Toward Detecting Mapping Strategies for Ontology Interoperability

Jie Tang
Department of Computer, Tsinghua University, P.R.China
Knowledge Engineering Group, Department of Computer, Tsinghua University, P.R.China, 100084
+86-010-62789831
j-tang02@mails.tsinghua.edu.cn

Bang-Yong Liang
Department of Computer, Tsinghua University, P.R.China
Knowledge Engineering Group, Department of Computer, Tsinghua University, P.R.China, 100084
+86-010-62789831
liangby97@mails.tsinghua.edu.cn

Juan-Zi Li
Department of Computer, Tsinghua University, P.R.China
Knowledge Engineering Group, Department of Computer, Tsinghua University, P.R.China, 100084
+86-010-62789831
ljz@keg.cs.tsinghua.edu.cn

ABSTRACT
Ontology mapping is one of the core tasks for ontology interoperability. It is aimed to find semantic relationships between entities (i.e. concept, attribute, and relation) of two ontologies. It benefits many applications, such as integration of ontology based web data sources, interoperability of agents or web services. To reduce the amount of users' effort as much as possible, (semi-) automatic ontology mapping is becoming more and more important to bring it into fruition. In the existing literature, many approaches have found considerable interest by combining several different similar/mapping strategies (namely multi-strategy based mapping). However, experiments show that the multi-strategy based mapping does not always outperform its single-strategy counterpart. In this paper, we mainly aim to deal with two problems: (1) for a new, unseen mapping task, should we select a multi-strategy based algorithm or just one single-strategy based algorithm? (2) if the task is suitable for multi-strategy, then how to select the strategies into the final combined scenario? We propose an approach of multiple strategies detections for ontology mapping. The results obtained so far show that multi-strategy detection improves on precision and recall significantly.

Categories and Subject Descriptors
D.2.12 [**SOFTWARE ENGINEERING**]: Interoperability *–Data mapping.*

General Terms
Algorithms, Measurement

Keywords
Ontology Interoperability, Ontology Mapping, Semantic Web, Multi-view detection

1. INTRODUCTION
Ontology, as the means for conceptualizing and structuring domain knowledge, has become the backbone to enable the fulfillment of the Semantic Web vision [4]. It aims to make data more sharable. However, ontologies themselves can be heterogeneous. Mapping between different ontologies thus becomes essential to ontology interoperability. Ontology mapping is the task of finding semantic relationships between entities (i.e. concept, attribute, and relation) of two ontologies. It benefits many applications, such as integration of ontology based web data sources, interoperability of agents or web services. Figure 1 shows an example of ontology mapping. The two ontologies O_1 and O_2 describe the domain of college course[1]. A reasonable mapping between the two ontologies is given in the figure by the dashed lines. Table 1 gives the details of the mapping.

Currently, ontology mapping is largely performed manually by domain experts, therefore a time-consuming, tedious and error-prone process. Hence, approaches for automating the ontology mapping tasks as much as possible are badly needed to simplify and speed up the development, maintenance and use of such applications. Numerous researchers have addressed the ontology mapping problem either for specific applications [6, 8, 9, 13, 14, 16, 19] or in a more generic way for different applications [15]. The proposed techniques for automating ontology mapping exploit various types of information in ontology, e.g. entity names, taxonomy structures, constraints, and characteristics of entities' instances. To achieve high accuracy for a large variety of ontologies, a single strategy (e.g., name based strategy) may be unlikely to be successful. Hence, to combine different approaches is an effective way. For this purpose, many approaches combining the results of several independently executed mapping algorithms are proposed [8, 9, 10, 16, 19]. However, in some cases, our experiments show that the composite approach does not always outperform the single strategy algorithm [18]. Consider, for instance, the course mapping problem (cf. figure 1), in which the three strategies consist of "name based strategy", "taxonomy based strategy" and "instance based strategy". In principle, one can use these three strategies in the mapping task to find relationship between entities of the two ontologies. But, it may happen that the entities' instances are so sparse and uninformative that one is better off by only using the combination of name based strategy and taxonomy based strategy.

In the previous work [26], we have proposed RiMOM to find mapping by combining strategies. In this paper, we propose an approach of multi-strategies detection into RiMOM to automatically detect the optimal composite of multiple strategies for a new mapping task. We call the new version of RiMOM as iRiMOM. We here focus on dealing with the following problems: (1) for a new, unseen mapping task, should we select a multi-strategy based algorithm or just one single-strategy based algorithm? (2) if the task is suitable for multi-strategy, then how to select the strategies into the final combined scenario?

WWW 2005, May 10--14, 2005, Chiba, Japan.

[1] http://anhai.cs.uiuc.edu/archive/summary.type.html

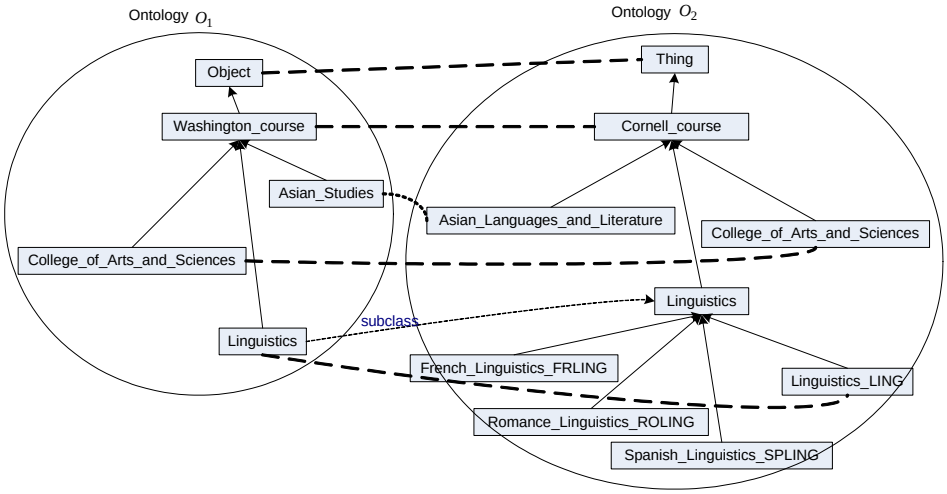

Figure 1. Example Ontogolies and their Mappings

Table 1. Mapping table for O_1 and O_2

Ontology O_1	Ontology O_2
Object	Thing
Washington_course	Cornell_course
Asian_Studies	Asian_Languages_and_Literature
College_of_Arts_and_Sciences	College_of_Arts_and_Sciences
Linguistics	Linguistics_LING

Content of this paper is structured as following. Section 2 clarifies the related terminologies. Section 3 describes the mapping process and an overview of current mapping strategies in iRiMOM. Section 4 presents the multi-strategy detection for mapping discovery. The evaluation and experiment are given in Section 5. Finally, before conclude the paper with a discussion, we gives the survey of the related work.

2. Terminology

This section introduces the basic definition in the mapping process/algorithms and familiarizes the reader with the notations and terminology used throughout the paper.

2.1 Ontology

The underlying data models in our process are ontologies. To facilitate the further description, we briefly summarize their major primitives and introduce some shorthand notations. Our ontologies are built on OWL[2]. Ontology can be defined by a six tupel [10]:

$$O = \{C, Attr, H^C, REL, A^O, I\}$$

An ontology O is defined by a set of concepts C (instances of "owl:Class") and a corresponding hierarchy $H^C \in C \times C$ (instances of "rdfs:subClassOf", each of which denotes a directed relation called concept hierarchy or taxonomy, e.g. $H^C(c_i, c_j)$ means that concept c_i is the sub-concept of c_j). *Attr* is a set of data properties for concepts C (instances of "owl:DatatypeProperty"). Relations $REL \in C \times C$ (instances of "owl:ObjectProperty") is a set of non-taxonomical relations between concepts. Axioms A^O, expressed in a logical language, can be used to infer knowledge from existing one. I denotes a collection containing all instances of concepts C.

[2] http://www.w3.org/

For a concept $c \in C$, we define $I_c \subset I$ as the set of its instances. Let i_c be an instance of c, i.e. $i_c \in I_c$. Let j denote a value of c's data property $attr \in Attr$ or its object property $rel \in REL$. We call the triple (i_c, *attr*, j) a data property instance and triple (i_c, *rel*, j) a object property instance, respectively.

2.2 Heterogeneity of Ontology

In order to reach interoperability over heterogeneous ontologies, two problems must be dealt with, i.e.: *metadata heterogeneity* and *instance heterogeneity* [12, 14]. Metadata heterogeneity concerns the intended meaning of described information. There are two kinds of conflicts in metadata heterogeneity: *structure conflict*, which means that ontologies for same domain knowledge may have different semantic structures; and *name conflict*, which means that the same concept may use different names or different concepts may use the same name.

Figure 1 shows some examples of above problems. The concept *Asian_Studies* in Ontology O_1 has the same meaning as concept *Asian_Lanugages_and_Literature* in ontology O_2, though they have different name. But the concepts *Linguistics* in O_1 and O_2, though have the same name, do represent slightly different meanings. We can also see the structure conflict from this example. In O_1, concept *Linguistics* is a single concept, while in O_2, it has four sub concepts.

Instance heterogeneity concerns the different representations of instances. Information described by the same ontology can be represented in different ways, also called representation conflict. For example, date can be represented as "*2004/2/27*" or "*Feb, 27, 2004*"; person name can be represented as "*Jackson Michael*" and "*Michael, Jackson*", etc. Representation conflict requires normalization before ontology interoperation.

2.3 Ontology Mapping

Ontology mapping takes two ontologies as input and creates a semantic correspondence between the entities in the two input ontologies [23]. Due to the wide range of expressions used in this area (merging, alignment, integration etc.), we adopt the following definition for the term "mapping": Given two ontologies O_1 and O_2, mapping from ontology O_1 to another O_2 means for each entity in ontology O_1, we try to find a corresponding entity, which has the same intended meaning, in ontology O_2 [1]. Ontology O_1 is called source ontology and O_2 is

called target ontology. We also call the process of finding mapping from O_1 to O_2 as (Ontology) mapping discovery or mapping prediction.

Formally an ontology mapping function can be defined by the following way:

$$Map(\{e_{i1}\},\{e_{i2}\},O_1,O_2) = f\ ,$$

with $e_{i1} \in O_1$, $e_{i2} \in O_2$:$\{e_{i1}\} \xrightarrow{f} \{e_{i2}\}$. $\{e_{i1}\}$ denotes a collection of entities, $e_{i1} \in C \cup Attr \cup REL$. f can be one of the mapping types (e.g. equivalentClass, subclass, sameIndividualAs, unionOf, disjointWith, etc.) or null. When equivalent mapping is the only concern, we usually leave out O_1 and O_2 and write as $Map(\{e_{i1}\},\{e_{i2}\})$. We use the notation $Map(O_1,O_2)$ to indicate all entity mappings from O_1 to O_2.

Once a mapping $Map(\{e_{i1}\},\{e_{i2}\})$, between two ontologies O_1 to O_2 is established, we also say that "entities $\{e_{i1}\}$ is mapped onto entities $\{e_{i2}\}$". For each pair of entity sets $(\{e_{i1}\},\{e_{i2}\})$, we call it *candidate mapping*. An entity set can be mapped to at most one other entity set.

In this paper, we only consider the 1:1 mappings between single entities. However, we do not consider its further use, e.g. for knowledge reasoning or query answering over multiple heterogeneous ontologies and translation of ontology instances (cf. [16]) or theory approximation (cf. [25]).

2.4 Multi-strategy Detection in Ontology Mapping

To achieve high accuracy for a large variety of ontologies, a single strategy (e.g., name based strategy) may be unlikely to be successful. Multiple strategies method proves useful in ontology mapping [8, 9, 10, 16, 19]. As for multiple strategies, there are two kinds of methods: hybrid and composite combination [8]. Hybrid approach is the most common where different mapping criteria or properties (e.g. name and data type) are used within a single algorithm. Typically these criteria are fixed and used in a specific way. By contrast, a composite mapping approach combines the results of several independently executed mapping algorithms, which can be simple of hybrid. This allows for a high flexibility, as there is the potential for selecting the mapping algorithms to be executed based on the mapping task at hand. In this paper, we exploit the later to combine multi-strategy in ontology mapping.

We define a strategy as S, and define multi-strategy as a collection: $\{S_j\}$. The notation $<Map_j(e_{i1},e_{i2}), S_j>$ denotes a mapping discovered by S_j for e_{i1} and e_{i2}. $<Map_{\{j\}}(e_{i1},e_{i2}), \{S_j\}>$ denotes the composite mapping determined by multiple strategies $\{S_j\}$ for e_{i1} and e_{i2}. Given two ontologies O_1, O_2 and multi-strategy algorithms $\{S_j\}$, the target of multi-strategy detection is to detect the most appropriate strategy combination so as to result into mappings with minimum error.

$$Map^*(O_1,O_2) = \arg_{Map} \min Err(Map_{\{j\}} \mid \{S_j\}) \ldots (1)$$

$Err(Map_{\{j\}}(e_{i1},e_{i2})|\{S_j\})$ is the mapping error by using $\{S_j\}$.

3. Multiple Strategies in Ontology Mapping

In this section, we first briefly introduce the mapping process in iRiMOM, and then illustrate the strategies that are exploited to determine mappings.

3.1 Process

Figure 2 illustrates mapping process in iRiMOM with two input ontologies, one of which are going to be mapped onto the other. It consists of six main steps:

1. User Interaction (optional). iRiMOM supports a optional user interaction phase to capture information provided by user including: mapping correction, creation of new mapping. The user's specified mappings will influence the similarity computations for the neighborhood of the respective entities and thus can improve the mapping accuracy of structural algorithms.
2. Multi-strategy execution. A main step during a mapping iteration is the execution of multiple independent mapping algorithms. Each algorithm determines similarity values between a candidate mappings (e_{i1},e_{i2}) based on their definitions in O_1 and O_2, respectively. The information exploited to determine the similarity value includes: entity names, data types, entity path and textual content of instances, etc. Each mapping algorithm determines an intermediate mapping result according to the similarity value between 0 and 1 for each possible candidate mapping. The result of the mapping execution phase with k algorithms, m entities in O_1 and n entities in O_2 is a k x m x n cube of similarity values, which is stored in the repository for later strategies detection and combination steps.
3. Multi-strategy detection. This step selects the most 'promising' strategies for the final combination so as to result into optimal mappings.
4. Strategies combination. By multiple mapping algorithms, there are several similarity values for a candidate mapping (e_{i1},e_{i2}). For example, one is the similarity of their name and another one is the similarity of their instances. This step is to derive the combined mapping result from the individual algorithm results stored in the similarity cube. For each combination of ontology entities the algorithm-specific similarity values are aggregated into a combined similarity value, e.g. by taking the average or maximum value.
5. Mapping discovery. This step uses the individual or combined similarity values to derive mappings between entities from O_1 to O_2. Some mechanisms here are, using thresholds or maximum values for similarity mappings, performing relaxation labeling [6], or other criteria.
6. Iteration. Mapping process takes place in either one or multiple iterations depending on whether an automatic or interactive determination of mapping candidates is to be performed. In interactive mode, the user can interact with iRiMOM in each iteration to specify the mapping strategies (selection of mapping strategies), correct mistake mapping, and create new mapping from the previous iteration. In automatic mode, several algorithms perform an iteration over the whole process in order to bootstrap the amount of structural knowledge. Each iteration subsumes two sub-iteration: the first is to discover concept mapping, the later is to discover attributes/relations mapping for the concept mapping. Iteration stops when no new mapping are discovered.

Eventually, the output is a mapping table (includes multiple entries of $Map(e_{i1},e_{i2})$ from O_1 to O_2.

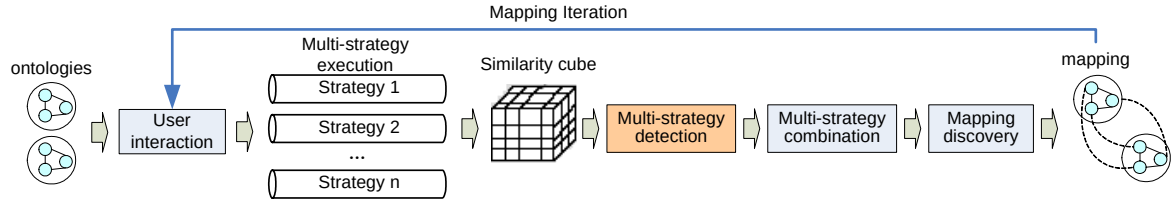

Figure 2. Mapping process in iRiMOM

3.2 Multiple strategies for ontology mapping

In order to discover a mapping $Map(O_1,O_2)$ from O_1 to O_2, many different information can be exploited, e.g. entity names, entity description, taxonomy structure, constraints as well as textual content of annotated instances.

3.2.1 Instances based strategy

This strategy exploits the frequencies of words in the textual content of instances to discover mapping. Recall that an instance typically has a name and a set of attributes together with their values. We treat them and their values as the textual content of the concept's instance. The other textual content of the instance is the text content it related to. For example, the textual content of the instance *CHIN-101-102-Elementary-Standard-Chinese* of concept *Chinese* can be the web page that it related to.

The instance based strategy employs Naïve Bayesian classifier. Firstly, input instances of O_2 as training samples for Bayesian classifier, and concepts in the ontology are regarded as classes for the classifier, then decision is made based on the instances of O_1. It is defined as:

$$p(e_{i2} \mid I_{i1}) = p(I_{i1} \mid e_{i2})\, p(e_{i2}) / p(I_{i1})$$

where $I_{i1}=\{i_{i1k}\}$ denotes a set of instances of e_{i1}. The textual content of each instance is treated as a bag of tokens, which are generated by tokenizing, stop-word removal and word stemming on the textual content. Let $i_{i1k}=\{w\}$ be the content of an input instance where w is a token. $p(I_{i1})$ can be ignored because it is just a constant. $p(e_{i2})$ is estimated as the probability of training instances that belong to e_{i2}. To compute $p(I_{i1}|e_{i2})$, we make the assumption that tokens appear in instances I_{i1} independently of each other for the given e_{i2}. Thus $p(I_{i1}|e_{i2})$ can be computed by $p(I_{i1} \mid e_{i2}) = \prod_{w \in I_{i1}} p(w \mid e_{i1})$. Therefore, $p(e_{i2}|I_{i1})$ can be rewritten as:

$$p(e_{i2} \mid I_{i1}) = \prod_{w \in I_{i1}} p(w \mid e_{i2}) \bullet p(e_{i2})$$

$p(w|e_{i2})$ is estimated as $n(w,e_{i2})/n(e_{i2})$, where $n(e_{i2})$ is the number of tokens occurring in the training instances of e_{i2}, and $n(w,e_{i2})$ is the times that token w appears in the training instances of e_{i2}.

For each possible mapping $Map(e_{i1},e_{i2})$ for e_{i1}, thus, the strategy computes the probability $p(e_{i2}|I_{i1})$, and predicts the mapping by $\arg\max_{e_{i2}} p(e_{i2} \mid I_{i1})$.

Instance based decision works well on long or distinct contents, but is less effective on short and indistinct contents.

3.2.2 Name based strategy

In the efforts using entity name to find mapping, some use VSM (Vector Similarity Model) by casting the decision as an information retrieval problem [15], while some others use classifier to make prediction [9]. However, because of the name conflicts, the former always results in unsatisfactory result. Furthermore, as classifier is less efficient on short text content, the later cannot work well either. We approach name decision by combining semantic dictionary with statistic technique.

Firstly, the similar measurement between words w_1 and w_2 is defined as:

$$sim(w_1, w_2) = (sim_d(w_1, w_2) + sim_s(w_1, w_2))/2$$

Where: $sim_d(w_1,w_2)$ denotes the meaning similarity between w_1 and w_2 by semantic dictionary. In this paper, it is computed through Wordnet, one of the most popular semantic dictionaries. $sim_s(w_1,w_2)$ is the similarity by statistical technique.

Wordnet is a semantic network about words, in which each node is a synset. Each synset may contain multiple words with similar meaning and each word may exist in multiple synset indicating that the word has multiple senses. Lin defines the similarity between two senses as [21]:

$$sim_d(s_1, s_2) = \frac{2 \times \log p(s)}{\log p(s_1) + \log p(s_2)}$$

where, $p(s) = count(s)/total$, is the probability that a randomly selected word occurs in synset s or any synsets below it. *total* is the number of word in Wordnet and $count(s)$ is the word count in s and any synsets below it. s is the common hypernym of s_1 and s_2. Let $s(w_1) = \{s_{1i} \mid i=1,2,\ldots,m\}$ and $s(w_2) = \{s_{2i} \mid i=1,2,\ldots,n\}$ are senses of w_1 and w_2 respectively. We define $sim_d(w_1,w_2)$ as:

$$sim_d(w_1, w_2) = \max(sim_d(s_{1i}, s_{2j})) \qquad s_{1i} \in s(w_1), s_{2j} \in s(w_2)$$

By the use of statistic technique, Lin constructs a thesaurus, in which similarities between words are listed. The value of $sim_s(w_1,w_2)$ for each pairwise-word can be obtained by directly looking up the thesaurus.

To compute the similarity of names, it is necessary to perform some pre-processing steps, in particular the tokenization to derive a bag of tokens. Moreover, it is required to expand abbreviations and acronyms, e.g. "*CS*" results in {Computer, Science}. The name based strategy then computes the similarity matrix for the two token sets. Each value in the matrix denotes the similarity of each pairwise-word. Formally, by inputting two entity names, $name_1$ and $name_2$, they are pre-processed into two token sets $\{w_i\}$ and $\{w_j\}$. Then for each w_i, we select the highest similarity $sim(w_1,w_2)$ as the similarity between w_i and $name_2$, i.e. $sim(w_i, name_2)$. Finally, the similarity of $name_1$ and $name_2$ is defined as:

$$sim(name_1, name_2) = \sum_{i=1\ldots n} sim(w_i, name_2)/n$$

where n is the word count in $name_1$.

3.2.3 Entities' description based strategy

Description, expressed by natural language, is valuable information for entity in ontology. Typically, it reflects the

semantic of the entity more than entity name itself. The words in entity descriptions of target ontology can be exploited to train the Bayesian classifier and the entity descriptions of source ontology can be used for prediction. The principle of this decision is the same as that in instance based decision.

3.2.4 Name path based strategy

This strategy exploits the hierarchical names of the entities, i.e. both structural aspects and entity name are considered. It first builds a long name by concatenating all names of the entities in a path to a single string. It then applies the method in name based strategy to compute the similarity between these two long names. The name path of an entity provides additional tokens for name similarity which may improve the mapping accuracy.

3.2.5 Taxonomy context based strategy

Taxonomy structure describes the taxonomy context for the entity in the ontology. Taxonomy context based strategy is derived from the intuition that entities occurring in the similar contexts tend to be matchable, e.g. "two concepts match if their sub-classes match". A concept's taxonomy context includes its super class, subclasses, properties and relations. A property's taxonomy context includes its concept, super property, sub properties, sibling properties and its constraints. Thus, the taxonomy context similarity of two entities is defined by the combined similarity between the entities in their context. The combined similarity is obtained from other strategies, i.e. the strategies mentioned above.

3.2.6 Constraints based strategy

Constraints, which are often used to restrict concept and properties in ontology, are very useful. In terms of such constrains, we define heuristic rules to refine the learned mappings. Examples of such rules are: "*datatypeproperty* with *range* 'Date' can only be mapped to the *datatypeproperty* with *range* 'Date'->confidence: 1.0"; "*datatypeproperty* with *range* 'float' may be mapped to one with *range* 'string'->confidence: 0.6", etc. Each constraint is assigned with a confidence (e.g. 1.0 and 0.6) to extend the traditional Boolean constrain (i.e. yes or no). The confidences are specified manually.

3.2.7 Combination of multiple strategies

Outputs of the strategies need to be combined. In this paper, multiple strategies are combined by:

$$Map(e_{i_1}, e_{i_2}) = \sum_{k=1...n} w_k \sigma(Map_k(e_{i_1}, e_{i_2})) \Big/ \sum_{k=1...n} w_k$$

where w_k is the weight for individual strategy, and σ is a sigmoid function, which transforms the original similarity value (from [0, 1] to [0, 1]). Sigmoid function makes the combination emphasize high individual similarities and de-emphasize low individual similarities. σ is defined as: $\sigma(x) = 1/(1+e^{-5(x-\alpha)})$, where x is the individual similarity/predicting value, α is tentatively set as 0.5.

4. Multi-Strategy Detection for Mapping Discovery

In this section we describe how to detect the strategies when discover mapping for specified ontologies.

4.1 Multi-Strategy Detection

The key idea behind multi-strategy detection is based on the observation: the higher different the results obtained by the strategies, the lower probability the combined results outperform the single one. Figure 4 illustrates the relationship between difference of the two strategies and applicability of them [18].

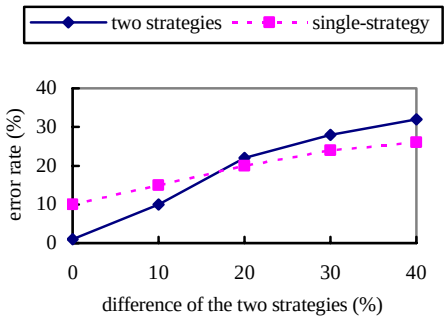

Figure 4. the relationship between the difference of the strategies and applicability of the two strategies.

Multi-strategy detection exploits this observation. We now explain how to detect optimal combination of multiple strategies. According to the equation (1), the best combination of multiple strategies is the one that results into mapping $Map^*(O_1,O_2)$ with minimum error $Err(Map_{\{j\}}|\{S_j\})$ from ontology O_1 to O_2. We first define the mapping error for strategies $\{S_j\}$:

$$\begin{aligned} Err(Map_{\{j\}} | \{S_j\}) &= \sum_j Err(Map_j | S_j, Map_{\{j\}}) \\ &= \sum_j (1 - p(Map_j | S_j, Map_{\{j\}})) \end{aligned} \quad \text{... (2)}$$

where $Map_{\{j\}}$ denotes $Map_{\{j\}}(O_1,O_2)$ that obtained by combination of the multiple strategies $\{S_j\}$, and Map_j is the mapping obtained by single strategy S_j. $Err(Map_j|S_j, Map_{\{j\}})$ denotes the mapping error of strategy S_j. It is quantified by using $1\text{-}p(Map_j|S_j, Map_{\{j\}})$. $p(Map_j|S_j, Map_{\{j\}})$ captures how much the mapping Map_j is consistent with the mapping $Map_{\{j\}}$. Assuming that the entities' mappings are independent of each other for the given S_j, we have

$$\begin{aligned} & p(Map_j | S_j, Map_{\{j\}}) \\ &= \prod_{e_{i1} \to e_{i2}} p(Map_j(e_{i1}, e_{i2}) | S_j, Map_{\{j\}}(e_{i1}, e_{i2})) \end{aligned} \quad (3)$$

where $e_{i1} \to e_{i2}$ means one discovered mapping.

$p(Map_j(e_{i1},e_{i2})|S_j, Map_{\{j\}}(e_{i1},e_{i2}))$ is the probability of difference for (e_{i1},e_{i2}) between combined mapping and S_j's mapping. For short, it is rewritten as $p((e_{i1},e_{i2})|S_j, Map_{\{j\}})$. It is estimated by the degree of difference between S_j's score and the combined score, since each mapping has a score in the automatic mapping scenario, e.g. S_j's score on mapping (e_{i1}, e_{i2}) is 0.5 and combined score is 0.6, then $p((e_{i1},e_{i2})|S_j, Map_{\{j\}})=(0.6\text{-}0.5)/0.6= 0.167$.

Thus, by substituting equation (3) into equation (2), we obtain

$$\begin{aligned} & Err(Map_{\{j\}} | \{S_j\}) \\ &= \sum_j (1 - \prod_{e_{i1} \to e_{i2}} p(Map_j(e_{i1}, e_{i2}) | S_j, Map_{\{j\}}(e_{i1}, e_{i2}))) \end{aligned} \quad (4)$$

4.2 Efficient Implementation of Multi-strategy Detection

Equation (4) captures the error rate for each possible combination of the strategies. A naive implementation of the multi-strategy detection algorithm would enumerate all possible combination of

the strategies. And then select the combination with minimum error. For n strategies, this enumeration will up to $O(n!)$.

This naive implementation does not work in practice because of the vast computation. To make it efficient, we developed a rapid algorithm. The basic idea is based on the hypothesis that the higher error the strategy S_j is, the lower probability of S_j being selected in the optimal strategies. Thus, if we can estimate the error of each strategy, the optimal strategies can be obtained by enumerating the strategies by linear searching times. In this way, the task of detection can be accomplished by the following steps: first compute the mapping by all strategies, secondly remove the strategy with highest error, and then remove the strategy with second highest error, and so on. Then the problem is to compute error of each strategy S_j. We define error of S_j as

$$Err(S_j) = Err(Map|\{S\}) - Err(Map_{\{\bar{j}\}}|\{S_{\bar{j}}\}) \quad \ldots (5)$$

where $Err(Map|\{S\})$ is the mapping error by all strategies. $Err(Map_{\{\bar{j}\}}|\{S_{\bar{j}}\})$ is the mapping error by all strategies without S_j. Intuition of equation (5) is that S_j's error is the error reduction when remove it from the strategies combination. Now, for n strategies, enumeration in the detection is only n times.

5. Experiments and Discussions

5.1 Experiment Setup

1. Evaluation We use standard information retrieval metrics to evaluate our method and to compare with other methods.

Precision(*P*) It measures the number of correct discovered mappings versus the total number of discovered mappings.

Recall(*R*) It describes the number of correct mappings found in comparison to the total number of existing mappings.

$$P = \frac{|m_a \cap m_m|}{|m_a|}, \quad R = \frac{|m_m \cap m_a|}{|m_m|}$$

where: m_a are mappings discovered by iRiMOM and m_m are mapping assigned manually.

2. Data Sets We evaluated iRiMOM on three data sets[1], whose characteristics are shown in table 2 [9].

The ontology Course Catalog Ⅰ and Ⅱ describe courses at Cornell University and the University of Washington. The ontologies of Course Catalog Ⅰ have 34-39 concepts, and are fairly similar to each other. The ontologies of Course Catalog Ⅱ are larger and less similar to each other.

Table 2. Ontologies in experiments

Ontologies		concepts	instances number	manual mapping
Course Catalog Ⅰ	*Cornell*	34	1526	34
	Washington	39	1912	37
Course Catalog Ⅱ	*Cornell*	176	4360	54
	Washington	166	6957	50
Company Profiles	*Standard.com*	333	13634	236
	Yahoo.com	115	9504	104

The company Profile uses ontologies from Yahoo.com and The Standard.com and describes the current business status of companies.

3. Experiments setup. We took the RiMOM [7] as the baseline method to test the effect of strategies detection. As well, we compared the result with GLUE, one of popular methods to find mapping between ontologies by using machine learning. Results of GLUE come from [9], where possible we used the same data sets and the same evaluation metrics.

- RiMOM, a prior version of iRiMOM [26]. It exploits all possible strategies to discover mappings.
- GLUE. It uses instance, entities' name and relaxation labeler [6] to find mappings. For each strategy, it defines a measure, called *joint probability distribution*, to estimate the similarity between any two entities.
- iRiMOM, the enhanced version of RiMOM. iRiMOM focuses on strategies detection so as to obtain the optimal mappings. Because these three data sets don't have the entities' descriptions and constraint definitions, we made use of the four other strategies as the candidate strategies for detection, including instance based, name based, name path based, and taxonomy based strategies.

5.2 Experiment Results

Table 3 shows the comparison between RiMOM and iRiMOM using precision and recall as the evaluation metrics. In the table: $\{S\}$ denotes the detected strategies, where N—name based strategy, P—name path based strategy, I—instance based strategy, T—taxonomy context based strategy.

Table 3. Experimental comparison between RiMOM and iRiMOM

Data set	mapping	RiMOM		iRiMOM		
		P	*R*	*P*	*R*	{*S*}
Course Catalog Ⅰ	*Cornell to Wash.*	88.2	88.2	97.1	97.1	NP
	Wash. To Cornell	92.1	94.6	94.7	97.3	PIT
Course Catalog Ⅱ	*Cornell to Wash.*	78.3	87.0	83.9	96.3	NIT
	Wash. To Cornell	75.4	94.0	81.5	96.0	PIT
Company Profiles	*Standard to Yahoo*	81.0	85.0	82.3	91.2	NIT
	Yahoo to Standard	71.4	89.5	71.4	89.5	NPIT

On five mapping tasks of the three data sets, iRiMOM outperforms RiMOM (vary from +1.3% to +8.9% on precision and from +2.7% to +9.3% on recall).

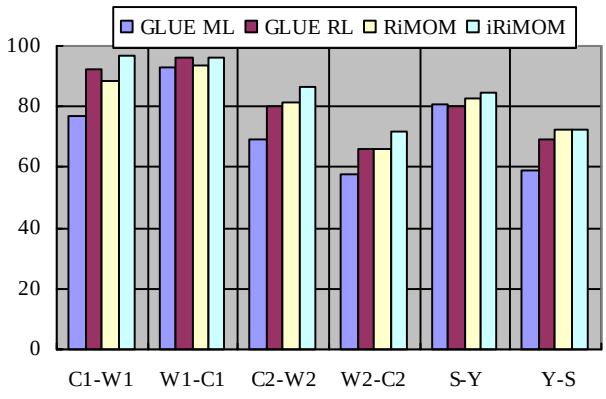

Figure 5. Comparison of mapping accuracy

Figure 5 shows the comparison between GLUE and iRiMOM. Because the meta learner and relaxation labeler of GLUE are the

two most promising methods. We compared the results of them with RiMOM and iRiMOM.

In figure 5, GLUE ML denotes meta learner of GLUE and GLUE RL denotes relaxation labeler of GLUE. C1 and W1 indicate *Cornell* and *Washington* in Course Catalog Ⅰ. C2 and W2 indicate *Cornell* and *Washington* in Course Catalog Ⅱ. S and Y indicate *Standard* and *Yahoo* in Company Profile. On most mapping tasks, based on the strategies detection, iRiMOM outperforms GLUE.

5.3 Discussions

On the three data sets, iRiMOM outperforms both RiMOM and GLUE. We here focus on the analysis of the experiments results.

First, strategies detection is efficient in ontology mapping. In five of the six mapping tasks, strategies detection improves both the precision and recall of mapping. Experiments also show that the method of multi-strategy is useful in ontology mapping. In all six mapping tasks, the selected strategies are composite of at least two strategies.

Second, taxonomy and instance based strategies are especially useful. Note that they are almost selected in each task. Taxonomy based strategy can also be regarded as the post-process of other strategies. It makes use of the information from other strategies and domain knowledge, thus can correct some error mappings. But in some cases, it may not work, such as in the mapping task from *Cornell* to *Washington* in data set Course Catalog Ⅰ.

Finally, in RiMOM and iRiMOM, the name based strategy and name path based strategy are quite different from that in GLUE. In GLUE, the name based strategy is similar to its instance based strategy except that it makes predictions using the full name of the input instance instead of its content. Our name based strategy exploits semantic thesaurus and statistic technique to estimate the name similarity. It is very efficient, in particular to find the mapping between semantic similar entities, such as *Asian-Studies* and *Asian-Languages-and-Literature*, or *Earth-and-Atmospheric-Sciences* and *Earth-and-Space-Sciences*. Experiments show that name and name path based strategy can be even qualified alone in some mapping tasks, for example the mapping from *Cornell* to *Washington*.

6. Related Works

In this section, we review the research efforts related to our work. First, we introduce existing works on ontology mapping. Then we present the related approaches to multi-strategy detection.

6.1 Ontology Mapping

Most previous research efforts on ontology integration have used ad-hoc mapping rules between ontologies (as surveyed in [23, 28]). This approach allows flexibility in ontology integration, but most works do not provide automatic mapping. [6] proposes an Ontology Integration Framework, which provides clear semantics for ontology integration by defining sound and complete semantic conditions for each mapping rule. [22] describes an extension to Protégé to map between ontologies. In this method, a valuable set of desiderata and mapping dimensions are defined, however its implementation lacks some important features especially in allowing multiple concepts mapping or functional mapping.

Also some researchers are working on automatic method of mapping (as survey in [23]). GLUE exploits joint distribution of the annotated instances, element name and taxonomy structure to learn the mapping rules [9]. Chimaera [17], PROMPT [19] and RDFT [20] all support the automatic mapping or merging of ontological terms i.e. class and attribute names from various sources. These algorithms exploit taxonomy structures and textual names to looks for an exact match in class names or a match on prefixes, suffixes, and word root of class names. Some other works also provides the function of semi-automatic mapping [8, 13, 15]. [7, 29] focus on complex mapping discovery. They exploits beam search and equation discovery to mine the complex text mapping and numeric function mapping. QOM focuses on the efficiency of mapping [10].

To the best of our knowledge, no previous work has been done exactly on the mapping strategies detection.

6.2 Multi-Strategy Detection

Multi-strategy detection is similar to the method of "learning to learn" [27]. A typical such algorithm is given a learning task T and a number of learning algorithms l_1, l_2, … , l_n, and it is required to predict which of the n learners obtains the best accuracy on the task T. There are four main approaches to deal with this problem: experimental, knowledge-driven, transfer of learning and meta-learning.

The main idea in experimental strategy detection is based on the assumption: if an algorithm trained on a subset of the training data makes accurate predictions, it is likely to also make high accuracy predictions when trained on the entire dataset [18]. A typical approach is to use cross-validation to estimate the accuracy of each learner on the task T, and then the system solves the task T by applying the learner l_w that obtains the highest accuracy during cross-validation. But this approach needs training data, i.e. previously specified mappings between ontologies. However, here we assume that such training mappings are not available.

The knowledge-driven approach is to select the most appropriate algorithms for a particular task by a set of heuristic rules [5]. Its disadvantage includes: the tune for the number of parameters in the heuristics; adjustment of the heuristics for a particular task.

Transfer of learning refers to finding a model that is appropriate for a group of related tasks. Its main intuition is: when solving several tasks in the same environment, one can exploit the information from all these tasks to find the model that is most appropriate for the entire class of the problems [2]. The main problem with transfer of learning is that it needs training data and it can't be applied to new, unseen tasks.

Meta-learning is the process of learning how to guide a user in applying machine learning techniques. Meta-learning is used to increase the efficiency of the learning process by capturing how different factors influence the success or failure for a particular learner [18]. Meta-learning is used for variety of tasks such as feature selection [12], discriminating between a base learner's correct and incorrect predictions [1], finding the best way to deal with missing attributes [11]. But similar to experimental approach, meta-learning also needs adequate training data.

7. Conclusions

Ontology mapping is one of the main challenges for semantic web [3]. This paper introduces mapping strategies detection into ontology interoperability for the first time, and proposes an approach, called iRiMOM, to deal with the problem. Experiments show that by using strategies detection, iRiMOM improves on precision and recall by +8.9% and +9.3%, respectively.

There are many potential applications of ontology mapping on the semantic web including question answer, data mediation and device interoperability. Ontology mapping supports question answer over distributed ontologies.

In the future work, we intend to continue the work along several main directions: (1) to introduce active learning into iRiMOM. (2) Discovery of complex mapping.

8. REFERENCES

[1] S.D. Bay and M. Pazzani. Characterizing model errors and differences. In Proceedings of the 17th International Conference of Machine Learning (ICML-2000), 2000, pp:49–56.

[2] J. Baxter. Learning model bias. In Advances in Neural Information Processing Systems, 1996,v(9), pp:169–175.

[3] R. Benjamins, J. Contreras. White Paper Six Challenges for the Semantic Web. Intelligent Software Components. Intelligent software for the networked economy (isoco). April, 2002.

[4] T. Berners-Lee, M. Fischetti, and M. L. Dertouzos. Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. 1999.

[5] CE. Brodley. Recursive automatic bias selection for classifier construction. Machine Learning, 1995,20:63–94.

[6] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In the Emerging Semantic Web. IOS Press. 2002. 201–214.

[7] R. Dhamankar, Y. Lee, and A.H. Doan. iMAP: Discovering Complex Semantic Matches between Database Schemas. Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, 2004. Paris, France: ACM Press.

[8] H. Do and E. Rahm. Coma: A system for flexible combination of schema matching approaches. In Proc. of VLDB-2002.

[9] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In Proceedings of the World-Wide Web Conference (WWW-2002), pages 662–673. ACM Press, 2002.

[10] M. Ehrig and S. Staab. QOM – Quick Ontology Mapping. In Proc. of the third ISWC. Japan. 2004

[11] J. P. Feng. Meta-CN4 for unknown values processing via combiner and stacked generalization. In I. Buha and A. Famili, editors, KDD-2000 Workshop on Postprocessing in machine learning and data mining. 2000

[12] N. Kamolvilassatian. Property-based feature engineering and selection. Master's thesis, Department of Computer Sciences, University of Texas at Austin. 2002

[13] J. Kang and J. Naughton. On schema matching with opaque column names and data values. In Proc. of SIGMOD-2003.

[14] W. Kim and J. Seo. Classifying schematic and data heterogeneity in multidatabase systems. *IEEE* Computer, 1991, 24(12):12-18

[15] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In Proc. of VLDB-2001.

[16] A. Maedche, B. Moltik, N. Silva and R. Volz. MAFRA — An Ontology MApping FRAmework in the Context of the Semantic Web. In Proceeding of the EKAW'2002, Siguenza, Spain. 2002.

[17] D. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning. Colorado, USA.

[18] I. Muslea. Active learning with multiple views. Ph.D. dissertation. (USC, 2002)

[19] N. F. Noy and M. A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies, 2003,59(6):983-1024.

[20] B. Omelayenko. RDFT: A Mapping Meta-Ontology for Business Integration; Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at ECAI'2002. Lyon, France; 2002:76-83

[21] P. Pantel, D. Lin. Discovering Word Senses from Text. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2002:613-619.

[22] J. Y. Park, J. H. Gennari, and M. A. Musen. Mappings for Reuse in Knowledge-based Systems. 11th Workshop on Knowledge Acquisition, Modelling and Management (KAW 98); Banff, Canada; 1998.

[23] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. The VLDB Journal, 10:334–350, 2001.

[24] N. Silva and J. Rocha. Semantic Web Complex Ontology Mapping. IEEE/WIC International Conference on Web Intelligence (WI'03) October 13-17, 2003 Halifax, Canada:82-100

[25] X. Su. A text categorization perspective for ontology mapping. Technical report, Department of Computer and Information Science, Norwegian University of Science and Technology, Norway, 2002.

[26] J. Tang, B.Y. Liang, J.Z. Li, and K.H. Wang. Risk Minimization based Ontology Mapping. AWCC 2004.

[27] S. Thrun and L. Pratt, editors. Learning to learn. Kluwer Academic Publishers. 1997

[28] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information – a survey of existing approaches. In Proc. of IJCAI 2001 Workshop on Ontologies and Information Sharing. 2001

[29] L. Xu and D. Embley. Using domain ontologies to discover direct and indirect matches for schema elements. In Proc. of the Semantic Integration Workshop at ISWC-2003.