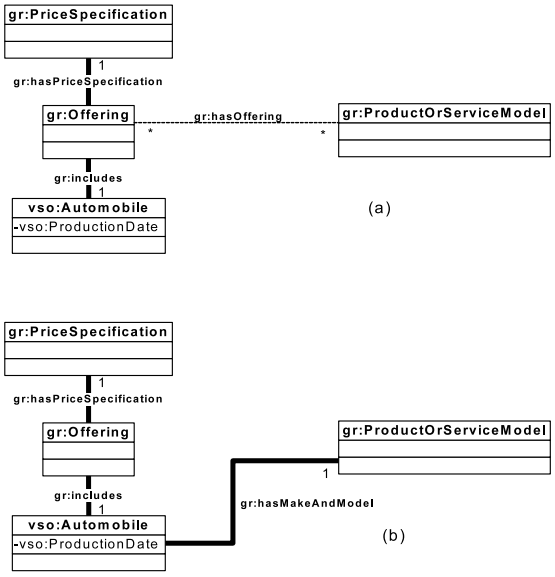

Figure 6: Candidate Connections in Ontology

- To determine a hypothesis h which can infer the desired semantic mapping $M = \{F_T : \mathsf{e} \xrightarrow{m} \mathcal{O} : \mathsf{p}\}$ between a new form tree instance F_T and the ontology $\mathcal{O}$.

To complete the design of the learning algorithm, we must now decide

1. what type of knowledge to be learned?
2. how to represent the hypothesis to be learned?
3. what is the learning mechanism?

EXAMPLE 3.3. *Suppose we have learn a hypothesis h from a set of training examples. For the two candidate connections in Figure 6, the hypothesis should be able to infer that the connection (b) is the most likely one given the form tree in Figure 5. More specifically, the hypothesis h should compute the probabilities of the candidate connections given the form tree, and return the one with the highest conditional probability. Thus, our goal is to learn a probabilistic model that can estimate the conditional probability of each subgraph in the ontology graph given a form tree.* ■

4. A NAIVE BAYESIAN APPROACH

Ontology graphs are often not simple graphs. That is, it is very common that there are multiple parallel relationships between two

concepts. One of the main obstacles for learning a hypothesis to estimate the conditional probability of each subgraph in an ontology graph given a form tree is the intractable number of possible subgraphs. In this section, we propose a naive Bayesian approach to the problem.

At the most general level, we can define the probabilistic inference problem as follows. Let G be a random variable ranging over the set $\mathcal{G}$ of all possible subgraphs of the ontology graph $\mathcal{O}$, and let F_T be a given form tree. Then the most likely subgraph G_{target} that corresponds to the form tree is the one that maximizes $P(G|F_T)$, the conditional probability of a subgraph given the form tree, i.e.,

$$G_{target} = \arg\max_{G \in \mathcal{G}} P(G|F_T) \quad (1)$$

Suppose a set of simple correspondences between the elements of the form tree and the elements of the ontology have been specified. Given the simple correspondences, we can focus on those subgraphs (essentially, subtrees) each of which spans the set of nodes identified by the set of simple correspondences. Note that when we apply a schema matching system to discover simple correspondences, the system may generate multiple sets of candidate correspondences for a single form tree. Each set of candidate correspondences identifies a set of candidate nodes in the ontology graph. Our method searches the space of all the subtrees spanning all the sets of candidate nodes.

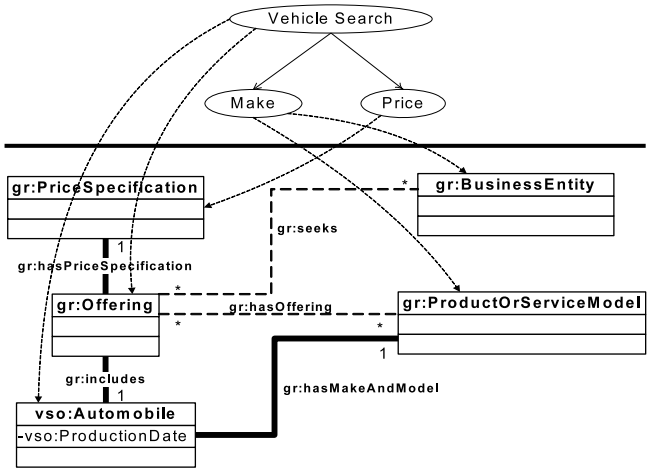

Figure 7: Multiple Candidate Sets Identified by Correspondences

EXAMPLE 4.1. *Figure 7 shows an additional simple correspondence between the form tree and the ontology as shown in Figure 5 before. The* Make *element of the form tree is linked to the ontology concept* gr:BusinessEntity *as well. Taking the correspondences emitting from* Make *in turn, there is an additional candidate connection (using minimum spanning tree) in addition to the previous two in Figure 6. The probabilistic model will estimate the conditional probabilities of all the candidate connections.* ■

Let $F_T = (N, E, <_{sib}, root)$ be a form tree with a set of nodes N and a set of edges E. Let $G = (V_G, E_G)$ be a subtree in the ontology that is identified by a set of correspondences between elements of the form tree and elements of the ontology. Consequently, for each edge $e_i = (v_i, v_j) \in E$ in the form tree, there are two correspondences $v_i \rightsquigarrow s_{e_i}$ and $v_j \rightsquigarrow t_{e_i}$ connecting the end nodes of the edge e_i to two objects in the ontology graph (see Figure 8). We use s_{e_i} to represent the ontology object corresponding to the source node of the edge e_i, and t_{e_i} the object corresponding to the target node of e_i. In the subtree G, there is a unique path between s_{e_i} and t_{e_i} corresponding to each edge e_i in the form tree. Hence, we write a complex mapping between a form tree and an ontology as $M = \{F_T : \mathsf{e}_i \xrightarrow{m} \mathcal{O} : \mathsf{p}^{(s_{e_i}, t_{e_i})}\}$, where $\mathsf{p}^{(s_{e_i}, t_{e_i})}$ is a path in the ontology graph corresponding to an edge e_i in the form tree.

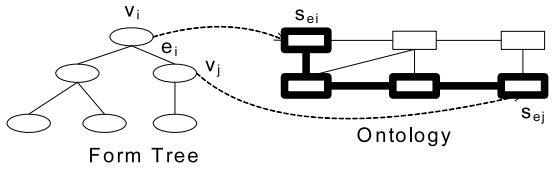

Figure 8: Mapping From a Form-Tree Edge e_i to an Ontology Path $\mathsf{p}^{(s_{e_i}, t_{e_i})}$ (thick lines)

In general, the probability of a subtree G given F_T is the joint probability of all the paths $p^{(s_{e_i}, t_{e_i})}$ for all the edges $e_i \in E$ given F_T, i.e.,

$$P(G|F_T) = P(p^{(s_{e_1}, t_{e_1})}, p^{(s_{e_2}, t_{e_2})}, ..., p^{(s_{e_n}, t_{e_n})}|F_T) \quad (2)$$

where $e_i \in E, i = 1...n$ is the set of edges in the form tree. Now, the question is how to estimate the probabilities of all such subtrees (actually, Steiner trees [15]) spanning the sets of candidate nodes given the form tree F_T. In this paper, we adopt the strategy of the naive Bayes classifier by assuming *a series of conditional independences*. Specifically, we first assume that the paths of a Steiner tree G are *conditionally independent given the form tree.* Having this assumption, therefore, we have:

$$\begin{aligned} P(G|F_T) &= P(p^{(s_{e_1}, t_{e_1})}, p^{(s_{e_2}, t_{e_2})}, ..., p^{(s_{e_n}, t_{e_n})}|F_T) \\ &= \prod_{i=1}^{n} P(p^{(s_{e_i}, t_{e_i})}|F_T) \end{aligned} \quad (3)$$

Let us now focus on a single term $P(p^{(s_{e_i}, t_{e_i})}|F_T)$. Our goal is to learn a model that can estimate the probability from training examples. As is often the case, applying Bayes' rule is helpful:

$$P(p^{(s_{e_i}, t_{e_i})}|F_T) \propto P(F_T|p^{(s_{e_i}, t_{e_i})}) P(p^{(s_{e_i}, t_{e_i})}) \quad (4)$$

We now have two new terms, for which, hopefully, we can learn from examples. We call the term $P(F_T|p^{(s_{e_i}, t_{e_i})})$ a **design model** which describes how a form tree would be designed given an ontology path. The term $P(p^{(s_{e_i}, t_{e_i})})$ is the **prior** probability of ontology paths being modeled by the users in the application domain. Using the element correspondences, we can now treat the nodes of the form tree as the objects in the ontology graph. Given a path $p^{(s_{e_i}, t_{e_i})}$ in the ontology graph, the design model specifies how the designer uses direct edges to connect pairs of the objects in the ontology when designing the form tree.

EXAMPLE 4.2. *Figure 9 illustrates how the design model is used to estimate the probability of a form tree. Figure 9 (a) and (b) show two possible ontology connections among three concepts identified by the correspondences from a form tree:* gr:Offering, gr:PriceSpecification, *and* gr:ProduceOrServiceModel. *The form tree is in the cloud whose node names have been replaced with the corresponding ontology concept names. Each possible connection has two paths between the identified concepts:* p_{a1}, p_{a2} *for the*

connection in (a) and p_{b1}, p_{b2} for the connection in (b). The design model gives rise to the conditional probability of the form tree given a path. For example, $Pr(e_1, e_2|p_{a1})$ is the conditional probability of the form tree given the path p_{a1}. Similarly, $Pr(e_1, e_2|p_{a2})$, $Pr(e_1, e_2|p_{b1})$, and $Pr(e_1, e_2|p_{b2})$ are conditional probabilities of the form tree given other three paths. ■

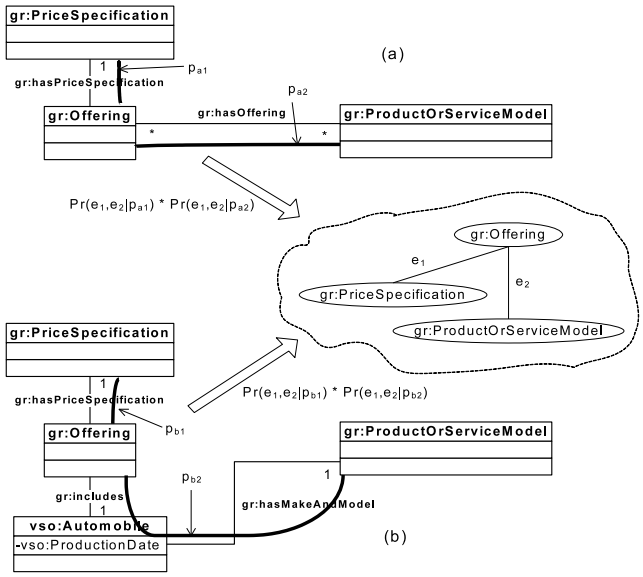

Figure 9: Estimating the Probabilities of a Form Tree Designed from Different Sets of Paths

With the help of the simple correspondences, the nodes of a form tree can be treated as concepts in the corresponding ontology. Despite that, the number of possible form trees is still intractable. To make the learning process feasible, we further assume that the edges of a form tree are *conditionally independent* **given an ontology path**. Therefore, we have:

$$P(F_T|p^{(s_{e_i},t_{e_i})}) = \prod_{j=1}^{n} P(e_j|p^{(s_{e_i},t_{e_i})}), \forall e_j \in E \qquad (5)$$

Now the learning is reduced to learn a model that can estimate the conditional probability of a single edge given a ontology path, i.e., $P(e_j|p^{(s_{e_i},t_{e_i})})$. We will use a corpus of forms $\mathcal{F}$ as training examples to estimate the probability terms $P(e_j|p^{(s_{e_i},t_{e_i})}), \forall e_j \in E$. For each form tree in the training set, the nodes of the tree are labeled with the corresponding objects, and the edges are labeled with the corresponding paths in the ontology. Therefore, each pair of objects in the ontology can be considered as if the pair of objects define a potential edge in the form tree. Hence, we estimate $P(e = (u, v)|p^{(s_{e_i},t_{e_i})})$, the probability of a pair of ontology nodes (u, v) given the path $p^{(s_{e_i},t_{e_i})}$.

To avoid the zero probability problem, we adopt the **m-estimate probability** as defined as: $\frac{n_c+mp}{n+m}$, where n is the number of times the path $p^{(s_{e_i},t_{e_i})}$ appears in the training set, n_c is the number of times the potential edge $e = (u, v)$ appears in the training set together with the path $p^{(s_{e_i},t_{e_i})}$, p is the prior estimate of the probability, and m is a constant called the *equivalent sample size*. With uniform priors and with m equal to the number of pairs of objects in the ontology, the estimate for $P(e|p^{(s_{e_i},t_{e_i})})$ will be:

$$\frac{n_c + 1}{n + m}.$$

To complete the design of our learning algorithm, we still need to estimate the prior probability $P(p^{(s_{e_i},t_{e_i})})$. In general, we can estimate the probability by counting the frequency with which each path occurs in the training data. However, the number of paths in a (multi-)graph is numerous. We will make another reasonable assumption by breaking up a path into edges in the ontology graph and assuming that edges are independent of each other (analogous to the independence assumption of dictionary words in a document in the naive Bayes classifier.) Let $oe_1^p, oe_2^p, ..., oe_m^p \in p^{(s_{e_i},t_{e_i})}$ be the set of ontology edges in the path $p^{(s_{e_i},t_{e_i})}$. By the independence assumption, we have

$$P(p^{(s_{e_i},t_{e_i})}) = \prod_{k=1}^{m} P(oe_k^{p^i}), \qquad (6)$$

where, p^i is a shorthand for $p^{(s_{e_i},t_{e_i})}$).

Consequently, we reduce the problem of estimating the prior probability of a path to estimating the prior probability of each ontology edge. Specifically, for an ontology edge e, we adopt the **m-estimate** once again: $\frac{n_c+1}{n+m}$, where n_c is the number of times the edge e appears in the training data, n is the total count of different ontology edges in the training data, and m is the total number of edges in the ontology graph.

Finally, we need to identify the set $\mathcal{G}$ of possible connections in $G_{target} = \arg\max_{G \in \mathcal{G}} P(G|F_T)$. By adopting the principle of Occam's razor: *prefer the simplest hypothesis*, we will consider the set of *minimum Steiner trees* among the sets of nodes identified by the simple correspondences. Since finding a minimum Steiner tree is NP-complete [15], we use an approximate method in the literature to generate minimum Steiner trees.

Let us put all together. For a Steiner tree G identified by a form tree $F_T = (N, E = \{e_1, e_2, ..., e_n\}, <_{sib}, root)$, let the set of paths of G corresponding to the edges in F_T be $\{p^{(s_{e_1},t_{e_1})}, p^{(s_{e_2},t_{e_2})}, ..., p^{(s_{e_n},t_{e_n})}\}$. We have,

$$\begin{aligned}
P(G|F_T) &\propto P(p^{(s_{e_1},t_{e_1})}, p^{(s_{e_2},t_{e_2})}, ..., p^{(s_{e_n},t_{e_n})}|F_T) \\
&\propto \prod_{i=1}^{n} P(p^{(s_{e_i},t_{e_i})}|F_T) \\
&\propto \prod_{i=1}^{n} P(F_T|p^{(s_{e_i},t_{e_i})}) \times P(p^{(s_{e_i},t_{e_i})}) \\
&\propto \prod_{i=1}^{n} (\prod_{j=1}^{n} P(e_j|p^{(s_{e_i},t_{e_i})}) \times P(p^{(s_{e_i},t_{e_i})})) \\
&\propto \prod_{i=1}^{n} (\prod_{j=1}^{n} P(e_j|p^{(s_{e_i},t_{e_i})}) \times \prod_{k=1}^{m} P(oe_k^{p^i})) \qquad (7)
\end{aligned}$$

where, $oe_k^{p^i}$ is an ontology edge in the path p^i (shorthand for $p^{(s_{e_i},t_{e_i})}$) that is identified by the form tree edge e_i. It is convenient for computation to take logarithm of multiplications, therefore,

Domain	Number of Forms	Avg. Form Tree Size (Total # of Nodes and Edges)	Ontology	Number of Nodes	Number of Edges
Automobile	120	11	Vehicle Sale Ontology	553	3771
Movie	126	13	Movie Ontology	56	115
Book	135	11	Bibliographic Data	1011	2188
Hotel	90	13	Hotel Extension to the GoodRelations Ontology	604	3267
Real Estate	130	15	Real Estate Extension to the GoodRelations Ontology	384	1323

Table 1: Characteristics of Forms and ontologies for the Experiments

$$
\begin{aligned}
P(G|F_T) &\propto \log(\prod_{i=1}^{n}(\prod_{j=1}^{n} P(e_j|p^{(s_{e_i},t_{e_i})}) \times \prod_{k=1}^{m} P(oe_k^{p^i}))) \\
&\propto \sum_{i=1}^{n} \log(\prod_{j=1}^{n} P(e_j|p^{(s_{e_i},t_{e_i})}) \times \prod_{k=1}^{m} P(oe_k^{p^i})) \\
&\propto \sum_{i=1}^{n} (\log(\prod_{j=1}^{n} P(e_j|p^{(s_{e_i},t_{e_i})})) + \log(\prod_{k=1}^{m} P(oe_k^{p^i}))) \\
&\propto \sum_{i=1}^{n} (\sum_{j=1}^{n} \log P(e_j|p^{(s_{e_i},t_{e_i})}) + \sum_{k=1}^{m} \log P(oe_k^{p^i})) \quad (8)
\end{aligned}
$$

This concludes the design of our learning algorithm.

5. EXPERIMENTS

We implemented the proposed learning algorithm and conducted experiments to evaluate its performance. In this section, we report on the experimental results. To the best of our knowledge, we have not found similar work on discovering *complex* mappings between Web forms and ontologies in the literature. Therefore, our evaluation focuses on the performance of the proposed algorithm in this paper. The experiments have several objectives. First, we want to check whether the approach can discover and return the expected mapping as the top-1 result (i.e., the result with the highest ranking in the candidate list) for a test form. Second, If the approach does not return the expected mapping as the top-1 result, we want to know the number of candidates (the k for top-k results) the approach needs to return so that the expected mapping is among the top-k results. Third, we want to evaluate the performance of the approach when incorporated with automatic matching tools. If a set of candidate matchings is discovered by an automatic matching tool, we are interested in whether the approach can discover the expected mapping in the top-k results where k is small.

Datasets: We considered a variety of domains: *Automobile*, *Movie*, *Book*, *Hotel*, and *Real Estate*. For each, a number of forms and a domain ontology were identified. Forms were collected from the Internet starting from two form repositories: the form library[2] created by the survey work [18] and the UIUC Web integration repository[3]. We identified a corresponding ontology for each domain. In particular, we selected the Vehicle Sale Ontology (VSO) [4] for the automobile domain. We used the bibliographic data ontology[5] for the book domain. The movie ontology[6] was selected for the movie domain. Finally, we extended the GoodRelations ontology [14] to the hotel and the real estate domains. Table 1 illustrates the characteristics of the forms and ontologies used in the experiments.

Methodology: For each form, we created a semantic mapping from the form to the corresponding domain ontology. We then conducted 10-fold cross validation for testing the model. In each round, we first used 9-subset training examples to learn the parameters. We then applied the method to the remaining subset that had been set aside for testing. In the process, we incorporated a matching tool that was able to automatically discover sets of simple correspondences between the nodes of the form tree and some nodes in the ontology graph. For each set of simple correspondences, the naive Bayesian method first discovered a set of candidate subgraphs among the set of corresponding nodes in the ontology graph. The method then computed the logarithm value of the posterior probability of a subgraph given the form tree, and ranked all the subgraphs in descending order based on the logarithm values. We then compared the results returned by the method to the expected mappings. We recorded the ranked orders of the expected mappings in the entire lists of results returned. Finally, we analyzed the results to answer the questions we had at the beginning of the experiments.

Results: Figure 10 shows the analysis results about whether the method can return the expected mappings as the top-1 results. The results are illustrated as the average percentages of the test cases when the model returned the expected mappings as the top-1 result. Each column represents the average percentage for a domain. The results show that above 80% of the times, the expected mapping is returned as the top-1 result.

In the same Figure 10, we also show the average total numbers of candidate mappings returned for the test domains. The information is illustrated by the line. For example, for the *Automobile* domain, on average there were about 995 candidate subgraphs returned as the mappings. About 80% of the times, the top-1 result was the expected mapping. Similar information can be read off from the figure. Notably, the sizes of the candidate lists were quite large except for the *Book* domain, where most book search forms use only attributes of single Book concept in the ontology. Without ranking, a user may need to sift through a long list in order to find the expected mapping.

When the expected mappings were not returned as the top-1 results in several domains, we wondered whether the expected mappings could be returned in candidate lists with very small sizes. Figure 11 demonstrates the results. Each column shows that in 100% of the times, the expected mappings were returned by the method. The line shows the numbers of the candidate results in each domain. The maximum number of candidate mappings is 4, i.e., the lowest ranking of an expected mapping is 4. Therefore, we believe that it is possible to dramatically reduce the number of candidate mappings and save a great deal of human efforts for selecting the expected results.

[2] http://cluster.ischool.drexel.edu:8080/ibiosearch/datasets.html
[3] http://metaquerier.cs.uiuc.edu/repository/
[4] http://www.heppnetz.de/ontologies/vso/ns.owl
[5] http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/bibliographic-data/index.html
[6] http://www.movieontology.org/

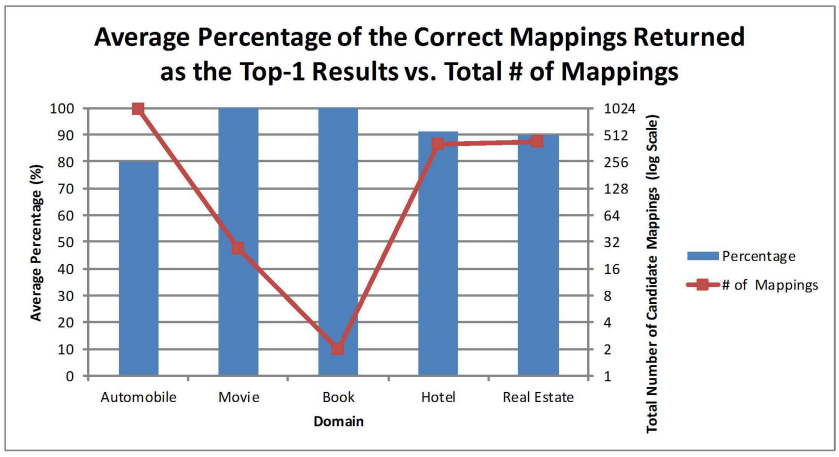

Figure 10: Are the Top-1 Results the Expected Mappings?

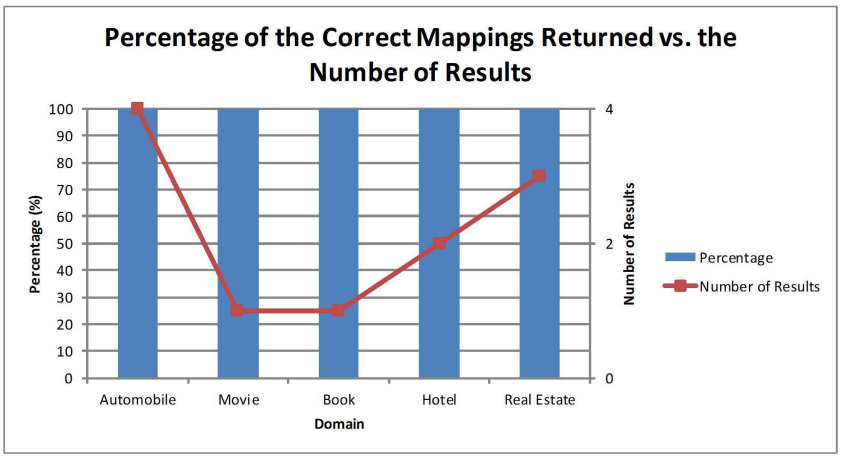

Figure 11: Are the Sizes of the Results Reduced When All the Expected Mappings are Returned?

Discussion: Although the experimental results show that the naive Bayesian method was able to discover the correct mappings and return them as the top-k ($k \leq 4$) results for the test data sets, we need to understand why the correct mappings were not ranked as the top-1 results for some cases. We examined the results and found that the sample sizes of some events were very small in the training data. As a result, the posterior probability of the combination of a set of correct paths might not win over the posterior probability of a set of paths that appeared more frequently in the training data. One solution is to increase the size of training examples. Moreover, we could modify the naive independence assumptions by considering the dependence between a pair of paths or edges.

6. RELATED WORK

We have discussed the limitations of the existing solutions for discovering complex mappings between different data representations including database schemas [22, 26] and ontologies [2, 1]. In this paper, we proposed a machine learning-based approach for discovering complex mappings from Web forms to ontologies. For discovering simple correspondences, various techniques have been proposed [20, 7, 21, 19, 8, 9]. These techniques exploit linguistic and structural information of schemas as well as data instances. In this paper, we used the existing schema matching techniques to generate a set of simple correspondences.

Machine learning techniques have been applied to the problem of discovering schema matchings, i.e., simple correspondences (e..g, Doan et al. in [8, 9]). In this paper, we develop a method that learns probabilistic models through corpora of forms and their associated mappings. A similar corpus-based approach has been proposed for schema matching as well [19], where useful statistics are calculated from schema corpora to improve the accuracy of schema matching.

The study on *form modeling and understanding* [10, 11, 12, 13, 24, 29, 30, 31] aims to model and extract semantic information from a form. Specifically, a form is modeled as a "flat" set of attributes in [11], a visual layout in [12, 13], a set of "query capabilities" in [31], or a hierarchical structure in [30, 10]. To extract these kinds of models, various rule-based and model-based techniques have been proposed. These techniques exploit various information including visual layout, corpora of forms, Web browser rendering, and expert knowledge.

A closely related work is the study on *extracting ontologies from forms* [4, 28]. FAETON [4] is a semi-automatical tool for building tailored ontologies from a set of data acquisition forms. The

method proposed for FAETON analyzes the basic fields and labels, visual and geometric features, and type information on a form to extract a logical model, and then builds a tailored ontology through a set of heuristics. The OntoBuilder project [28] is another pioneer work on supporting the extraction of ontologies from Web interface forms. The algorithms in OntoBuilder identify structural information using composition and precedence constructs. Ontologies are used as the underlying data model for discovering matching between forms. Our study is different from the above works in that we explored machine learning techniques for discovering complex mapping between Web forms and *existing* ontologies.

7. CONCLUSION

We are motivated to study an automatic approach for linking various Web forms to ontologies on the Semantic Web. We proposed a solution by exploiting machine learning techniques. In the learning process, we were faced with two challenges: the need to collect large amounts of training examples and the need to estimate an intractable number of joint probabilities. To overcome, we adopted the strategy of the naive Bayes classifier. A form is first converted to a tree structure using an existing technique. Then a semantic mapping between a form tree and an ontology is specified in terms of a set of relationships between tree edges and ontology paths. The naive Bayesian approach assumes that ontology paths are conditionally independent given a form tree. Our experimental results show that the approach is effective and has the potential to save significant efforts in discovering mappings from forms to ontologies. Such a mapping would be very useful for integrating data residing in heterogeneous online databases.

There are a couple of limitations in the naive Bayesian approach. The first one is obviously the oversimplified assumptions about the conditional independence between ontology paths and edges. The second one is using Steiner trees as candidate connections among a set of identified concepts in the ontology. In future, we intend to exploit more sophisticated probabilistic models such as Bayesian networks to capture more realistic conditional independence among ontology elements. In addition, we will examine more efficient algorithms to connect ontology concepts through meaningful links.

References

[1] Y. An, A. Borgida, and J. Mylopoulos. Constructing Complex Semantic Mappings between XML Data and Ontologies. In *ISWC'05*, pages 6–20, 2005.

[2] Y. An, A. Borgida, and J. Mylopoulos. Inferring Complex Semantic Mappings between Relational Tables and Ontologies from Simple Correspondences. In *ODBASE*, pages 1152–1169, 2005.

[3] Y. An, R. Khare, I.-Y. Song, and X. Hu. Automatically mapping and integrating multiple data entry forms into a database. In *the Proceedings of the 30th International Conference on Conceptual Modeling (ER'11)*, 2011.

[4] R. Berlanga, E. Jimenez-Ruiz, V. Nebot, and I. Sanz. Faeton: Form analysis and extraction tool for ontology construction. 2(2), 2008.

[5] A. Bonifati, E. Q. Chang, T. Ho, V. S. Lakshmanan, and R. Pottinger. HePToX: Marring XML and Heterogeneity in Your P2P Databases. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, pages 1267–1270, 2005.

[6] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering Complex Semantic Matches between Database Schemas. In *Proceedings of the ACM SIGMOD*, pages 383–394, 2004.

[7] H. H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proceedings of the International Conference on Very Large Data bases (VLDB)*, pages 610–621, 2002.

[8] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine learning approach. In *Proceedings of ACM SIGMOD*, pages 509–520, 2001.

[9] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the International WWW Conference*, pages 662–673, 2002.

[10] E. C. Dragut, T. Kabisch, C. T. Yu, and U. Leser. A hierarchical approach to model web query interfaces for web source integration. *PVLDB*, 2(1):325–336, 2009.

[11] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 217–228, New York, NY, USA, 2003. ACM.

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

[13] H. He, W. Meng, C. Yu, and Z. Wu. Automatic integration of web search interfaces with wise-integrator. *The VLDB Journal*, 13(3):256–273, 2004.

[14] M. Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008)*, volume LNCS 5268, pages 332–347, 2008.

[15] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. Annals of Discrete Mathematics, 53, 1992.

[16] Z. Kedad and X. Xue. Mapping Discovery for XML Data Integration. In *Proceedings of International Conference on Cooperative Information Systems (CoopIS)*, pages 166–182, 2005.

[17] R. Khare and Y. An. An empirical study on using hidden markov model for search interface segmentation. In *Proceedings of 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 17–26, 2009.

[18] R. Khare, Y. An, and I.-Y. Song. Understanding search interfaces: A survey. *SIGMOD Record*, 39(1):33–40, 2010.

[19] J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpus-Based Schema Matching. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 57–68, 2005.

[20] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB '01*, pages 49–58, 2001.

[21] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 117–128, 2002.

[22] R. J. Miller, L. M. Haas, and M. A. Hernandez. Schema Mapping as Query Discovery. In *VLDB*, pages 77–88, 2000.

[23] T. Milo and S. Zohar. Using Schema Matching to Simplify Heterogeneous Data Translation. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, pages 122–133, 1998.

[24] H. Nguyen, T. Nguyen, and J. Freire. Learning to extract form labels. *Proc. VLDB Endow.*, 1(1):684–694, 2008.

[25] H. Nottelmann and U. Straccia. Information retrieval and machine learning for probabilistic schema matching. *Information Processing and Management*, 43:52–576, 2007.

[26] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating web data. In *VLDB*, pages 598–609, 2002.

[27] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10:334–350, 2001.

[28] H. Roitman and A. Gal. Ontobuilder: Fully automatic extraction and consolidation of ontologies from web sources using sequence semantics. In *EDBT Workshops*, pages 573–576, 2006.

[29] J. Wang and F. Lochovsky. Data extraction and label assignment for web databases. In *12th International Conference on World Wide Web*, pages 187–196, 2003.

[30] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *SIGMOD '04*, pages 95–106, New York, NY, USA, 2004. ACM.

[31] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: best-effort parsing with hidden syntax. In *SIGMOD '04*, pages 107–118, New York, NY, USA, 2004. ACM.