Discovering Simple Mappings Between Relational Database Schemas and Ontologies

Wei Hu and Yuzhong Qu

School of Computer Science and Engineering, Southeast University,
Nanjing 210096, P.R. China
{whu, yzqu}@seu.edu.cn

Abstract. Ontologies proliferate with the growth of the Semantic Web. However, most of data on the Web are still stored in relational databases. Therefore, it is important to establish interoperability between relational databases and ontologies for creating a Web of data. An effective way to achieve interoperability is finding mappings between relational database schemas and ontologies. In this paper, we propose a new approach to discovering simple mappings between a relational database schema and an ontology. It exploits simple mappings based on virtual documents, and eliminates incorrect mappings via validating mapping consistency. Additionally, it also constructs a special type of semantic mappings, called contextual mappings, which is useful for practical applications. Experimental results demonstrate that our approach performs well on several data sets from real world domains.

1 Introduction

The popularity of ontologies is rapidly growing since the emergence of the Semantic Web. To date, the amount of available Web ontologies continues increasing at a phenomenal rate. For example, Swoogle [10] has collected more than 10,000 ontologies so far. However, most of the world's data today are still locked in data stores and are not published as an open Web of inter-referring resources [4]. In particular, as reported in [6], about 77.3% data on the current Web are stored in relational databases (the so-called "*Deep Web*"). Therefore, it is necessary to actualize interoperability between (Semantic) Web applications using relational databases and ontologies.

In order to achieve such interoperability, an effective way is to discover mappings between relational database schemas and ontologies. Although relational databases are based on closed-world assumption while ontologies use open-world semantics, there usually exist some approximate correspondences between them. For instance, an attribute in a relational database schema may correspond to a property in an OWL ontology. In fact, relational databases can be formalized by *First Order Logic* (FOL) [21]; while the logical foundation for OWL ontologies is *Description Logic* (DL) [2], which is a subset of FOL. Thereupon, it is feasible to construct mappings between relational database schemas and ontologies.

Discovering mappings between a relational database schema and an ontology usually employs a two-phase paradigm: (i) searching simple mappings between entities in the relational database schema and the ontology, and (ii) constructing complex compositions based on simple mappings. Finding simple mappings is an early and fundamental stage for constructing complex compositions [1]. In this paper, we focus on the first problem, i.e., discovering simple mappings between a relational database schema and an ontology.

Manually discovering such simple mappings is tedious and improbable at the Web scale. Although many (semi-)automatic approaches have been proposed to address this issue (e.g. [7, 13, 17]), pursuant to the results of our investigation, they have not well considered the characteristics of relational database schemas and ontologies, so the mappings they exploited are not accurate enough. Besides, most of the present approaches cannot construct semantic mappings, which are demonstrated to be useful in various practical applications (e.g. [8]).

In this paper, we propose a new approach to discovering simple mappings. It constructs virtual documents for the entities in a relational database schema as well as an ontology, so it can discover mappings from a semantic perspective, and it validates mapping consistency, so it can eliminate certain incorrect mappings. In addition, the approach constructs a special type of semantic mappings, called *contextual mappings* [5], between relations in the relational database schema and classes in the ontology. The contextual mappings can be transformed directly to view-based mappings with selection conditions, which are useful for applications from real world domains.

The remainder of this paper is organized as follows. Section 2 gives the definitions of discovering simple mappings between a relational database schema and an ontology. Section 3 sketches out our approach. Section 4 describes the approach in details. Section 5 evaluates the approach on several cases from real world domains. Section 6 discusses some related works. Finally, Section 7 concludes the paper with future work.

2 Problem Statement

Followed by [19], a *data model* is a collection of high-level data description constructs that hide many low-level storage details. A description of data in terms of a data model is called a *schema*.

Definition 1. *A relational database schema, $\mathbb{R}$, is a finite collection of relation schemas. A relation schema consists of the name of the relation, the names of the attributes in the relation along with their associated domains. A domain is referred to in a relation schema by the domain name and has a set of associated values. A relational database schema specifies a set of integrity constraints (ICs), which restrict the data instances that can be stored in the database.*

In our notation, $\mathcal{R}$ denotes a relation, and $\mathcal{A}$ denotes an attribute. $type(\mathcal{A})$ gets the domain name of $\mathcal{A}$. $rel(\mathcal{A})$ gets the relation which specifies $\mathcal{A}$. $pk(\mathcal{R})$ returns the attributes appeared as the primary keys of $\mathcal{R}$. $ref(\mathcal{A})$ returns the attributes referenced by $\mathcal{A}$. Without ambiguous, we use *relational schemas* instead of relational database schemas.

An *ontology* is an explicit specification of a shared conceptualization [14]. In this paper, we propose a simple definition based on [15, 18].

Definition 2. *An ontology, $\mathbb{O}$, is a pair $\mathbb{O} = (S, A^0)$, where S is the signature describing the vocabulary, while A^0 is a set of axioms specifying the intended interpretation of the vocabulary in some domain of discourse. Further, S is the disjoint union of sets of classes, properties and individuals in OWL DL.*

Without explanation, ontologies used in this paper are expressed by OWL DL. For notation, we use $\mathcal{C}$ to represent a class, and $\mathcal{P}$ to represent a property. Further, $\mathcal{P}_\mathcal{D}$ denotes a datatype property and $\mathcal{P}_\mathcal{O}$ denotes an object property. $d(\mathcal{P})$ gets the domain(s) of $\mathcal{P}$, and $r(\mathcal{P})$ gets its range(s).

Similar to the definition in [23], we define discovering simple mappings between a relational schema and an ontology as follows.

Definition 3. *Let $\mathbb{R}$ be a relational schema and $\mathbb{O}$ be an ontology, discovering simple mappings between $\mathbb{R}$ and $\mathbb{O}$ gets a set of mappings $M = \{m\}$. A mapping m is a 5-tuple: $< id, u, v, t, f >$, where id is a unique identifier; u is an entity in $\{\mathcal{R}\} \cup \{\mathcal{A}\}$, and v is an entity in $\{\mathcal{C}\} \cup \{\mathcal{P}\}$; t is a relationship (e.g. equivalence $(=)$, subsumption $(\sqsupseteq)$) holding between u and v; and f is a confidence measure in the $[0, 1]$ range.*

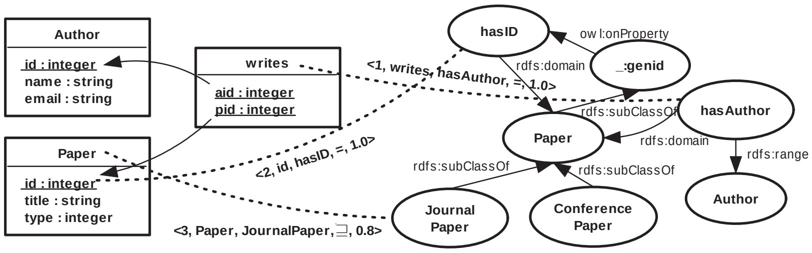


Fig. 1. A toy example

To help understanding, we illustrate a toy example here. Looking at the relational schema depicted in the left part of Fig. 1, it contains three relations: `Author`, `Paper`, and `writes`. Each relation has a set of attributes, e.g., `Author` has three attributes: <u>`id`</u>, `name`, and `email`. The underlined attribute, such as <u>`id`</u>, indicates the primary key of the relation. The arrow represents a *referential* IC, e.g., the foreign key <u>`aid`</u> in `writes` references the key <u>`id`</u> in `Author`. The ontology shown in the right part of Fig. 1 contains four classes, i.e., `Author`, `Paper`, `ConferencePaper` and `JournalPaper`, and two properties, i.e., `hasID` and hasAuthor. We could further recognize that hasID is a datatype property and hasAuthor is an object property. Besides, ConferencePaper and JournalPaper are two subclasses of Paper. Paper and hasID are linked by a *restriction* construct. If we match the relational schema with the ontology, we possibly obtain some mappings (numbered 1–3 with dotted lines) in Fig. 1, where the first and the second ones are two mappings holding the equivalence relationships respectively, and the third is a mapping holding the subsumption relationship.

3 Overview of the Approach

The overview of our approach is illustrated in Fig. 2. In general, it starts with a relational schema and an ontology, and after four processing stages, it outputs a set of simple mappings.

- *Phase 1: Classifying entity types.* This phase is a preprocessing process. It heuristically classifies entities in the relational schema and the ontology into four different groups to limit the searching space of candidate mappings. Besides, this phase coordinates different characteristics between the relational schema and the ontology.

- *Phase 2: Discovering simple mappings.* This phase firstly constructs virtual documents for the entities in the relational schema and the ontology to capture their implicit semantic information. Then, it discovers simple mappings between entities by calculating the confidence measures between virtual documents via the TF/IDF model [22].

- *Phase 3: Validating mapping consistency.* This phase uses mappings between relations and classes to validate the consistency of mappings between attributes and properties. It considers the compatibility between data types of attributes and properties as well. In addition, some inference rules are also integrated in this process.

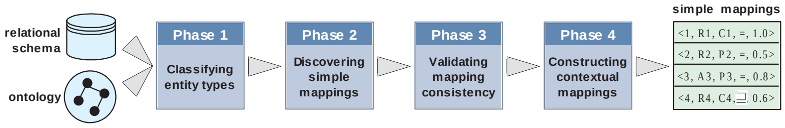


Fig. 2. Overview of the approach

- *Phase 4: Constructing contextual mappings.* This phase operates on mappings between relations and classes found in the previous phases, and supplies them with sample instances. It constructs a set of contextual mappings, which indicate the conditions how they could be transformed to view-based mappings with selection conditions.

4 Details of the Approach

In this section, we describe each of the four processing stages in details.

4.1 Classifying Entity Types

A relation in a relational schema can be classified into four disjoint types based on the properties of its primary keys: *Strong Entity Relation* ($\mathcal{SER}$), *Weak Entity Relation* ($\mathcal{WER}$), *Regular Relationship Relation* ($\mathcal{RRR}$), and *Specific Relationship Relation* ($\mathcal{SRR}$). An attribute can be distinguished into two categories by whether it is a foreign key: *Foreign Key Attribute* ($\mathcal{FKA}$) and *Non Foreign Key Attribute* ($\mathcal{NFKA}$). Please refer to [9] for formal definitions.

Generally, a strong (or weak) entity relation would heuristically match a class in an ontology; while a regular (or specific) relationship relation would heuristically match an object property. For example, the relation `Author` in Fig. 1 is a strong entity relation, and matches the class `Author`; while the relation `writes` is a regular relationship relation, and matches the object property `hasAuthor`.

Similarly, if an attribute is a foreign key attribute, it would match an object property; otherwise, it would match either a datatype or an object property. One exception needs to be noticed. If a relation is a regular (or specific) relationship relation, all its attributes appeared as primary keys as well as foreign keys are unnecessary to participate in the process of discovering mappings. For instance, in Fig. 1, the relation `writes` is a regular relationship relation, so the attributes `aid` and `pid` should not be considered anymore.

According to the heuristic classification, we partition entities in the relational schema and the ontology into the following four groups.

- *Group 1:* $\{\{\mathcal{SER}\} \cup \{\mathcal{WER}\}\} \times \{\mathcal{C}\}$.
- *Group 2:* $\{\{\mathcal{RRR}\} \cup \{\mathcal{SRR}\}\} \times \{\mathcal{P_O}\}$.
- *Group 3:* $\{\mathcal{FKA}\} \times \{\mathcal{P_O}\}$.
- *Group 4:* $\{\mathcal{NFKA}\} \times \{\{\mathcal{P_D}\} \cup \{\mathcal{P_O}\}\}$.

Besides, we consider some preprocessing steps to coordinate different characteristics between the relational schema and the ontology. As an example, a regular (or weak) relationship relation should be copied so that it could match two object properties holding the *inverseOf* construct. As another example, an n-arity relationship ($n \geq 3$) should be reified as a group of binary relationships, because OWL ontologies can only express unary and binary relationships [2]. Please note that the heuristic rules above are not complete, but they are effective in a lot of application scenarios.

4.2 Discovering Simple Mappings

Inspired by [20], we present a method in this paper, which considers the structures of both the relational schema and the ontology to exploit their semantic information. The rationalities are as follows: the semantic information of a relational schema is characterized mainly by its ICs. For instance, a referential IC involves two relations and associates a group of attributes in one relation to the keys of another relation. Likewise, an OWL ontology can be mapped to an RDF graph [18], which also indicates the semantic information in its structure.

We construct virtual documents, denoted by $VD(.)$, for the entities in both the relational schema and the ontology to capture their structural information. A virtual document represents a collection of weighted tokens, which are derived not only from the description of the entity itself, but also from the descriptions of its neighbors. The weights of the tokens indicate their importance, and could then be viewed as a vector in the TF/IDF model [22].

The formulae in (1)–(2) build virtual documents for relations and attributes respectively. In general, for a relation, if it is a strong (or weak) entity relation, then the virtual document is from its local description; otherwise, it is a regular (or specific) relationship relation, then the virtual document is from its local description as well as the descriptions of the referenced relations. For an attribute, if it is a foreign key attribute, besides involving the description itself, we further consider the descriptions of its referenced relations; otherwise, we complement its data type into account.

$$VD(\mathcal{R}) = \begin{cases} Des(\mathcal{R}) & \mathcal{R} \in \{\mathcal{SER}\} \cup \{\mathcal{WER}\} \\ Des(\mathcal{R}) + \alpha \cdot \sum_{\substack{\mathcal{A}' \in ref(\mathcal{A}) \\ \mathcal{A} \in pk(\mathcal{R})}} Des(rel(\mathcal{A}')) & \mathcal{R} \in \{\mathcal{RRR}\} \cup \{\mathcal{SRR}\} \end{cases}, \tag{1}$$

$$VD(\mathcal{A}) = \begin{cases} Des(\mathcal{A}) + \alpha \cdot (Des(rel(\mathcal{A})) + \sum_{\mathcal{A}' \in ref(\mathcal{A})} Des(rel(\mathcal{A}'))) & \mathcal{A} \in \{\mathcal{FKA}\} \\ Des(\mathcal{A}) + \alpha \cdot Des(rel(\mathcal{A})) + \beta \cdot Des(type(\mathcal{A})) & \mathcal{A} \in \{\mathcal{NFKA}\} \end{cases}. \tag{2}$$

Analogously, the formulae in (3)–(4) construct virtual documents for classes and properties respectively. In brief, for a class, its virtual document is its local description. For a property, we consider not only its local description, but also the descriptions of its domain and range classes. Please note that if the property is a datatype property, its range is actually its data type.

$$VD(\mathcal{C}) = Des(\mathcal{C}), \tag{3}$$

$$VD(\mathcal{P}) = \begin{cases} Des(\mathcal{P}) + \alpha \cdot (\sum_{\mathcal{C} \in d(\mathcal{P})} Des(\mathcal{C}) + \sum_{\mathcal{C} \in r(\mathcal{P})} Des(\mathcal{C})) & \mathcal{P} \in \{\mathcal{P}_{\mathcal{O}}\} \\ Des(\mathcal{P}) + \alpha \cdot \sum_{\mathcal{C} \in d(\mathcal{P})} Des(\mathcal{C}) + \beta \cdot Des(r(\mathcal{P})) & \mathcal{P} \in \{\mathcal{P}_{\mathcal{D}}\} \end{cases}. \tag{4}$$

For simplicity, in this paper, we define $Des(.)$ merely returns the name of an entity as its local description. α and β are fixed rational numbers in $[0, 1]$. The values should be configured with respect to practical cases. In our experience, α should be a little larger than β.

As an example to explain the construction of virtual documents, let us see the regular relationship relation `writes` in Fig. 1. Its local description only includes "write", and its two neighbors are `Paper` and `Author`. The virtual document of `writes` is $VD(\texttt{writes}) = \{\text{"write"}, \alpha \cdot \text{"paper"}, \alpha \cdot \text{"author"}\}$.

We discover simple mappings by calculating the confidence measures between entities. The confidence measure between any two entities is calculated by the cosine value between two vectors N_i and N_j, corresponding to two virtual documents VD_i and VD_j in the TF/IDF model:

$$confidence(VD_i, VD_j) = cosine(N_i, N_j) = \frac{\sum_{k=1}^{l} n_{ik} n_{jk}}{\sqrt{\sum_{k=1}^{l} n_{ik}^2 \sum_{k=1}^{l} n_{jk}^2}}, \quad (5)$$

where l is the dimension of the vector space, and n_{ik} (n_{jk}) is the component of the vector. If the two virtual documents do not share any tokens, the confidence measure would be 0.0. If all the token scores equal completely, it would be 1.0.

4.3 Validating Mapping Consistency

In a relational database, relations are the unique building structures while attributes are defined by relations within their local scopes, i.e., attributes cannot stand alone without relations. In contrast, classes and properties in an OWL ontology are both first-class citizens. Nevertheless, the restriction construct in an OWL ontology provides a way of specifying local domain and range constraints on the classes [18].

In this paper, we use this kind of constraints between relations (classes) and attributes (properties) to check the consistency between mappings. We firstly assume that the mappings between relations and classes are correct, and then we utilize these mappings to validate the consistency of the candidate mappings between attributes and properties. Please note that some inference rules should be considered as well. For example, in Fig. 1, the domains of the datatype property `hasID` should include the classes `JounalPaper` and `ConferencePaper`. Besides, we also check the compatibility of data types between non foreign key attributes and datatype properties.

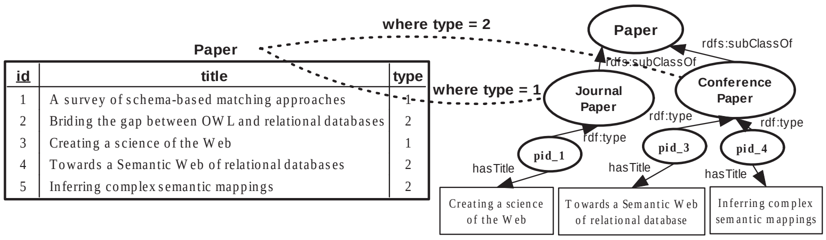
Considering the example shown in Fig. 1 again, we assume that the mapping between the relation `Paper` and the class `Paper` has been discovered, then we validate two candidate mappings: the mapping between the attribute `id` in `Paper` and the property `hasID`, and the mapping between the attribute `id` in `Author` and the property `hasID`. It is obvious that the latter one is inconsistent with the mapping between the relation `Paper` and the class `Paper`.

4.4 Constructing Contextual Mappings

Data integration is a traditional application for matching [23]. As an important infrastructure, query answering provides certain answers of queries over mappings. In [8], it has been proven that subsumption relationships are helpful for the optimization of query answering. It inspires us to construct mappings holding the subsumption relationships between a relational schema and an ontology.

A naive approach to construct such mappings holding the subsumption relationships is reusing the hierarchies of entities in a relational schema and an ontology. For the ontology, the hierarchy of entities is explicitly specified by the *subClassOf* constructs; while for the relational schema, the *Reverse Engineering* techniques (e.g. [9]) could also help us recover such hierarchy. But the naive approach would suffer from finding too many mappings holding the subsumption relationships, and most of them are not useful in practical applications.

In this paper, we focus on searching a special type of mappings holding the subsumption relationships, called *contextual mappings*. It can be directly translated to conditional mappings or view-based mappings [5]. Let us see the example illustrated in Fig. 3. In the example, when the value of the attribute `type` in the relation `Paper` equals to "1", the relation `Paper` matches the class `JournalPaper`; and when the value equals to "2", it matches the class `ConferencePaper`. So the contextual mappings not only hold the subsumption relationships, but also explain the conditions how they can be converted to the equivalence relationships.



Paper

id	title	type
1	A survey of schema-based matching approaches	1
2	Briding the gap between OWL and relational databases	2
3	Creating a science of the Web	1
4	Towards a Semantic Web of relational databases	2
5	Inferring complex semantic mappings	2

Fig. 3. The toy example with instances

Constructing contextual mappings requires two preconditions: (i) certain input mappings, and (ii) sample instances for both the relational schema and the ontology. To the first precondition, we have already found such mappings in the previous subsections. To the second one, a database is a collection of data [19], so it always contains instances; while according to the report by Swoogle [10], only 1.5% Semantic Web ontologies have few individuals. So it is possible to provide some overlapped instances for both the relational schema and the ontology.

The algorithm ContextMatch is shown in Table 1. The input of the algorithm is a relational schema with associated sample data, an ontology with associated sample individuals, and a set of simple mappings found previously. The goal of the algorithm is to assemble a collection of contextual mappings. To accomplish this, the algorithm considers each input mapping between a relation and a class in turn at line 1, and selects the data $J_{\mathcal{R}}$ from the relation at line 2.

Next, in lines 3–5, the algorithm enumerates all the disjoint subclasses of the class, and for each subclass, it selects its individuals $I^k_{\mathcal{C}}$, and finds the instances $J^k_{\mathcal{R}}$ from $J_{\mathcal{R}}$ which match $I^k_{\mathcal{C}}$ by InstanceMatch. As a result, $J_{\mathcal{R}}$ is partitioned into some disjoint subsets corresponding to the subclasses. In our current implementation, InstanceMatch is developed by comparing the values of the instances through the input mappings between attributes and properties. Please note that InstanceMatch is the most time-consuming routine throughout the whole algorithm, and its complexity is relevant to the sizes of sample instances.

Table 1. An algorithm for constructing contextual mappings

```
Algorithm: ContextMatch($\mathbb{S}, \mathbb{O}, J, I, M$).

Input: A relational schema $\mathbb{S}$, with associated sample data $J$,
       an ontology $\mathbb{O}$, with associated sample individuals $I$,
       and a set of simple mappings $M$.
Output: A set of contextual mappings $M'$.

 1.  for each $m =< id, u, v, t, f >$ in $M$ s.t. $u$ is a relation and $v$ is a class
 2.      $J_{\mathcal{R}}$ := InstanceSelect($u, \mathbb{S}, J$);
 3.      for each $\mathcal{C}_k$ s.t. $\mathcal{C}_k$ is a subclass of $v$ and disjoint with any other $\mathcal{C}_{k'}$
 4.          $I^k_{\mathcal{C}}$ := InstanceSelect($\mathcal{C}_k, \mathbb{O}, I$);
 5.          $J^k_{\mathcal{R}}$ := InstanceMatch($J_{\mathcal{R}}, I^k_{\mathcal{C}}, M$);
 6.      for each $\mathcal{A}_i$ in $u$ s.t. $\mathcal{A}_i$ is a categorical attribute
 7.          $ig_i$ := InfomationGain($\mathcal{A}_i, \{J^k_{\mathcal{R}}\}$);
 8.      $l := \arg\max(ig_i)$;
 9.      for each $\mathcal{C}_k$ s.t. $ig_l > \tau$
10.          $M' := M' \cup \{< new\_id, u, \mathcal{C}_k, \mathcal{A}_l =$ "xxx"$, ig_l >\}$;
11.  return $M'$;
```

In lines 6–8, the algorithm repeatedly examines each attribute in the relation whether it is a categorical attribute or not. If so, InformationGain computes the information gain (IG) of the attribute on $\{J^k_{\mathcal{R}}\}$. The attribute $\mathcal{A}_l$, which has the maximal IG, is chosen as the best splitting attribute. Please note that computing IG for classification (e.g. *decision tree*) has been widely studied in the fields of Machine Learning and Data Mining.

In the end, in lines 9–11, if the IG of $\mathcal{A}_l$ is larger than a given threshold τ, then we construct a new contextual mapping, and add it to the set of contextual mappings as output.

5 Evaluation

We have implemented the proposed approach in Java, called Marson (Mapping between relational schemas and ontologies). In this section, we report on some results of an experimental study. Please note that all the test cases and experimental results are available at our website [1].

5.1 Case Study

In our evaluation, we choose the data sets used in [1], which can be downloaded from the website [2]. The data sets are obtained from a variety of real world domains, and the relational database schema and the ontology in each data set are

[1] http://iws.seu.edu.cn/infores/tools/falcon-ao/marson.zip

[2] http://www.cs.toronto.edu/∼yuana/research/maponto/relational/testData.html

developed independently. Volunteers are trained to set up reference mappings. The statistical data of the data sets are listed in Table 2.

Table 2. Characteristics of data sets

ID	$\mathbb{R}$	$\sharp\,\mathcal{R}$	$\sharp\,\mathcal{A}$	$\mathbb{O}$	$\sharp\,\mathcal{C}$	$\sharp\,\mathcal{P}$	Ref.
1	UTCS	8	32	Univ. CS	53	35	18
2	VLDB	9	38	Conference	18	29	27
3	DBLP	5	27	Bibliography	66	81	21
4	OBSERVER	8	115	Bibliography	66	81	72
5	Country	6	18	Factbook	43	209	22

5.2 Experimental Methodology

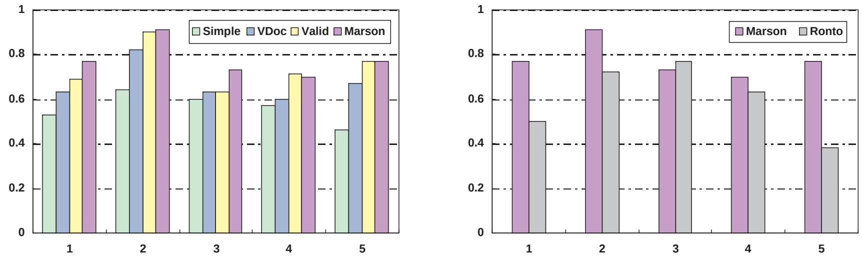
Two experiments are designed to evaluate Marson. In the first experiment, we measure the performance of Marson on discovering simple mappings (without contextual mappings) between a relational schema and an ontology. Four approaches are set up for comparison: (a) a simple approach, denoted by Simple, which only utilizes the local descriptions of the entities (i.e. $\alpha = 0, \beta = 0$ in (1)–(4)) for calculating the confidence measures in the TF/IDF model, and does not validate the consistency between mappings; (b) an approach, denoted by VDoc, which discovers simple mappings by constructing virtual documents, but does not check mapping consistency; (c) an approach, denoted by Valid, which only validates the consistency between mappings found by the simple approach; and (d) a simple version of an existing prototype Ronto [17]. We have implemented it based on I-Sub [24] as its elementary matcher for calculating the confidence measures between entities. Please refer to Section 6 for a detailed introduction. The parameters of VDoc and Marson in (1)–(4) are uniformly set as follows: $\alpha = 0.2, \beta = 0.1$. Please note that our tests also show that Marson is stable with slight changes on α and β.

In the above experiment, we use the well known *F1-Measure* (a combination of *precision* and *recall*) to evaluate the performance of each approach. We have tested a variety of cutoffs or thresholds for each approach, and selected the best ones in our experiments. It seems fair to all the approaches.

In the second experiment, we evaluate the effectiveness of Marson on constructing contextual mappings. Some real instances are collected from the Web corresponding to the relational schemas and ontologies in the first three data sets (more than 50 instances for each relation and class). We look into the contextual mappings found by our algorithm by comparing with the mappings established by experienced volunteers.

5.3 Discussion on Experimental Results

The results on measuring the F1-Measures of Simple, VDoc, Valid and Marson are illustrated in Fig. 4(a). It shows that either VDoc or Valid performs better than SIMPLE, and MARSON is dominant in most data sets. More specifically, VDOC improves SIMPLE in tests 1, 2, and 5, because it can discover the mappings between the entities having little commonality in their local descriptions. VALID enhances SIMPLE in almost all the data sets, since it often occurs that the relational schema in each data set has some attributes in different relations owning the same names such as "id" or "name". But the mappings found additionally by VDOC and VALID are not completely orthogonal, some of them are overlapped. Based on the experiment, MARSON is the best one on nearly all the data sets except for a slight lag in test 4. It demonstrates that it is feasible to integrate VDOC and VALID together and achieve a good result.



(a) SIMPLE, VDOC, VALID and MARSON (b) MARSON and RONTO

Fig. 4. Comparison of F1-Measure

The comparison results between MARSON and RONTO are shown in Fig. 4(b). It indicates that MARSON performs better than RONTO in average F1-Measure. The reason is that MARSON can find additional correct mappings by VDOC, and eliminate some inconsistent mappings by VALID.

Furthermore, it is valuable to mention that MARSON is quite efficient in the first experiment. Based on our environment (Intel Pentium IV 2.8GHz processor, 512MB memory, Windows XP Professional, and Java SE 6), it takes about 5 seconds to complete all the five tests (including the parsing time).

In the second experiment, the contextual mappings constructed by our algorithm are evaluated by experienced volunteers, and the results are exhibited in Table 3. MARSON constructs some interesting contextual mappings. For instance, in test 2, MARSON constructs a contextual mapping between the relation `Event` and the class `Conference`. It points out that when the values of the attribute `type` in `Event` equals to "Research Session" or "Industrial Session", the subsumption relationship between `Event` and `Conference` can be converted to the equivalence relationship. In most tests, our algorithm finds all the possible contextual mappings. But in test 1, it misses the contextual mapping between the relation `academic_staff` and the subclasses of `Faculty` (e.g. `Professor`), because without background knowledge, MARSON cannot discover the mapping between `academic_staff` and `Faculty`.

Table 3. Evaluation of contextual mappings

ID	Found	Existing	Correct
1	2	3	2
2	1	1	1
3	1	1	1

6 Related Work

Discovering mappings between relational database schemas and ontologies is an interdisciplinary research in both Database and Semantic Web communities. At an early stage, some works (e.g. [7]) try to implement visual toolkits in order to help users specify mappings manually. This kind of approaches may succeed in some specific scenarios, but they are impractical for the scale of the Web.

At present, many works focus on discovering mappings (semi-)automatically. For example, Dragut and Lawrence [13] transform relational schemas and ontologies into directed labeled graphs respectively, and reuse the schema matching tool COMA [11] to exploit simple mappings. Papapanagiotou et al. [17] develop a plug-in named Ronto, which introduces six different strategies to discover mappings by distinguishing the types of entities in relational schemas, and it is similar to the Simple approach in this paper. However, all the approaches mentioned above disregard the structural differences in models, and do not validate the consistency between mappings.

Furthermore, to the best of our knowledge, no existing work raises the issue of constructing semantic mappings between relational schemas and ontologies. In both Database and Semantic Web communities, more and more researchers have been aware of the importance for constructing semantic mappings (e.g. [5, 8]), and we believe it is also necessary to consider semantic mappings between relational schemas and ontologies. In this paper, we propose a novel algorithm to find a special type of semantic mappings, called contextual mappings, which can directly help query answering and data integration.

Besides, there exist some literatures addressing the problem from other directions. For example, Dou et al. [12] describe a general framework for integrating databases with ontologies via a first-order ontology language Web-PDDL. Barrasa et al. [3] design a language R2O for expressing complex mappings. Motik et al. [16] propose an extension of OWL with ICs that captures the intuition behind ICs in relational databases. An et al. [1] develop a prototype MapOnto for inferring complex semantic mappings formalized by *Horn-Clauses* between relational tables and ontologies deriving from simple mappings. It is worthy of note that our approach can provide such initial mappings.

7 Summary and Future Work

In summary, the main contributions of this paper are listed as follows. Firstly, we have presented a new approach to discovering simple mappings between entities in a relational database schema and an ontology. It captures semantic information contained in the structures of the entities based on virtual documents, and eliminates incorrect mappings by validating mapping consistency.

Secondly, we have proposed a novel algorithm to construct contextual mappings. The algorithm reuse simple mappings and supplies additional sample instances for a relational database schemas and an ontology. Contextual mappings specify the conditions for converting to view-based mappings with selection conditions, which further help query answering and data integration.

Finally, we have experimentally evaluated our approach on several data sets from real world domains. The results demonstrate that our approach performs well as compared to some existing approaches in average F1-Measure. Besides, the results also show that the contextual mappings constructed by our approach are useful and meaningful.

In the future work, we look forward to comparing our approach with some intermediate approaches which firstly convert one data model to the other, and then reuse certain schema matching or ontology matching methods to discover simple mappings. We also hope to consider some machine learning techniques for mining some other interesting and useful semantic mappings. Finally, we would like to integrate our approach into some existing data integration tools in order to evaluate its effectiveness.

Acknowledgements

The work is supported in part by the NSFC under Grant 60573083, and in part by the 973 Program of China under Grant 2003CB317004. We thank Dongdong Zheng and Yuanyuan Zhao for their work in the experiments. We also appreciate anonymous reviewers for their precious comments.

References

1. An, Y., Borgida, A., Mylopoulos, J.: Inferring complex semantic mappings between relational tables and ontologies from simple correspndences. In Proceedings of International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'05). (2005) 1152–1169
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The description logic handbook: Theory, implementation and applications. Cambridge University Press. (2003)
3. Barrasa, J., Corcho, O., Gomez-Perez, A.: R2O, an extensible and semantically based database-to-ontology mapping language. In Proceedings of the 1st International Workshop on Semantic Web and Databases (SWDB'04). (2004)
4. Berners-Lee, T., Hall, W., Hendler, J., Shatbolt, N., Weitzner, D.J.: Creating a science of the Web. Science. **313** (2006) 769–771
5. Bohannon, P., Elnahrawy, E., Fan, W., Flaster, M.: Putting context into schema matching. In Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06). (2006) 307–318

6. Chang, K.C., He, B., Li, C., Zhang, Z.: Structured databases on the Web: Observation and implications. SIGMOD Record. **33(3)** (2004) 61–70
7. Chen, H., Wang, Y., Wang, H., Mao, Y., Tang, J., Zhou, C., Yin, A., Wu, Z.: Towards a Semantic Web of relational databases: A practical semantic toolkit and an in-use case from traditional Chinese medicine. In Proceedings of the 5th International Semantic Web Conference (ISWC'06). (2006) 750–763
8. Chen, H., Wu, Z., Wang, H., Mao, Y.: RDF/RDFS-based relational database integration. In Proceedings of the 22nd International Conference on Data Engineering (ICDE'06). (2006) 94
9. Chiang, R.H.L., Barron, T.M., Storey, V.C.: Reverse engineering of relational databases: Extraction of an EER model from a relational database. Data & Knowledge Engineering. **12** (1994) 107–142
10. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., Kolari, P.: Finding and ranking knowledge on the Semantic Web. In Proceedings of the 4th International Semantic Web Conference (ISWC'05). (2005) 156–170
11. Do, H.H., Rahm, E.: COMA – A system for flexible combination of schema matching approaches. In Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02). (2002) 610–621
12. Dou, D., LePendu, P., Kim, S., Qi, P.: Integrating databases into the Semantic Web through an ontology-based framework. In Proceedings of the 3rd International Workshop on Semantic Web and Databases (SWDB'06). (2006) 54
13. Dragut, E., Lawrence, R.: Composing mappings between schemas using a reference ontology. In Proceedings of International Conference on Ontologies, Databases and Applications of Semantics (ODBASE'04). (2004) 783–800
14. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition. **5(2)** (1993) 199–220
15. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: The state of the art. The Knowledge Engineering Review. **18(1)** (2003) 1–31
16. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. In Proceedings of the 16th International World Wide Web Conference (WWW'07). (2007)
17. Papapanagiotou, P., Katsiouli, P., Tsetsos, V., Anagnostopoulos, C., Hadjiefthymiades, S.: RONTO: Relational to ontology schema matching. AIS SIGSEMIS BULLETIN. **3(3-4)** (2006) 32–36
18. Patel-Schneider, P.F., Hayes, P., Horrocks, I. (eds.): OWL Web ontology language semantics and abstract syntax. W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-semantics/
19. Ramakrishnan, R., Gehrke, J.: Database management systems (3rd ed.). McGraw-Hill. (2002)
20. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In Proceedings of the 15th International World Wide Web Conference (WWW'06). (2006) 23–31
21. Rybiński, H.: On first-order-logic databases. ACM Transaction on Database Systems. **12(3)** (1987) 325–349
22. Salton, G., McGill, M.H.: Introduction to modern information retrieval. McGraw-Hill. (1983)
23. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. Journal on Data Semantics IV. (2005) 146–171
24. Stoilos, G., Stamou, G., Kollias, S.: A string metric for ontology alignment. In Proceedings of the 4th International Semantic Web Conference (ISWC'05). (2005) 623–637