Machine Learning and Constraint Programming for Relational-To-Ontology Schema Mapping

Diego De Uña[1], **Nataliia Rümmele**[2], **Graeme Gange**[1], **Peter Schachte**[1] and **Peter J. Stuckey**[1,3]
[1]Department of Computing and Information Systems — The University of Melbourne
[2]Siemens, Germany
[3]Data61, CSIRO, Melbourne, Australia
d.deunagomez@student.unimelb.edu.au, nataliia.ruemmele@siemens.com
{gkgange,schachte,pstuckey}@unimelb.edu.au

Abstract

The problem of integrating heterogeneous data sources into an ontology is highly relevant in the database field. Several techniques exist to approach the problem, but side constraints on the data cannot be easily implemented and thus the results may be inconsistent. In this paper we improve previous work by Taheriyan *et al.* [2016a] using Machine Learning (ML) to take into account inconsistencies in the data (unmatchable attributes) and encode the problem as a variation of the Steiner Tree, for which we use work by De Uña *et al.* [2016] in Constraint Programming (CP). Combining ML and CP achieves state-of-the-art precision, recall and speed, and provides a more flexible framework for variations of the problem.

1 Introduction

Relational data sources are still one of the most popular ways to store enterprise or Web data. However, the issue with relational schema is the lack of a well-defined semantic description. A common ontology provides a way of representing the meaning of a relational schema and can facilitate the integration of heterogeneous data sources. Indicating semantic correspondences manually might be appropriate if only few data sources need to be integrated, however, it becomes tedious with the growing number of heterogeneous schemata.

Automatically integrating heterogeneous data sources is a long standing issue in the database research field and is of high relevance in many real-world domains [Rahm and Bernstein, 2001; Dhamankar *et al.*, 2004; Taheriyan *et al.*, 2013].

A standard approach to tackle this problem is to design a common ontology and to construct source descriptions which specify mappings between the sources and the ontology [Doan *et al.*, 2012]. In this paper, we approach the data integration problem consisting in automatically mapping a new relational data source onto a user provided ontology. To do so, we develop a new system that automatically builds a semantic model which describes a relational data source in terms of concepts and relationships defined by an ontology.

Consider the situation where we have some simple relational database tables with columns $\langle$*Surname, Event, Date*$\rangle$, and $\langle$*Company, Festival, Address*$\rangle$. We do not *a priori* know if "Date" is a date of birth of the person or the start date of the event (c.f. Fig. 1); or whether "Address" refers to the name of the place where the festival is located, or the email address of the company. Given an ontology like in Fig. 1, we would like to automatically map new data sources to the ontology.

In this paper we use Machine Learning (ML) techniques and Constraint Programming (CP) to learn mapping rules from previously mapped instances. To this end we formulate the Relational-To-Ontology Mapping Problem (REL2ONTO) as a Steiner Tree Problem (STP) with side constraints.

Firstly, we build a graph which includes attributes from the new source as well as ontology classes and properties. We name it *integration graph*. It uses information gathered from previously mapped data sources, and is further extended with information derived from the ontology. Secondly, we apply ML techniques to assign costs to its edges, by borrowing ideas from previous work by Taheriyan *et al.* [2016a]. Lastly, we use CP to find a minimum cost Steiner Tree in the graph by building a novel CP model and using solving techniques developed by De Uña *et al.* [2016]. The goal is to assign the costs of the edges in a way that the resulting Steiner Tree is a valid and coherent semantic model for the new source. Our contributions are: a novel modeling framework for the REL2ONTO problem that uses knowledge representation, ML and CP techniques; an efficient approach to handle attributes which cannot be matched to the ontology; a flexible and extensible approach to the problem, easy to reuse and adapt to specific situations through the use of CP.

2 Problem Statement

An ontology $\mathcal{O}$ includes basic elements such as classes (which represent concepts), literal values, individuals (members of classes) and properties [Spanos *et al.*, 2012]. Properties are classified into *object properties*, which relate two individuals, and *datatype properties*, which relate individuals

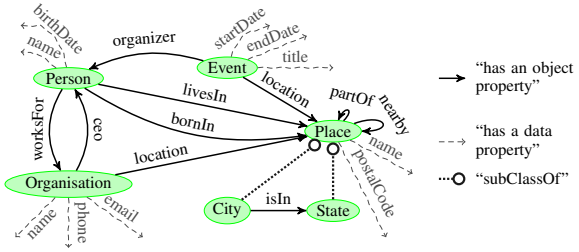


Figure 1: Example of ontology by Taheriyan *et al.* [2013].

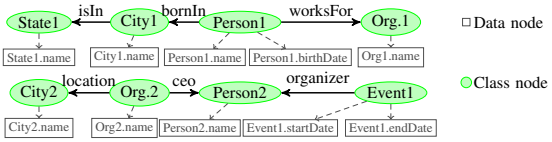


Figure 2: Examples of two semantic models

to literal values. Fig. 1 gives an example of an ontology. Here, an individual of the class "Organisation" can be related to an individual of the class "Person" via object properties "ceo" or "worksFor", and it can have a data property "name". We also consider a special "subclass" (or "is-a") type of object properties. For example, "Cities" and "States" are both "Places". A subclass inherits all the properties of the parent class.

A *semantic model* m is a directed graph with two types of nodes: *class nodes* and *data nodes*. We denote them as $\mathcal{C}_m$ and $\mathcal{D}_m$, respectively. $\mathcal{C}_m$ corresponds to classes in the ontology whereas $\mathcal{D}_m$ corresponds to data properties. Edges in the semantic model correspond to properties in the ontology, as shown in Fig. 2 (notice the same edges appear in Fig. 1). The semantic model may have several instances of the same ontology class, that is why class nodes are enumerated.

In our setting we work with relational sources. Hence, a *data source* s is a n-ary relation with a set of attributes $\mathcal{A}_s = (a_1, ..., a_n)$. We want to map them to the target ontology $\mathcal{O}$.

Following the traditional data integration framework [Doan *et al.*, 2012], we decompose the problem into two parts. First, *semantic labeling* [Rümmele *et al.*, 2018] finds correspondences between attributes from data sources and data nodes of the target ontology. Second, in the *schema mapping* part we want to generate the semantic models of data sources by identifying the connecting paths for the matched data nodes.

An *attribute mapping* function $\phi : \mathcal{A}_s \mapsto \mathcal{D}_m$ maps the attributes of the source s into the nodes of the semantic model m. It can be a partial mapping, (i.e. only some of the attributes are connected to the nodes of m). This function addresses the first part of the problem (i.e. the semantic labeling).

We define a *source description* as a triple $\delta = (s, m, \phi)$, where s is a source, m is a semantic model, and ϕ is an attribute mapping. Our problem can hence be stated as follows. We have an ontology $\mathcal{O}$ and a set of source descriptions $\Delta_T = \{(s_1, m_1, \phi_1), ..., (s_l, m_l, \phi_l)\}$. Given a new source $s^\star$, we want to build a semantic model $m^\star$ and an attribute mapping function $\phi^\star$ such that $\delta^\star = (s^\star, m^\star, \phi^\star)$ is an *appropriate* source description. We use the term "appropriate" since there might be many such triples which are well-formed source descriptions, but only one or a few will capture the intended meaning of the source. Our goal is to automatically build $\delta^\star$ such that it maximizes the *precision* and *recall* between the semantic model $m^\star$ and the semantic model $m^\dagger$ that the user considers correct.

3 ML for Training on Source Descriptions

3.1 Semantic Labeling

The semantic types $\mathcal{L}_\mathcal{O} = \{l_1, l_2, ..., l_p\}$ of an ontology correspond to all pairs (c, d), where c is a Class in $\mathcal{O}$, and d is a data property of that class (including inherited properties). E.g., in Fig. 1, we would get (City,name) and (State,name).

The first step to model the semantics of a new source $s^\star$ is to recognize the semantic types present in the source. We call this step *semantic labeling*, which assigns a confidence value to a match of an attribute from $s^\star$ to a type $l \in \mathcal{L}_\mathcal{O}$. Typically semantic labeling techniques encounter several problems such as: naming conflicts [Pinkel *et al.*, 2016], multiple data representations, and semantically different attributes might have syntactically similar content. Also, there may be a considerable number of attributes which have no corresponding property in the ontology (accidentally or deliberately).

We formulate the problem of semantic labeling as a multi-class classification problem. The known source descriptions Δ_T provide us the training sample. We compute a feature vector for each attribute in a data source and associate the known semantic type with the corresponding feature vector. The feature vector includes, among others, characteristics such as a number of whitespaces and other special characters, statistics of values in the column. One of the important features characterising information content of an attribute is Shannon's entropy. Shannon's entropy of a string X is defined as $H(X) = -\sum_i p_i \log_2 p_i$, where p_i is the probability of a character, whose index in character vocabulary is i, to appear in X, and the summation ranges over all characters in the vocabulary. To evaluate p_i, we evaluate normalized character frequency distribution of an attribute, as character counts in concatenated rows of the attribute, normalized by the total length of the concatenated rows. The alphabet consists of 100 printable characters. We then add the 100-dimensional vector of p_i to the attribute feature vector. We also compute a set of features based on similarity metrics inspired by works by Pham *et al.* [2016] or Ritze and Bizer [2017] (e.g. mean cosine similarity for character distributions of attribute values and string similarity metrics for attribute names). We train a random forest on the obtained sample. In this way, we learn the mapping $\psi : \mathcal{A}_s \times \mathcal{L}_\mathcal{O} \mapsto [0, 1]$, where $\psi(a_i, l_j)$ indicates the confidence that the attribute a_i is mapped to the semantic type l_j. Note that we keep all the matches, regardless of the confidence of the match. This is an important difference between our system and other approaches [Taheriyan *et al.*, 2016a] that remove some of the matches based on heuristics in order to simplify the task of finding the semantic model.

3.2 Alignment Graph

To provide an integrated view over the known source descriptions Δ_T, we align their semantic models as well as all semantic types. This is done by building an *alignment graph*.

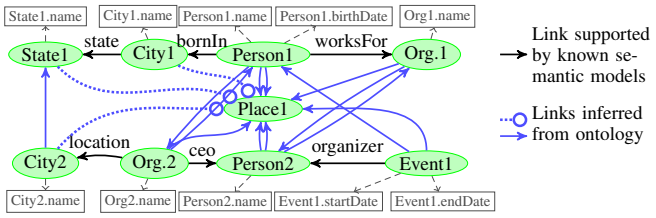


Figure 3: Example of alignment graph. We omit weights for clarity.

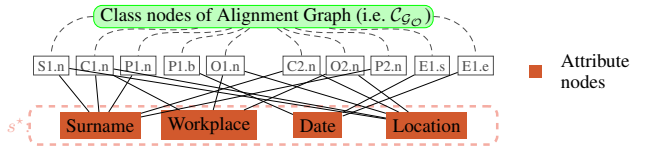


Figure 4: Example of integration graph. We omit weights and only show the most likely matches for clarity.

The alignment graph is a directed weighted graph $\mathcal{G}_\mathcal{O} = (\mathcal{V}_\mathcal{O}, \mathcal{E}_\mathcal{O})$ built on top of the known semantic models and expanded using the semantic types $\mathcal{L}_\mathcal{O}$ and the ontology $\mathcal{O}$. Similar to a semantic model, $\mathcal{G}_\mathcal{O}$ contains both class and data nodes. The links correspond to properties in $\mathcal{O}$ and are weighted. The full algorithm we use to construct the alignment graph $\mathcal{G}_\mathcal{O}$ was given by Taheriyan *et al.* [2016a].

Note how the alignment graph contains data nodes that correspond to semantic types. For instance, it could contain two nodes $City.name$ and $State.name$ rather than just one node $name$ connected to two nodes $City$ and $State$. We say these nodes are *induced* into the alignment graph by the semantic types. We note them $\mathcal{D}_{\mathcal{G}_\mathcal{O}}$ and call them the *data nodes* of the alignment graph. Class nodes are noted $\mathcal{C}_{\mathcal{G}_\mathcal{O}}$.

The graph is weighted by a function $w_\mathcal{O} : \mathcal{E}_\mathcal{O} \mapsto \mathbb{R}$ such that edges which are present in the known semantic models have lower weights than those which are inferred from the ontology. Taheriyan *et al.* [2016a] provide details on the weighting function.

3.3 Frequent Graph Pattern Mining

Certain patterns of connections can be prevalent in the domain. If we know that the "Person" works for the "Organization", then based on the known semantic models (Fig. 2), "City" is more likely to be the birth place of the "Person" rather than the location of the "Organization". To increase the coherence of the generated semantic models, we would like to discover such patterns of connections.

We mine these patterns from the set of semantic models in the training set Δ_T. This is known as Transactional Frequent Graph Pattern Mining. The frequency of a pattern is the number of semantic models which contain at least one subgraph isomorphic to the pattern. The *support* of the pattern is calculated as its frequency relative to the number of semantic models. This is an anti-monotonous measure, meaning that bigger patterns will have lower supports than their subgraphs. We solve the pattern mining task with the tool DIMSpan [Petermann *et al.*, 2017] which adapts gSpan [Yan and Han, 2002] pruning techniques to the typed graphs. We obtain patterns of size up to 6 in under 4 minutes for the biggest instances in our evaluation framework. Hence, this pattern mining procedure is a highly scalable approach compared to the technique based on SPARQL queries from [Taheriyan *et al.*, 2016b].

4 Steiner Tree Formulation

Given a graph $G = (V, E)$ and a subset of its nodes $T \subseteq V$, a Steiner Tree $G_s = (V_s, E_s)$ is a tree such that $T \subseteq V_s \subseteq V$ and $E_s \subseteq E$. That is, G_s spans all the nodes in T and may include additional nodes from V to ensure the connectedness of the constructed tree. The Steiner Tree Problem (STP) is stated as follows: given G and a weight function $w_f : E \mapsto \mathbb{R}$, find the Steiner Tree that minimizes the sum of the weights of the edges in E_s. This was proven to be NP-hard by Karp [1972].

To formulate the REL2ONTO schema mapping problem as a STP for a new source $s^\star$, we construct the *integration graph* $\mathcal{I}^{s^\star}_\mathcal{O} = (\mathcal{V}^{s^\star}_\mathcal{O}, \mathcal{E}^{s^\star}_\mathcal{O})$, where $\mathcal{V}^{s^\star}_\mathcal{O} = \mathcal{V}_\mathcal{O} \cup \mathcal{A}_{s^\star}$.

The set of edges $\mathcal{E}^{s^\star}_\mathcal{O}$ is constructed by using all the edges in the alignment graph, and edges connecting each attribute of $s^\star$ to the nodes in the alignment graph *induced* by the semantic types (i.e. the set of nodes in $\mathcal{D}_{\mathcal{G}_\mathcal{O}}$). We call this last set of edges $\mathcal{M}^{s^\star}_\mathcal{O}$ (for "matches"). Thus, $\mathcal{E}^{s^\star}_\mathcal{O} = \mathcal{E}_\mathcal{O} \cup \mathcal{M}^{s^\star}_\mathcal{O}$.

We associate a weighting function $w_\mathcal{I} : E \mapsto \mathbb{R}^+$ to the integration graph. For an edge $e \in \mathcal{E}_\mathcal{O}$, $w_\mathcal{I}(e) = w_\mathcal{O}(e)$. For an edge $e \in \mathcal{M}^{s^\star}_\mathcal{O}$ connecting attribute a_i to the node l_j induced by the semantic types, $w_\mathcal{I}(e) = -ln(\psi(a_i, l_j))$, making unlikely matches have a higher weight. An example of an integration graph can be found in Fig. 4.

Note that, although the alignment graph is directed, we disregard the direction of edges for the STP model. We disambiguate the direction of edges via their weights.

The goal is to build a subgraph $T^\star = (V^\star, E^\star)$ of $\mathcal{I}^{s^\star}_\mathcal{O}$ for the new source $s^\star$. The solution $T^\star$ will be used to build the source description $\delta^\star$. In particular, $(\mathcal{V}_\mathcal{O} \cap V^\star, \mathcal{E}_\mathcal{O} \cap E^\star)$ corresponds to the semantic model, and $(\mathcal{A}_{s^\star}, \mathcal{M}^{s^\star}_\mathcal{O} \cap E^\star)$ corresponds to the attribute mapping function.

The solution $T^\star$ must satisfy the following constraints: (i) $T^\star$ must be a subgraph of $\mathcal{I}^{s^\star}_\mathcal{O}$, (ii) $T^\star$ must be a tree, (iii) $\forall a \in \mathcal{A}_{s^\star}, a \in V^\star$, (iv) $\forall a \in \mathcal{A}_{s^\star}, degree(a) = 1$, (v) $\forall n \in \mathcal{D}_{\mathcal{G}_\mathcal{O}} \cap V^\star, degree(n) = 2$, (vi) $V^\star \cap \mathcal{C}_{\mathcal{G}_\mathcal{O}} \neq \emptyset$.

It is therefore natural to model this problem as a STP with side constraints. By designing the weighting function $w_\mathcal{I}$ through ML techniques, as shown in Section 3, our expectation is that the minimum cost Steiner Tree is a valid and coherent semantic model for the new source.

Patterns: As explained in Sec. 3.3, we use graph patterns to incentivise the solution tree $T^\star$ to contain subgraphs of the alignment graph that have been frequently seen in the training set. To do this, we use the support of each obtained patterns as a prize. If the tree contains a pattern, then its weight is automatically reduced by the value of the support of that pattern.

Unmatched Attributes: It is common that the data sources to be integrated will have columns that simply cannot be matched to the ontology. This can happen when a column of a source table contains some information that is uninteresting to the user, or because the ontology has not been properly designed. Examples of these situations can be found in domain

specific data [Pham *et al.*, 2016] or HTML tables [Ritze and Bizer, 2017]. In current systems, these columns are removed in a pre-processing step as a manual effort.

For this reason, we add two artificial Class nodes to the integration graph: *unknown* and *root*. The latter will be connected to every other Class node in $\mathcal{V}_{\mathcal{O}}^{s^\star}$, including *unknown*. We also add a set $U = \{unk_1, ..., unk_{|\mathcal{A}_{s^\star}|}\}$ of $|\mathcal{A}_{s^\star}|$ data nodes to the integration graph, each connected to exactly one node in $\mathcal{A}_{s^\star}$ and to the *unknown* Class node.

If an attribute a is matched to unk_a, then unk_a will be linked to Class node *unknown*. The other attributes can be matched normally and build a semantic model as usual. To maintain connectivity of $T^\star$, the *unknown* node and any of the other Class nodes in $T^\star$ will both be connected to the *root* node (in the optimal solution only one will be connected to the root). Note that if all attributes find a match, then neither *unknown* or *root* will be selected and the normal behavior will take place. The weights of the edges between these special nodes are assigned the same way as for the rest of the edges.

5 Modeling in Constraint Programming

A Constraint Optimization Problem (COP) is a tuple $P = (\boldsymbol{v}, C, o)$ where $\boldsymbol{v}$ is a set of *variables*, C is a set of n-ary *constraints* over variables $\boldsymbol{v}$ and o is an *objective function* $o : \boldsymbol{v} \mapsto \mathbb{R}$ to be (w.l.o.g.) minimized. A valuation such that all variables in $\boldsymbol{v}$ map to exactly one value, and all the constraints in C are satisfied by the valuation is a *solution* to P. A solution θ^* is optimal if $\nexists \theta, \theta(\boldsymbol{v}) < \theta^*(\boldsymbol{v})$. CP allows the user to model a COP and give it to a "black-box" solver to find solutions while optimizing the objective function.

Briefly, a CP solver assigns values to each variable of $\boldsymbol{v}$ in turn. CP uses a combination of complete search and *propagation*. The latter removes inconsistent values from the domains of variables. That way, values that cannot appear in a solution, given the current set of decisions, are never tried by the search. *Global constraints* are higher order constraints that enforce a complex constraint over a set of variables. They are implemented with specialized algorithms for performance.

5.1 REL2ONTO in Constraint Programming

We used the MINIZINC language [Nethercote *et al.*, 2007], and the CHUFFED solver [Chu, 2011]. Since attributes must be connected to exactly one node of the alignment graph, and that node will be in $\mathcal{D}_{\mathcal{G}_\mathcal{O}} \cap V^\star$, then the part of the problem between attribute nodes and the alignment graph is a matching problem. Each attribute must match exactly one node of $\mathcal{D}_{\mathcal{G}_\mathcal{O}}$. Note that not all nodes in $\mathcal{D}_{\mathcal{G}_\mathcal{O}}$ need to match to an attribute, as they are not all part of $T^\star$. Because there are global constraints in CP specialized in matching [Régin, 1994], we split the problem into two parts: the *steiner* global constraint [De Uña *et al.*, 2016] will only deal with the part of the integration graph that corresponds to the alignment graph, and the *alldifferent* global constraint will deal with the matching part of the problem. We use Boolean variables c_x for any node $x \in \mathcal{V}_\mathcal{O}$, c_y for any edge $y \in \mathcal{E}_\mathcal{O}$ and an array of variables $match$ indexed by the set of attributes of $s^\star$ to represent the tree $T^\star$. The semantics of the variables are: $c_x = true \Leftrightarrow x \in V^\star$, $c_y = true \Leftrightarrow y \in E^\star$ and $match[a] = d$ (for $a \in \mathcal{A}_{s^\star}$ and $d \in \mathcal{D}_{\mathcal{G}_\mathcal{O}}$) means that the edge $(a, d) \in \mathcal{M}_\mathcal{O}^{s^\star}$ is part of $T^\star$.

Additionally, for a given set of patterns $\mathcal{P}$ with a support function $w_\mathcal{P} : \mathcal{P} \mapsto \mathbb{R}$, we have a set of Boolean variables $c_p, \forall p \in \mathcal{P}$, that tell us whether a pattern p appears in $T^\star$ or not. The model is presented below.

$$\text{Minimize } w_{STP} + w_{ADIFF} - w_{PAT} \quad \text{such that} \tag{1}$$
$$\textit{steiner}(\{c_n | n \in \mathcal{V}_\mathcal{O}\}, \{c_e | e \in \mathcal{E}_\mathcal{O}\}, \mathcal{G}_\mathcal{O}, w_\mathcal{O}, w_{STP}) \tag{2}$$
$$\forall d \in \mathcal{D}_{\mathcal{G}_\mathcal{O}}, degree(d) \leq 1 \tag{3}$$
$$\forall d \in \mathcal{D}_{\mathcal{G}_\mathcal{O}}, c_d \Leftrightarrow degree(d) = 1 \tag{4}$$
$$\forall a \in \mathcal{A}_{s^\star}, match[a] \in \{d | (a, d) \in \mathcal{M}_\mathcal{O}^{s^\star}\} \tag{5}$$
$$\textit{alldifferent}(match) \tag{6}$$
$$\forall a \in \mathcal{A}_{s^\star}, c_{match[a]} = true \tag{7}$$
$$w_{ADIFF} = \sum_{(a,d) \in \mathcal{M}_\mathcal{O}^{s^\star}} w_\mathcal{I}((a, d)) * [\![match[a] = d]\!] \tag{8}$$
$$\forall p \in \mathcal{P}, \big(\forall e \in edges(p), c_e = true\big) \Leftrightarrow c_p \tag{9}$$
$$w_{PAT} = \textstyle\sum_{p \in \mathcal{P}} w_\mathcal{P}(p) * c_p \tag{10}$$
$$c_{unknown} \Rightarrow \big(c_{root} \wedge c_{(unknown, root)}\big) \tag{11}$$

Eq. 1 is the objective function: we minimize the cost of $T^\star$ while collecting prizes for each pattern we use. Eq. 2 enforces the solution $T^\star$ to be a tree defined by the c_n and c_e variables, subgraph of $\mathcal{G}_\mathcal{O}$ and of total weight w_{STP}. Eqs. 3 and 4 ensure that if a data node of $\mathcal{G}_\mathcal{O}$ is selected, then at most one edge reaches it (from the side of $\mathcal{G}_\mathcal{O}$) and otherwise it is disconnected. Eqs. 5 and 6 ensure that there is exactly one data node matched to an attribute. Eq. 7 ensures that if a data node of $\mathcal{G}_\mathcal{O}$ has been mapped to some attribute, then that data node must be in the solution tree, and vice-versa. Eq. 8 computes the cost w_{ADIFF} of the selected match edges in $\mathcal{M}_\mathcal{O}^{s^\star}$. Eq. 9 indicates that a pattern is used if and only if all its edges are selected in the tree. Eq. 10 computes the prizes collected by using patterns. Eq. 11 ensures that if the *unknown* class node is used, then it is connected to *root* through the edge $(unknown, root)$. Notice there is no further requirement for unmatched attributes, as the unk_i nodes behave like normal data nodes, and the connectivity requirement ensures that *root* is connected to the rest of the tree (i.e. the semantic model).

6 Results

From now on, we call our system SERENE if no patterns are used in the model, and SERENEPATS otherwise. We run experiments on two domains: museum (29 sources, 20 labels, 443 semantic attributes and 159 *unknown* attributes) and soccer (12 sources, 18 labels, 138 attributes and 45 *unknown*s).

We choose KARMA [Taheriyan *et al.*, 2016a] as our baseline. This system also phrases the REL2ONTO problem as STP and decomposes it further into two parts. However, it uses heuristic algorithms both for the matching and for the STP parts. It solves the problems sequentially, i.e., once it produces a set of candidate mappings for attributes into the ontology, it fixes this set and moves onto the STP part. Additionally, it does not consider unmatched attributes in the sources. To ensure that KARMA also handles such attributes,

we change its semantic labeling model, SemanticTyper [Ramnandan *et al.*, 2015], to ours and we add a special *unknown* ontology which gives specification for *root* and *unknown* nodes of the alignment graph. The first modification also ensures the fairness of evaluation since both KARMA and SERENE will have the same matches for attributes.

The performance of these systems is estimated in terms of *precision* and *recall*. Assuming m^p is the predicted semantic model and $m^\dagger$ is the correct semantic model, then: $prec = |rel(m^p) \cap rel(m^\dagger)| / |rel(m^p)|$, $recall = |rel(m^p) \cap rel(m^\dagger)| / |rel(m^\dagger)|$ where $rel(m)$ is the set of triples (u, e, v) with e being an edge from the vertex u to the vertex v in the semantic model m. Note that unmatched attributes as well as *unknown* and *root* nodes are not part of these sets. We also perform a modification to the ground truth semantic models: if the true semantic label of an attribute a is not present in any of the previously mapped data sources, we substitute it with unk_a and map it to the *unknown* class node. This way we can validate how well the systems can detect earlier unseen cases.

We compare our semantic labeling model against the state-of-the-art model *DSL* [Pham *et al.*, 2016], which was shown to perform even better than SemanticTyper. We use *mean reciprocal rank* (MRR) to evaluate semantic labeling models. This measure is useful to estimate how highly the true semantic label is ranked among the top k suggestions. It is defined as: $MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{r_i}$, where r_i is the rank of the correct semantic label for the attribute a_i among the top k predictions and n is the number of attributes in the data source.

We perform an evaluation strategy outlined by Taheriyan *et al.* [2016a]. Let M_j be the set of j known semantic models. For each data source s_i in the domain we perform experiments $t-1$ times, where t is the total number of data sources in the domain and each experiment has a different number of known semantic models $M_1, M_2, \ldots, M_{t-1}$. For example, in the soccer domain, for source s_1 we run experiments 11 times using $M_1 = \{m_2\}, M_2 = \{m_2, m_3\}, \ldots, M_{11} = \{m_2, m_3, \ldots, m_{12}\}$. We repeat the procedure for all sources in the domain and then average the results. This ensures that each source is at least once in the training and testing datasets.

We have run all our experiments on a Dell server with 252 GB of RAM, 2 CPUs (4 cores). We use a timeout threshold of 15s for CHUFFED, which runs on a single core.

6.1 Experimental Results

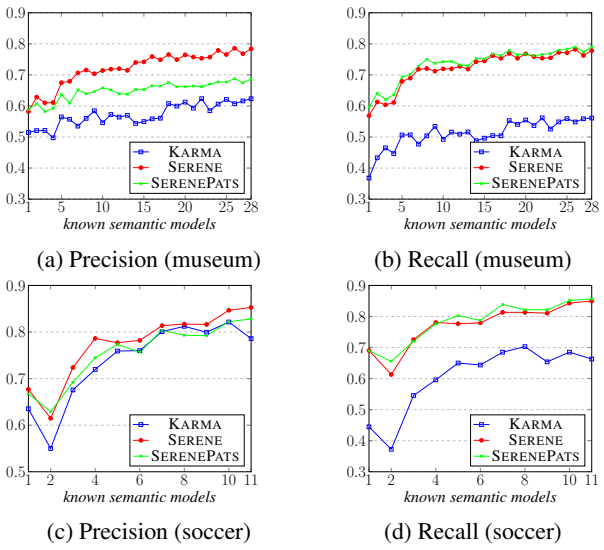
To evaluate our new system SERENE[1], we show that its semantic labeling model produces more accurate matches for attributes and that the CP formulation leads to better semantic models in terms of precision and recall.

Tab. 1 provides evidence that our new semantic labeling model is better suited for the task when there are unmatched attributes in domains. DSL uses heuristic measures to capture the similarity of attributes within the same class, but unmatched attributes are clearly dissimilar from known semantic types, thus similarity measures may be unsound in the presence of unmatched attributes. Our intuition as to why our approach performs better is that we have incorporated features which are derived directly from attribute values and are not based on the notion of similarity [Rümmele *et al.*, 2018].

[1]http://github.com/NICTA/serene-python-client/tree/stp/stp

Model	MRR scores		Train time (s)	
	museum	soccer	museum	soccer
DSL	0.560	0.618	156.6	36.3
SERENE(PATS)	0.866	0.827	100.6	6.80

Table 1: Average performance of semantic labeling models



(a) Precision (museum) (b) Recall (museum)

(c) Precision (soccer) (d) Recall (soccer)

Figure 5: Performance for the museum and soccer domains. Average size of integration graphs for the museum domain: 88 nodes and 1129 edges. Average run time for the museum domain: SERENE 0.5s, KARMA 0.5s, SERENEPATS 1s.

We then evaluate SERENE and SERENEPATS against the state-of-the-art system KARMA. All three systems have access to the same set of matches produced by our semantic labeling model. Fig. 5 shows their performance on the museum and soccer domains. We report average precision and recall for the systems with regard to variable number of semantic models in the training set. For SERENE and SERENEPATS we report the first solutions found by CHUFFED. All three systems find solutions in less than a second on average. We use default parameters for KARMA which were shown to yield the best results. As we can see, SERENE produces on average the best semantic models in terms of precision while SERENEPATS generates slightly better models in terms of recall. The t-tests run for the obtained results confirm statistical significance of our improvements at the level $p < 0.01$

In Tab. 2 we show how the systems perform if we manually remove the unmatched attributes for the leave-one-out setting. In such scenarios, SERENE also yields better solutions than KARMA, especially in terms of recall, and can often prove optimality of the found solutions. The first solutions found by SERENEPATS tend to be bigger than those of SERENE, hence the precision suffers but the recall is better. We have also observed that given enough time SERENEPATS finds much better solutions both in precision and recall. These observations provide further evidence of the validity of our framework.

In Fig. 6 we show how the performance of SERENE changes across sequentially found solutions. CHUFFED, the

Model	Precision		Recall	
	museum	soccer	museum	soccer
KARMA	0.72	0.81	0.54	0.64
SERENE	0.82	0.85	0.62	0.75
SERENEPATS	0.6	0.86	0.64	0.75

Table 2: Performance with unknowns manually removed

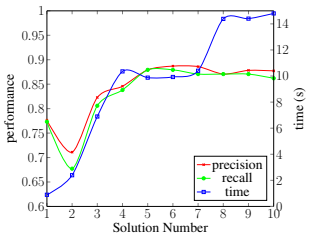


Figure 6: Performance of SERENE on the museum domain.

solver of SERENE, successively finds better solutions in terms of Eq. 1. However, that does not translate directly into more precise semantic models. On average, SERENE yields very good results already with the first solution found, and plateaus after the third solution.

Additionally, we investigate how the performance is influenced by introducing a weighting scheme on the patterns, i.e., costs of patterns are scaled by a factor. In Fig. 5 we use a scaling factor 1 for pattern costs. When trying scaling factors 5, 10 or 20 for pattern costs from the museum domain we can generate semantic models which are almost 90% in precision and recall. This is a 20% improvement in precision compared to the first solution found with a scaling factor of 1, and a 10% improvement both in precision and recall for the first solution from SERENE. Contrarily to SERENE, SERENEPATS with scaled factors produced better solutions in terms of precision over time. However, the sequential solutions are not always better, and hitting the optimal solution may take hours.

We have also observed that SERENE and SERENEPATS manage to find the ground truth on many instances among its sequential solutions. This property of our approach can be used to configure various parameters for the model, e.g., scaling factors for pattern costs or match score.

7 Related Work

Most approaches to solve the REL2ONTO problem are based on heuristic rules and alignment of constraints specified within relational schemata and ontologies. Pinkel *et al.* [2016] and Spanos *et al.* [2012] give a comprehensive overview and comparison of existing mapping generation tools based on this approach. As examples, there are BootOX [Jiménez-Ruiz *et al.*, 2015], MIRROR [de Medeiros *et al.*, 2015] and ontop [Fagin *et al.*, 2009]. Briefly, these tools first apply a default direct mapping specified by the W3C. Then, the default ontology is enriched by using explicit and implicit schema constraints. Finally, ontology alignment techniques are applied to match the default ontology to the target ontology. The main advantage of these systems is that they are fully automatic. Our approach is complementary to them and at its current stage is semi-automatic. However, CP offers a convenient framework to incorporate integrity constraints specified within relational or ontological schema as additional constraints to govern the search for the solution.

A major issue with fully automatic systems is that constraints may be inconsistent or absent completely, e.g., data from Web services or tables on the Web. To overcome this issue, Limaye *et al.* [2010] design a ML system to annotate web tables with entities for cell values, semantic labels for attributes and relationships for binary combinations of attributes. As in our approach, they decompose the process of mapping into two main stages: semantic labeling and finding relationships between matched semantic labels. Limaye *et al.* [2010] enrich their data sources by using YAGO Knowledge Base (KB)[Suchanek *et al.*, 2007]. Mulwad *et al.* [2013] extend this approach by leveraging information from Wikitology KB. Venetis *et al.* [2011] develop a scalable approach to recover the semantics of Web tables by incorporating data from the isA database KB. Ritze and Bizer [2017], on the other hand, use DBPedia as their KB. Hence, these approaches are limited to domains well represented in those knowledge bases. Also, they are not able to find the relation between attributes in the table if there is no direct connection between the attributes. Our approach, on the other hand, allows a model to be trained on any data and can infer complex semantic paths which might exist between attributes. However, it could be further bootstrapped by leveraging external knowledge bases. This is especially beneficial when the system does not have sufficient training data.

As mentioned above, the approaches for mapping Web tables also perform semantic labeling. They design various similarity metrics for attribute names and values. However, they disregard the attributes which are not matched to the ontology (the *unknown* set of attributes) which are especially abundant on the Web [Ritze and Bizer, 2017; Pham *et al.*, 2016]. Clearly we cannot speak about similarity for these attributes, since they are rather dissimilar from known semantic types. Our approach differs from previous work in that we incorporate an efficient method to handle the *unknown* class, compared to the state-of-the-art approach DSL [Pham *et al.*, 2016].

We build upon the work of Taheriyan *et al.* [2016a] and use their ideas for the construction of the alignment graph. The difference at this step is that we introduce the *unknown* and *root* class nodes and as many *unknown* data nodes as there are attributes in the modeled data source. These nodes serve to capture the unmatched attributes from the source. Though we have modified KARMA to treat these additional nodes as well, our approach outperforms KARMA since we have additional constraints for these nodes and use an exact algorithm to solve the STP. Taheriyan *et al.* [2016a] treat the matching and STP parts of REL2ONTO independently and use heuristic algorithms for both. We, on the other hand, use exact algorithms for both parts and address them within a unified CP model. In their follow up work, Taheriyan *et al.* [2016b] suggest using graph patterns to boost the performance of their system. To this end, they had to revise their algorithm by introducing additional heuristics. However, in our case we only had to add pattern variables and to modify the objective function in the MINIZINC model. No changes to the solver were required. This makes our system very convenient and opens

directions for validating various additional constraints.

8 Conclusion

We have introduced a new approach to solve the problem of relational-to-ontology schema mapping. Our experiments show that SERENE generates on average more consistent semantic models in terms of precision and recall compared to the state-of-the-art approach KARMA. SERENEPATS produces, on average, better semantic models in terms of recall, and its precision can be enhanced by adding a scaling factor for pattern costs. Furthermore, our approach is highly flexible and easy to extend with arbitrary side constraints thanks to the use of CP instead of a specific heuristic algorithm.

References

[Chu, 2011] G.G. Chu. *Improving combinatorial optimization*. PhD thesis, The University of Melbourne, 2011.

[de Medeiros *et al.*, 2015] L.F. de Medeiros, F. Priyatna, and O. Corcho. MIRROR: Automatic R2RML mapping generation from relational databases. In *ICWE*, pages 326–343. Springer, 2015.

[De Uña *et al.*, 2016] D. De Uña, G. Gange, P. Schachte, and P.J. Stuckey. Steiner tree problems with side constraints using constraint programming. In *AAAI*, pages 3383–3389. AAAI Press, 2016.

[Dhamankar *et al.*, 2004] R. Dhamankar, Y. Lee, A. Doan, A.Y. Halevy, and P.M. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proc. of SIGMOD*, 2004.

[Doan *et al.*, 2012] A. Doan, A. Halevy, and Z. Ives. Principles of data integration, 2012.

[Fagin *et al.*, 2009] R. Fagin, L.M. Haas, M. Hernández, R.J. Miller, et al. Clio: Schema mapping creation and data exchange. In *Conceptual modeling: foundations and applications*, pages 198–236. Springer, 2009.

[Jiménez-Ruiz *et al.*, 2015] E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, I. Horrocks, et al. BootOX: Practical mapping of RDBs to OWL 2. In *ISWC*, pages 113–132. Springer, 2015.

[Karp, 1972] R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. 1972.

[Limaye *et al.*, 2010] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. of the VLDB Endowment*, 3(1-2):1338–1347, 2010.

[Mulwad *et al.*, 2013] V. Mulwad, T. Finin, and A. Joshi. Semantic message passing for generating linked data from tables. In *Proc. of ISWC*, 2013.

[Nethercote *et al.*, 2007] N. Nethercote, P.J. Stuckey, R. Becket, S. Brand, G.J. Duck, and G. Tack. MiniZinc: Towards a Standard CP Modelling Language. In *Procs. of CP2007*, volume 4741 of *LNCS*. Springer, 2007.

[Petermann *et al.*, 2017] A. Petermann, M. Junghanns, and E. Rahm. DIMSpan - transactional frequent subgraph mining with distributed in-memory dataflow systems. *arXiv preprint arXiv:1703.01910*, 2017.

[Pham *et al.*, 2016] M. Pham, S. Alse, C.A. Knoblock, and P. Szekely. Semantic labeling: a domain-independent approach. In *Proc. of ISWC*, pages 446–462. Springer, 2016.

[Pinkel *et al.*, 2016] C. Pinkel, C. Binnig, E. Jiménez-Ruiz, E. Kharlamov, et al. RODI: Benchmarking relational-to-ontology mapping generation quality. *Semantic Web*, 2016.

[Rahm and Bernstein, 2001] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[Ramnandan *et al.*, 2015] S.K. Ramnandan, A. Mittal, C.A. Knoblock, and P. Szekely. Assigning semantic labels to data sources. In *Proc. of ESWC*, 2015.

[Régin, 1994] J.-C. Régin. A filtering algorithm for constraints of difference in csps. In *AAAI*, volume 94, pages 362–367, 1994.

[Ritze and Bizer, 2017] D. Ritze and C. Bizer. Matching web tables to DBpedia - A feature utility study. In *Proc. of EDBT*, pages 210–221, 2017.

[Rümmele *et al.*, 2018] N. Rümmele, Y. Tyshetskiy, and A. Collins. Evaluating approaches for supervised semantic labeling. In *Proc. of the Workshop on Linked Data on the Web*, 2018.

[Spanos *et al.*, 2012] D.-E. Spanos, P. Stavrou, and N. Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web*, 3(2):169–209, 2012.

[Suchanek *et al.*, 2007] F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proc. of the international conference on World Wide Web*, pages 697–706. ACM, 2007.

[Taheriyan *et al.*, 2013] M. Taheriyan, C. A. Knoblock, P. Szekely, and J.L. Ambite. *A Graph-Based Approach to Learn Semantic Descriptions of Data Sources*, pages 607–623. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Taheriyan *et al.*, 2016a] M. Taheriyan, C.A. Knoblock, P. Szekely, and J.L. Ambite. Learning the semantics of structured data sources. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37:152–169, 2016.

[Taheriyan *et al.*, 2016b] M. Taheriyan, C.A. Knoblock, P. Szekely, and J.L. Ambite. Leveraging linked data to discover semantic relations within data sources. In *Proc. of ISWC*, pages 549–565, 2016.

[Venetis *et al.*, 2011] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, et al. Recovering semantics of tables on the web. *Proc. of the VLDB Endowment*, 4(9), 2011.

[Yan and Han, 2002] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proc. of ICDM*, pages 721–724. IEEE, 2002.