Schema Mapping Discovery From Example Data Using ILP

Manuel Fink and Heiner Stuckenschmidt

Research Group Data and Web Science, University of Mannheim, Germany

Abstract. We frame the task of schema mapping discovery for Data Exchange as an ILP problem to discuss resulting challenges and advantages. The challenges stem from the necessity of learning complex mappings in practice as we illustrate with simple examples.

1 Introduction

Data Exchange describes the problem of making data from a source schema accessible in a target schema by the use of a schema mapping that describes the relationship between the heterogeneous schemata. In this work, we discuss schema mapping discovery from example data that is given in both the source and target schema. We restrict ourselves to the case of two relational database schemata and assume metadata is available that specifies primary and foreign key constraints within both schemata as well as data domains of attributes. This use case is motivated by an industry problem in which the content of a database needs to be transformed to be used with a newer ERP software. While an official tool is available that does this task for the two ERP systems at hand, it is slow, inflexible (all data or nothing) and further preprocessing steps on the source data are required. These problems could be alleviated if one could capture the underlying logic with rules that express how the data needs to change during the workflow. The formal nature of the rules allows an easy transition to a definition of the data migration task in an arbitrary conversion tool. As the official tool is essentially a black box, the idea is to execute the complete workflow once on a system filled with dummy data and learn transformation rules by comparing the source data to the resulting target data.

2 Approach

To illustrate one kind of transformation logic that we are interested in, consider the data example in Figure 1 that shows the same foods in both source and target. In this example, the content of the three source tables *Vegetable*, *AnimalProduct* and *NutritionalValue* is stored in the target within two tables *Ingredient* and *VegetarianIngredient*. Some of the *VegetarianIngredient* tuples stem from a join between *Vegetable* and *NutrionalValue*, some from a join between *AnimalProduct* and *NutrionalValue*. More specifically, it is filled with all foods from the *Vegetable* table and those foods from the *AnimalProduct* table which are of type *egg*. During the transformation, a renaming and reordering of the macro nutrition attributes has taken place. A schema mapping that correctly describes this relationship is given by the two horn rules in Figure 1.
The example suggests that ILP could be used to infer the transformation rules. Indeed, the problem of learning rules to construct a specific target table from the source tables, can be formulated within the ILP framework [7] as follows.

Vegetable

ID	Name	NID
017	Mushroom (white)	11
032	Mushroom (brown)	11
067	Potato	24

AnimalProduct

ID	Name	Type	NID
7254	Beef (Minced)	meat	34
8696	Chicken (Breast)	meat	65
8920	Egg (Chicken)	egg	67

NutritionalValue

ID	KCal	Protein	CarboHydrates	Fat
11	16	2.7g	0.6g	0.2g
24	76	1.9g	15.6g	0g
34	208	20.5g	0g	14g
65	102	23g	0g	0.7g
67	137	11.9g	1.5g	9.3g

VegetarianIngredient

ID	Name	E	C	F	P
017	Mushroom (white)	16	0.6g	0.2g	2.7g
032	Mushroom (brown)	16	0.6g	0.2g	2.7g
067	Potato	76	15.6g	0g	1.9g
8920	Egg (Chicken)	137	1.5g	9.3g	11.9g

$$AnimalProduct(ID, Name, \text{egg}, NID) \wedge NutritionalValue(NID, E, P, C, F) \rightarrow VegetarianIngredient(ID, Name, E, C, F, P) \quad (1)$$

$$Vegetable(ID, Name, NID) \wedge NutritionalValue(NID, E, P, C, F) \rightarrow VegetarianIngredient(ID, Name, E, C, F, P) \quad (2)$$

Fig. 1. Source (left) and target (right) databases storing food information in heterogenous schemata whose relationship is described by the schema mapping below.

The ground facts derived from tuples in source relations are the background knowledge B, while ground facts from tuples in the given target relation form the positive evidence E^+. Any ground fact that does not resemble a tuple in the target relation can be used negated as negative evidence E^-. Then, a hypothesis H is a potential schema mapping. The following two constraints on an ILP problem are trivially satisfied as B does not share any predicates with E^- or E^+ and because B and E^- are always satisfiable here:
(i) Prior Satisfiability: $B \wedge E^- \nvDash false$
(ii) Prior Necessity: $B \nvDash E^+$
Concerning the two criteria that a hypothesis H has to satisfy to be accepted as solution, we find that they are justified for the problem and represent intuitive properties that the rules should have:
(iii) Posterior Sufficiency: $B \wedge H \vDash E^+$
(iv) Posterior Satisfiability: $B \wedge H \wedge E^- \nvDash false$
Posterior sufficiency (iii) requires that all target tuples are inferred from the source tuples. Posterior Satisfiability (iv) demands from a solution that it does not produce *too much*, i.e. tuples that should not be there.

3 Challenges

We will now present further characteristics of Data Exchange that complicate the application of ILP algorithms to it. We justify them with the more complex transformation scenario shown in Figure 2.

Person

ID	Name	CityID	EmpID
0	Chris	5	13
1	Liz	4	25

City

ID	Name	LatLong
3	New York	40.7128N 74.0060W
4	Philadelphia	39.9526N 75.1652W
5	Baltimore	39.2904N 76.6122W

Employer

ID	Name	CityID
13	Comcast	4
25	HBO	3

Commute

ID	Employee	Start	Stop
1	Chris	1	2
2	Liz	2	3

GeoLocation

ID	LatLong
1	39.2904N 76.6122W
2	39.9526N 75.1652W
3	40.7128N 74.0060W

$$\begin{aligned} &Person(ID, PName, CID1, EID) \wedge City(CID1, CName1, Loc1) \\ &\wedge Employer(EID, EName, CID2) \wedge Ctiy(CID2, CName2, Loc2) \\ &\rightarrow \exists\, CoID, GID1, GID2 :\ Commute(CoID, PName, GID1, GID2) \\ &\wedge GeoLocation(GID1, Loc1)\ \wedge\ GeoLocation(GID2, Loc2) \end{aligned} \tag{3}$$

Fig. 2. Source database (left) containing data about peoples' work place and a Target (right) database storing commute information derived from it with the given rules.

First, we can see that there is now information in the target that is not inferrable from source information (Commute.ID, Commute.Start, Commute.Stop and GeoLocation.ID). In the practical problem we are trying to solve, such attributes are for example integer keys that appear to be randomly generated depending on the order in which the objects in the database are transformed. As they do not represent characteristics of the stored objects, they are typically called *labeled nulls* in Data Exchange. Learning rules that are so specific that they generate exactly the given labeled nulls from the source examples would be overfitting. However, as long as a schema mapping produces labeled null values that preserve the semantic relation between data objects, their actual values do not even matter.

This affects the language bias needed for practical schema mappings in two ways. First, to produce labeled nulls, a schema mapping needs existential quantifiers in the head of the rule for any attributes that are not determined by source values. Furthermore, generating target tuples with labeled nulls for one relation at a time, does not ensure correct foreign key relationships between target relations. Allowing more than one relation in the head of a schema mapping is a solution to this problem, as this allows the

4 Fink et al.

joint generation of semantically related tuples from different relations, connected via shared labeled nulls. Consequently, GLAV (global-and-local-as-view) mappings, i.e. sourge-to-target tuple-generating dependencies, as the one in Figure 2 have been the focus of most existing Data Exchange works [1–4, 6]. Given a vector $\bar{x}$ of variables, a vector $\bar{y}$ of variables representing labeled nulls and ϕ, ψ being first-order formulas with atoms using variables from $\bar{x}$ and $\bar{y}$, a GLAV schema mapping has the following form: $\forall \bar{x}\ \phi(\bar{x}) \rightarrow \exists \bar{y}\ \psi(\bar{x}, \bar{y})$.

As ILP systems typically learn rules with only one relation and no existential quantifier in the head, it is not straight-forward how to achieve GLAV mapping expressivity with them. One idea is using metadata information about foreign key relationships in the target schema to join related target relations into one universal table, so one relation in the head of the mapping rule is sufficient. The tuples generated upon application of the mapping rule would need to be split to gain tuples for the individual relations. As in our problem there are thousands of tables, some with more than 80 attributes, scalability is already a big challenge and this could make it even more ambitious.

To make a bottom-up search possible, one needs data examples that describe individual objects in their source and target representation. As the complete database content is given as a whole, instance matching would be needed to produce these fine-grained examples to bootstrap a bottom-up search. However, as the schema is heterogenous, schema matching is a prerequisite to be able to identify similar instances. It gets further complicated by the fact that information about objects is spread across multiple tables and even with information about foreign key relations, it is often ambiguous how to denormalize tables.

On the other hand, metadata about key relationships could be leveraged to more efficiently navigate the hypothesis space because shared variables of relations in a rule's body could be restricted to those attributes that are in a key relationship. Similarly, if a variable in x is bounded to an attribute of a certain data domain, one could restrict its appearance in the head to attributes of the same domain.

An immediate advantage of ILP systems is that schema mappings with constants in the rule body could be learned. This could prove very valuable in practice as Figure 1 showed. On the other hand, regarding constants in our problem, their usage could also pose a problem for applying ILP to it. In a logic context, multiple ground facts containing the same constant x are supposed to express relations of some abstract entity x. In a database though, a constant can occur multiple times in different tuples with no semantic relationship between those tuples which could mislead the search. This is especially common for numerical values.

A long-term goal is to support data transformation functions on top of schema changes as in $\forall \bar{x}\ \phi(\bar{x}) \rightarrow \exists \bar{y}\ \psi(\mathbf{f}(\bar{x}), \bar{y})$. Imagine *Ingredient.E* in Figure 1 storing all energy values suffixed by "kCal" or converted to kJ instead by multiplying the values with a constant. So far, we developed a prototype that learns such transformation functions in a setting where the database schema is fixed. In the scenario we dealt with, a mapping i described a copy process for table T_i with columns $x_1, ..., x_{n_i}$ affected by functions $f_1, ..., f_{n_i}$ which where either the identity function id or from a pool of parameterized meta functions: $\forall x_1, ..., x_{n_i}\ T_i^{Source}(x_1, ..., x_{n_i}) \rightarrow T_i^{Target}(f_1(x_1), ..., f_{n_i}(x_{n_i}))$. The difficulty in this case was identifying corresponding source and target tuples due to

transformed keys to generate examples for the learning of the transformation functions. Another challenge was the existence of columns storing the concatenation of multiple referenced key values as a violation of first database normal form. The number of splits as well as the offset positions had to be learned as well as individual transformation functions for the different parts. In addition, there were multiple rules on how to split and transform the same column. The choice of rule for a given tuple was determined by the value(s) of one or multiple control columns within the same table.

4 Related Work

Finding rules that correctly capture the relationship between the source and target schemata is a non-trivial task for which multiple different approaches have been studied. For example, early works such as Clio [5] use as input a set of correspondences between attributes that can either be specified by a user or automatically generated with schema matching techniques. Together with metadata information about the schemata, such as foreign key relationships, schema mappings can be suggested that fit these specifications. Systems like these aim to help a user formalize a desired schema mapping without the need for database expert knowledge. As they do not assume data examples to be available, they can not evaluate the correctness of the candidate mappings. The use of data examples for schema mapping discovery was explored later-on.

Muse [1] was one of the first systems to do so by generating data examples for the user of a mapping tool to visualize the effect of schema mappings.

EIRENE [2] is able to synthesize GLAV schema mappings by asking a user to illustrate its desired outcome with a small data example including source and target tuples. In the problem we consider, the two database contents are provided as a whole and it is unknown which source relations need to be mapped to which target relations. In EIRENE, the section within source and target schema, that includes the relations for the rule, is implicitly given by the predicates that occur in the examples resulting in a smaller search space. Moreover, due to the iterative input of examples, correspondences between source and target tuples are known.

MWeaver [8] is another system that allows a user to enter data examples which are used to infer schema mappings. As the user enters more and more examples, the number of possible candidate mappings tends to decrease until only the intended mapping remains. Unlike in EIRENE, the user does not enter examples consisting of both source and target information. Instead, she defines a set of attributes for a target table and fills in the cells of the table one-by-one to demonstrate which information she expects in the table. As the user enters more and more values for an attribute, it becomes clear which source attributes are needed in the body of the rule because only they contain these values. Still, unless all attributes stem from the same source table, it remains a challenge on how to join all source tables that contain needed attributes in the correct way. This is similar to the problem that the first approaches tackled. The difference this time is that data examples can be used to guide the search and reject joins that do not produce the target tuples intended by the user.

CMD [6] is a recent approach that can be used to find a subset among given candidate schema mappings that collectively best describe the given data examples. The

schema mappings used as input are assumed to satisfy metadata constraints and known correspondences and could be provided by existing works such as Clio. A central contribution is the tolerance of dirty data examples. Probabilistic Soft Logic allows this by formulating the selection of the best mapping subset as optimization problem. As input the solver is given the information which target tuple is produced by which schema mappings and also about tuples generated by a mapping that are not correct target tuples. It then computes a subset of the schema mappings that minimizes the amount of unexplained target tuples, wrongly produced tuples and number of mappings in the set. Due to labelled nulls and corrupted examples, it is not always possible to decide if a target tuple is correctly produced by a schema mapping or if a tuple produced by it corresponds to a valid target tuple. Therefore the degree of equality between tuples is not expressed binary but as a number between 0 and 1. This warrants the use of a probabilistic logic system. While the use of logic suggests a connection to mapping discovery using ILP, it is worth noting that logic is not used to learn schema mappings from example data but to select the best fitting mapping subset under noisy examples.

While EIRENE and CMD focus on GLAV mappings, MWeaver is limited to learning project-join queries, i.e. a subset of GAV mappings. Therefore none of these systems could produce the mapping given in Figure 1 as constants in the body are not supported. Furthermore, MWeaver does not support more than one relation in the head which means mappings involving foreign key relationships in target relations like in Figure 2 are not supported.

A theoretical framework for the discovery of GLAV schema mappings from data examples was developed in [4] and extended in [3]. Apart from complexity results, it introduces criteria for a schema mapping to be called *valid* and *fully explaining* which are semantically equal to constraints (iii) and (iv) that ILP requires from a solution (see Section 2). The framework also mentions schema mappings with constants in the body of the rules when introducing a language of repairs for GLAV mappings.

While these works deal with certain aspects of the problem already, exploring it with ILP is (to the best of our knowledge) a new approach.

References

1. Alexe, B., Chiticariu, L., Miller, R.J., Tan, W.C.: Muse: Mapping understanding and design by example. In: 2008 IEEE 24th International Conference on Data Engineering. pp. 10–19 (April 2008)
2. Alexe, B., ten Cate, B., Kolaitis, P.G., Tan, W.C.: Designing and refining schema mappings via data examples. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. pp. 133–144. SIGMOD '11, ACM (2011)
3. Cate, B.T., Kolaitis, P.G., Qian, K., Tan, W.C.: Approximation algorithms for schema-mapping discovery from data examples. ACM Trans. Database Syst. **42**(2), 12:1–12:41 (Apr 2017)
4. Gottlob, G., Senellart, P.: Schema mapping discovery from data instances. J. ACM **57**(2), 6:1–6:37 (Feb 2010)
5. Haas, L.M., Hernández, M.A., Ho, H., Popa, L., Roth, M.: Clio grows up: From research prototype to industrial tool. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. pp. 805–810. SIGMOD '05, ACM (2005)
6. Kimmig, A., Memory, A., Miller, R.J., Getoor, L.: A collective, probabilistic approach to schema mapping. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE). pp. 921–932 (April 2017)
7. Muggleton, S., de Raedt, L.: Inductive logic programming: Theory and methods. The Journal of Logic Programming **19-20**, 629 – 679 (1994)
8. Qian, L., Cafarella, M.J., Jagadish, H.V.: Sample-driven schema mapping. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. pp. 73–84. SIGMOD '12, ACM (2012)