Using Domain Ontologies to Discover Direct and Indirect Matches for Schema Elements *

Li Xu
Department of Computer Science
University of Arizona South
lxu@email.arizona.edu

David W. Embley
Department of Computer Science
Brigham Young University
embley@cs.byu.edu

Abstract

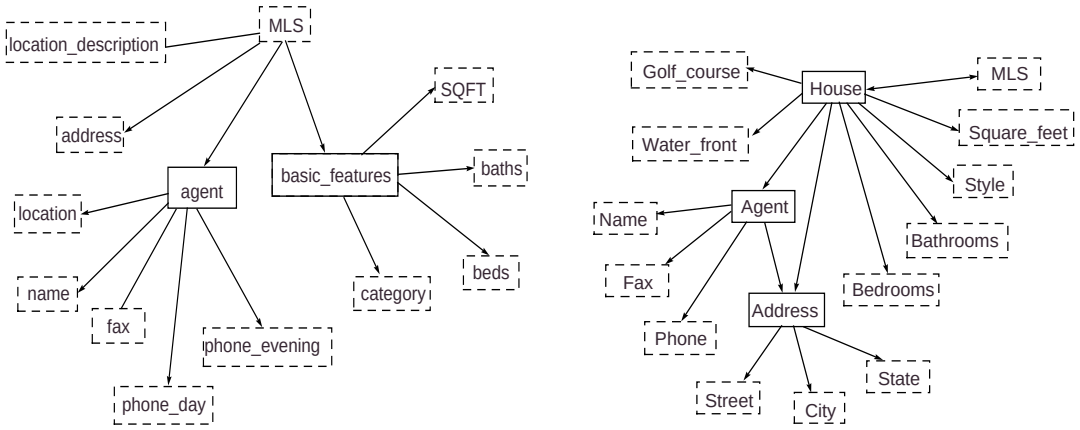
Automating schema matching is challenging. Previous approaches (e.g. [DDH01, RB01]) to automating schema matching focus on computing direct matches between two schemas. Schemas, however, rarely match directly. Thus, to complete the task of schema matching, we must also compute indirect matches. In this paper, we focus on recognizing expected values as a technique to discover many direct and indirect matches between a source schema and a target schema. This technique relies on domain ontologies, which must be handcrafted. The benefits appear to justify the cost as demonstrated in the experiments we have conducted over a real-world application. The experiments show that this technique increases the results by an increase about 20 percentage points, yielding an overall result above 90% precision and recall for both direct and indirect matches.

1 Introduction

In this paper, we focus on the long-standing and challenging problem of automating schema matching [RB01]. Schema matching is a key operation for many applications including data integration, schema integration, message mapping in E-commerce, and semantic query processing [RB01]. Schema matching takes two schemas as input and produces a semantic correspondence between the schema elements in the two input schemas [RB01]. In this paper, we assume that we wish to map schema elements from a *source* schema into a *target* schema. In its simplest form, the semantic correspondence is a set of *direct matches* each of which binds a source schema element to a target schema element if the two schema elements are semantically equivalent. To date, most research [DDH01, RB01] has focused on computing direct matches. Such simplicity, however, is rarely sufficient, and researchers have thus proposed the use of queries over source schemas to form virtual schema elements to bind with target schema elements [BE03, MHH00]. In this more complicated form, the semantic correspondence is a set of *indirect matches* each of which binds a virtual source schema element to a target schema element through appropriate manipulation operations over a source schema.

We assume that all source and target schemas are described using conceptual-model graphs (a conceptual generalization of XML). We augment schemas with sample data and regular-expression recognizers. For each application, we construct a domain ontology [ECJ+99], which declares the regular-expression recognizers for a set of concepts and relationships among the concepts. We use the regular-expression recognizers and relationships among the concepts to discover both direct and indirect matches between two arbitrary schemas. In this paper, we offer the following contributions: (1) a way to discover many direct and indirect semantic correspondences between a source schema S and a target schema T and (2) experimental results of our implementation to show that our approach to schema matching performs as well (indeed better) than other approaches for direct matches and also performs exceptional well for the indirect matches with which we work. The cost for this increased performance is the development of a domain ontology for a particular application. The benefits, as demonstrated in the experimental results, appear to justify the cost to develop the domain ontology. We present the details of our contribution as follows. Section 2 explains the internal representation of the input and output for schema matching. Section 3 describes the schema-matching technique by applying a domain ontology to discover both direct and indirect matches. Section 4 give an experimental result for a data set used in [DDH01] to demonstrate the contribution of applying domain ontologies to schema matching. In Section 5 we summarize, consider future work, and draw conclusions.

*This material is based upon work supported by the National Science Foundation under grant IIS-0083127.



(a) Schema 1 (b) Schema 2

Figure 1: Conceptual-model graphs for Schema 1 and Schema 2

2 Internal Representation

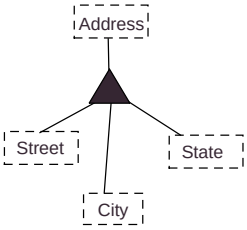
We use conceptual graphs to represent both the target schema and the source schemas as conceptual-model specifications. Each conceptual schema has an *object/relationship-model instance* that describes sets of objects, sets of relationships among objects, and constraints over object and relationship sets. An object set contains either data values or object identifiers, which we respectively call a *lexical object set* or a *nonlexical object set*. A relationship set contains tuples of objects representing relationships connecting object sets. Figure 1, for example, shows two schema graphs. In a schema graph we denote lexical object sets as dashed boxes, nonlexical object sets as solid boxes, functional relationship sets as lines with an arrow from domain object set to range object set, and nonfunctional relationship sets as lines without arrowheads. For either a target or a source schema, we use an object/relationship-model instance to represent schema-level information in our approach for schema mapping. An optional component of a conceptual schema is a set of *data frames*, each of which describes the data of a lexical object set. A data frame is like a type which describes data instances, but can be much more expressive. A data-frame description can be as simple as a list of potential values for an object set and can be as complex as a regular-expression specification that represents values for the object set. For target and source schemas in this paper, data frames are lists of actual or sample values.

In addition to the schema- and instance-level information available from the input source and target schemas, for a particular application domain, we can specify a domain ontology [ECJ$^+$99], which includes a set of concepts and relationships among the concepts, and associates with each concept a set of regular expressions that matches values and keywords expected to appear for the concept. Then using techniques described in [ECJ$^+$99], we can extract values from sets of data for source and target elements and categorize their data-value patterns based on the expected values and keywords declared for application concepts. The derived data-value patterns and the declared relationship sets among concepts in the domain ontology can help discover both direct and indirect matches for schema elements. Figure 2 shows three components in a real-estate domain ontology, which we used to automate matching of the two schemas in Figure 1 and also for matching real-world schemas in the real-estate domain in general. The three components include an address component specifying $Address$ as potentially consisting of $State$, $City$, and $Street$;[1] a phone component specifying $Phone$ as a possible superset of $Day\ Phone$, $Evening\ Phone$, $Home\ Phone$, $Office\ Phone$, and $Cell\ Phone$;[2] and a lot-feature component specifying $Lot\ Feature$ as a possible superset of $View$ and $Lot\ Size$ values and individual values $Water\ Front$, $Golf\ Course$, etc.[3] Behind a dashed box (or individual value), a regular-expression recognizer [ECJ$^+$99] describes the expected values and keywords for a potential application concept. The ontology explicitly declares that (1) the expected values in $Address$ match with a concatenation of the expected values for $Street$, $City$ and $State$; (2) the set of values associated with $Phone$ is a superset of the values in $Day\ Phone$, $Evening\ Phone$, $Home\ Phone$, $Office\ Phone$, and $Cell\ Phone$; and (3) the set of values associated with

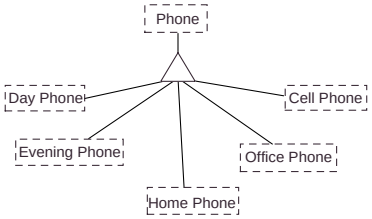
[1] Filled-in (black) triangles denote aggregation ("part-of" relationships).

[2] Open (white) triangles denote generalization/specialization ("ISA" supersets and subsets).

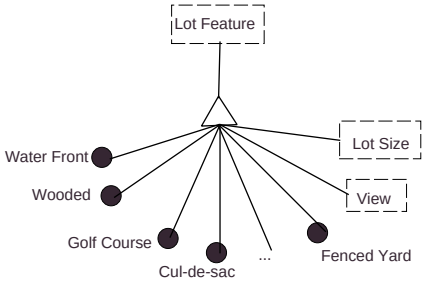
[3] Large black dots denote individual objects or values.



(a) Address



(b) Phone



(c) Lot Feature

Figure 2: Application domain ontology (partial)

$Lot\ Feature$ is a superset of the values associated with the set of $View$ values, the set of $Lot\ Size$ values, the singleton-sets including $Water\ Front$, $Golf\ Course$, $Wooded$, $Fenced\,Yard$, $Cul-de-sac$, etc.

For any schema H, which is either a source schema or a target schema, we let Σ_H denote the union of object sets and relationship sets in H. Our solution allows a variety of source derived data, including missing generalizations and specializations, merged and split values, and transformation of attributes with Boolean indicators into values and vice versa. Therefore, our solution "extends" the source schema elements in Σ_H to include view schema elements, each of which we call a *virtual* object or relationship set. We let V_H denote the extension of Σ_H with derived, virtual object and relationship sets. We consider a source-to-target mapping between a source schema S and a target schema T as a function f_{ST}. The domain of f_{ST} is V_S, and the range of f_{ST} is Σ_T. Thus we can denote a source-to-target mapping as a function $f_{ST}(V_S) \rightarrow \Sigma_T$. Intuitively, a source-to-target mapping represents an one-to-one mapping between a view-augmented source schema and a target schema.

3 Matching Technique

Provided with the domain ontology described in Figure 2 and a set of data values for elements in Schema 1 in Figure 1(a) and Schema 2 in Figure 1(b), we can discover indirect matches as follows. (We first introduce the idea with examples and then more formally explain how this works in general.)

1. *Merged/Split Values*. Based on the $Address$ declared in the ontology in Figure 2, the recognition-of-expected-values technique [ECJ+99] can help detect that (1) the values of $address$ in Schema 1 of Figure 1(a) match with the ontology concept $Address$, and (2) the values of $Street$, $City$, and $State$ in Schema 2 of Figure 1(b) match with the ontology concepts $Street$, $City$, and $State$ respectively. Thus, if Schema 1 is the source and Schema 2 is the target, we can use manipulation *Decomposition* operators to split the values for $address$ in the source as the values for three virtual object sets such that the three virtual object sets match with $Street$, $City$, and $State$ respectively in the target. If we let Schema 2 be the source and Schema 1 be the target, based on the same information, we can identify an indirect match that declares a virtual object set derived by applying a manipulation *Composition* operator over the source to merge values in $Street$, $City$, and $State$ to directly match with $address$ in the target.[4]

2. *Superset/Subset Values*. Based on the specification of the regular expression for $Phone$, the schema elements $phone\_day$ and $phone\_evening$ in Schema 1 of Figure 1(a) match with the concepts $Day\ Phone$ and $Evening\ Phone$ respectively, and $Phone$ in Schema 2 of Figure 1(b) also matches with the concept $Phone$. $Phone$ in the ontology explicitly declares that its set of expected values is a superset of the expected values of $Day\ Phone$ and $Evening\ Phone$. Thus we are able to identify the indirect matching schema elements between $Phone$ in Schema 2 and $phone\_day$ and $phone\_evening$ in Schema 1. If Schema 1 is the source and Schema 2 is the target, we can apply a manipulation *Union* operator over Schema 1 to derive a virtual $Phone'$ whose values are a superset of values in $phone\_day$ and $phone\_evening$. Thus $Phone'$ can directly match with $Phone$ in Schema 2. If Schema 2 is the source and Schema 1 is the target, we may be able to recognize keywords such as *day-time*, *day*, *work phone*, *evening*, and *home* associated with each listed phone

[4]When applying the manipulation operations over sources in data-integration applications, the data-integration system requires routines to merge/split values so that correctly retrieving data from sources.

in the source. If so, we can use a manipulation *Selection* operator to sort out which phones belong in which specialization (if not, a human expert may not be able to sort these out either).

3. *Object-Set Name as Value*. In Schema 2 of Figure 1(b) the features $Water\_front$ and $Golf\_course$ are object-set names rather than values. The Boolean values "Yes" and "No" associated with them are not the values but indicate whether the values $Water\_front$ and $Golf\_course$ should be included as description values for $location\_description$ of $house$ in Schema 1 of Figure 1(a). Because regular-expression recognizers can recognize schema element names as well as values, the recognizer for $Lot\ Feature$ recognizes names such as $Water\_front$ and $Golf\_course$ in Schema 2 as values. Moreover, the recognizer for $Lot\ Feature$ can also recognize data values associated with $location\_description$ in Schema 1 such as "*Mountain View*", "*City Overlook*", and "*Water-Front Property*". Thus, when Schema 1 is the source and Schema 2 is the target, whenever we match a target-schema-element name with a source $location\_description$ value, we can declare "Yes" as the value for the matching target concept by applying a manipulation *Boolean* operator over the $location\_description$ value. If, on the other hand, Schema 2 is the source and Schema 1 is the target, we can declare that the schema element name should be a value for $location\_description$ for each "Yes" associated with the matching source element by applying a manipulation *DeBoolean* operator.

We now more formally describe these three types of indirect matches. Let c_i be an application concept, such as $Street$, and consider a concatenation of concepts such as $Address$ components. Suppose the regular expression for concept c_i matches the first part of a value v for a schema element and the regular expression for concept c_j matches the last part of v, then we say that the concatenation $c_i \circ c_j$ matches v. In general, we may have a set of concatenated concepts C_s match a source element s and a set of concatenated concepts C_t match a target element t. For each concept in C_s or in C_t, we have an associated hit ratio. Hit ratios give the percentage of s or t values that match (or are included in at least some match) with the values of the concepts in C_s or C_t respectively. We also have a hit ratio r_s associated with C_s that gives the percentage of s values that match the concatenation of concepts in C_s, and a hit ratio r_t associated with C_t that gives the percentage of t values that match the concatenation of concepts in C_t. To obtain hit ratios for Boolean fields recognized as schema-element names, we distribute the schema-element names over all the Boolean fields that have "Yes" values.

We decide if s matches with t directly or indirectly by comparing C_s and C_t when the hit ratios r_s and r_t are above an accepted threshold. If C_s equals C_t, we declare a *direct* match (s, t). Otherwise, if $C_s \supset C_t$ ($C_s \subset C_t$), we derive an *indirect* match (s, t) through a *Decomposition* (*Composition*) operation. If both C_s and C_t contain one individual concept c_s and c_t respectively, and if the values of concept c_s (c_t) are declared as a subset of the values of concept c_t (c_s), we derive an *indirect* match (s, t) through a *Union* (*Selection*) operation. When we have schema-element names as values, distribution of the name over the Boolean value fields converts these schema elements into standard schema elements with conventional value-populated fields. Thus no additional comparisons are needed to detect direct and indirect matches when schema-element names are values. We must, however, remember the Boolean conversion for both source and target schemas to correctly derive indirect matches.

We compute the confidence value for a mapping (s, t), which we denoted as $conf(s,t)$, as follows. If we can declare a direct match or derive an indirect match through manipulating *Union*, *Selection*, *Composition*, *Decomposition*, *Boolean*, and *DeBoolean* operators for (s, t), we output the highest confidence value 1.0 for $conf(s,t)$. Otherwise, we construct two vectors v_s and v_t whose coefficients are hit ratios associated with concepts in C_s and C_t. To take the partial similarity between v_s and v_t into account, we calculate a VSM [BYRN99] cosine measure $cos(v_s, v_t)$ between v_s and v_t, and let $conf(s,t)$ be $(cos(v_s, v_t) \times (r_s + r_t)/2)$.

Figure 3 shows the matrix containing confidence values computed based on expected values declared in the domain ontology of Figure 2 using Schema 1 in Figure 1(a) as a source schema and Schema 2 in Figure 1(b) as a target schema.[5] The schema elements along the top are source schema elements taken from Schema 1. The schema elements on the left are target schema elements taken from Schema 2. Observe that the technique correctly identifies the indirect matches between $location\_description$ in the source and $Golf\_course$ and $Water\_front$ in the target, between $phone\_day$ and $phone\_evening$ in the source and $Phone$ in the target, and between $address$ and $location$ in the source and $Street$, $City$, and $State$ in the target. Note that in Figure 3 there are several nonlexical object sets whose values are object identifiers in Schema 1 and Schema 2. An *NA* in the matrix denotes that the object identifiers associated with either the source object set in a column or the target object set in a row are not applicable for value analysis. Furthermore, for this example, we did not include the specifications for expected values or keywords of "bedrooms" and "bathrooms" in our domain ontology. The values for $Bedrooms$ and $Bathrooms$ in

[5] In order to make the matrix fit the page, we use several abbreviations of object-set names in the source schema.

the target and the values for $beds$ and $baths$ in the source do not match any concept in the domain ontology. If one set of data values corresponds to the expected values specified for a concept and another set of data values does not correspond to any concept in the ontology, the confidence is 0.0. For example, the confidence $conf(baths, Phone)$ is 0.0 because the values for $Phone$ in the target correspond to the concept *Phone* in the ontology, but the values for $baths$ in the source do not. If neither values of a pair corresponds to any concept specification in the ontology,[6] the entry is *NA*. For example, the *NA* for the pair ($baths$, $Bathrooms$) denotes that the data values for neither $baths$ in the source nor $Bathrooms$ in the target match any concept in the real-estate domain ontology. If the domain ontology is not complete with respect to an application, our approach needs other matching techniques to discover matches that are not discovered through comparing expected values in the domain ontology.

4 Experimental Results

We evaluate the performance of our approach based on three measures: precision, recall and the F-measure, a standard measure for recall and precision together [BYRN99]. We considered a real-world application, *Real Estate*, to evaluate our matching technique. We used a data set downloaded from the LSD homepage [DDH01] for the applications, and we faithfully translated the schemas from DTDs to conceptual-model graphs. The *Real Estate* application has five schemas. We decided to let any one of the schema graphs be the target and let any other schema graph be the source. We decided not to test any single schema as both a target and a source. In summary, we tested 20 pairs of schemas for the *Real Estate* application. In order to evaluate the contribution of the domain ontology-based matching technique, we tested two runs when comparing a source schema and a target schema. In the first run, we considered only matching techniques that compare object-set names and exploit structure properties [XE03]. In the second run, we added our matching techniques based on domain ontologies.

In the 20 pairs of application schemas, the problems of *Merged/Split Values* appear four times, the problems of *Superset/Subset Values* appear 48 times, and the problems of *Object-Set Name as Value* appear 10 times. With all other indirect and direct matches, there are a total of 876 object-set and relationship-set matches. In the first run, the performance reached 73% recall, 67% precision, and an F-measure of 70%. In the second run, which used a real-estate domain ontology, the performance improved dramatically and reached 94% recall, 90% precision, and an F-measure of 92%. By applying the domain ontology, the algorithm successfully found all the indirect matches related to the four problems of *Merged/Split Values* and all the indirect matches related to the 10 problems of *Object-Set Name as Value*. For the problem of *Superset/Subset Values*, the algorithm correctly found all the indirect matches related to 44 of 48 problems and incorrectly declared four extra *Superset/Subset Values* problems. Of these eight, six of them were ambiguous, making it nearly impossible for a human to decide, let alone a machine. In four of the six ambiguous cases there were various kinds of phones for firms, agents, contacts, and phones with and without message features, and in another two cases there were various kinds of descriptions and comments about a house written in free-form text. The two clearly incorrect cases happened when the algorithm unioned (selected) office and cell phones and mapped them to phones for a firm instead of just mapping office phones to firm phones and discarding cell phones, which had no match at all in the other schema.

One obvious limitation to our approach is the need to manually construct an application-specific domain ontology. Our experience in teaching others to use our system suggests that a domain ontology of the kind we use can be created in a few dozen person-hours. This is not inordinately long; indeed, it is comparable to the time it takes to make a training corpus for machine learning. Moreover, since we predefine a domain ontology for a particular application, we can compare any two schemas for the application using the same domain ontology, so that the work of creating a domain ontology is amortized over repeated usage. Further, the domain ontology does not necessarily need to cover all concepts and relationships in the application schemas, even though it can be revised to help discover more direct and indirect matches.

5 Conclusions and Future Work

Based mainly on expected values declared in domain ontologies, we presented a matching technique for automatically discovering many direct and indirect matches between sets of source and target schema elements. We detected indirect matches related to problems such as *Superset/Subset values*, *Merged/Split values*, as well as *Object-Set Names as Value*. Without this technique, the precision and recall results of the experiments we conducted were only in the neighborhood of 70%, whereas with this technique the results increased to over 90%.

Since domain ontologies appear to play an important role in indirect matching, finding ways to semi-automatically generate them is a goal worthy of some additional work. It is possible to use learning techniques to collect a set of informative and representative keywords

[6] We are not able to compare the expected values without the help of the domain ontology.

	MLS	bath.	bed.	cat.	SQ.	location_desc.	basic_features	agent	fax	phone_day	phone_evening	name	location	address
House	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Bathrooms	0.0	NA	NA	NA	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Bedrooms	0.0	NA	NA	NA	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
MLS	1.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Square_feet	0.0	0.0	0.0	0.0	1.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Water_front	0.0	0.0	0.0	0.0	0.0	1.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Golf_course	0.0	0.0	0.0	0.0	0.0	1.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Address	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Agent	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Fax	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	1.0	0.0	0.0	0.0	0.0	0.0
Phone	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	1.0	1.0	0.0	0.0	0.0
Name	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	1.0	0.0	0.0
Street	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
State	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
City	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
Style	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3: Expected-data-values confidence-value matrix

for application concepts in domain ontologies. Thus, without human interaction except for some labeling, we can make use of many keywords taken from the data of the application itself and thus specify some regular-expression recognizers for the application concepts in a semi-automatic way. Furthermore, many values, such as dates, times, and currency amounts are common across many application domains and can easily be shared.

References

[BE03] J. Biskup and D.W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 28(3):169–212, May 2003.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Menlo Park, California, 1999.

[DDH01] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*, pages 509–520, Santa Barbara, California, May 21–24 2001.

[ECJ+99] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.

[MHH00] R. Miller, L. Haas, and M.A. Hernandez. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, pages 77–88, Cairo, Egypt, September 10–14 2000.

[RB01] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.

[XE03] L. Xu and D.W. Embley. Discovering direct and indirect matches for schema elements. In *Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA 2003)*, pages 39–46, Kyoto, Japan, March 26–28 2003.