SeMap: A Generic Mapping Construction System

Ting Wang
College of Computing
Georgia Tech
Atlanta, USA
twang@cc.gatech.edu

Rachel Pottinger
Computer Science Department
University of British Columbia
Vancouver, Canada
rap@cs.ubc.ca

ABSTRACT

Most previous schema mapping works focus on creating mappings in specific data models for data transformation, failing to capture a richer set of possible relationships between schema elements. For example, most schema matching approaches might discover that 'TA' in one schema equals 'grad TA' in another one, even though the relationship can be modeled more accurately by saying that 'grad TA' is a specialization of 'TA'. Deepening the mapping semantics in turn allow richer application semantics. This paper presents and proves the effectiveness of *SeMap*, a system that constructs a complex, semantically richer mapping (including '*Has-a*', '*Is-a*', '*Associates*' and '*Equivalent*' relationship types) that can be used across data models. We achieve this goal by: (1) exploiting semantic evidence for possible matches; (2) finding a globally optimal match assignment; (3) identifying the relationship embedded in the selected matches. We implemented our semantic matching approach within a prototype system, *SeMap*, and showed its accuracy and effectiveness.

1. INTRODUCTION

1.1 Motivation

The fundamental problem in sharing data from multiple sources is the sources' semantic heterogeneity. The key to overcoming the semantic heterogeneity is identifying the semantic correspondences between them. Semi-automatically creating mappings has attracted intensive research in both the database and AI communities (e.g., [7, 5, 13, 17, 25]). The procedure is comprised of two phases, *schema matching* and *mapping construction*. The goal of *schema matching* is to create equivalence correspondences between elements of both schemas. The equivalence correspondences can be one-to-one (1-to-1) matches, e.g., 'class' corresponds to 'course', or complex matches which may contain more than one element in each schema, e.g., 'TA' maps to some combination of 'grad TA' and 'ugrad TA'. These potential correspondences are then transformed into a final mapping in *mapping construction*, where the identified correspondences are built by adding specific semantic information to generate a semantically-rich mapping. For example, one possible final mapping for the above scenario is shown in Figure 1.

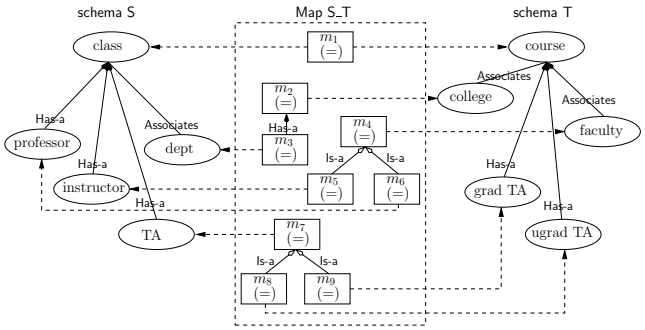

Figure 1: An example of input schemas and output mapping.

As a typical example of mapping construction, Clio [27] uses user-interaction techniques to create SQL-style mappings based on an initial schema match. Such semantic mappings are necessary to transform data. Clio, like most previous mapping constructors, concentrates on creating executable relational and XML mappings; it does not capture the richness of the relationships between elements in a data-model-independent fashion.

Data sources on the web, however, are of many different data models, (e.g., XML, HTML, RDF, Relational, OWL, ASN.1). Combining them in a semantic fashion would yield great benefits, but requires one of two solutions: (1) separate mapping construction algorithms between each pair of data models or (2) creating a system of general, rich relationships that can map between a wide variety of data models. The second option, creating richer, general relationships between schema elements, rather than concentrating on a specific data model, allows a better understanding of the space of possibilities. It also allows better reuse of ideas, rather than creating a separate algorithm for each ensuing data model. After a mapping with such general relationships is constructed, the transformations into a specific data model (e.g., SQL views or XSLT) can be made, thus removing the need of maintaining specific mappings separately. An additional application of a generic mapping is that it can create a uniform interface between domain knowledge (ontologies) and web interface (database schemas), which is helpful for semantic web applications. Furthermore, it can be used for

EDBT'08, March 25–30, 2008, Nantes, France.

knowledge inference when applied to an ontology domain, or be fed into a model management system [2], which aims to solve meta-data problems in a data model neutral fashion.

Figure 1 shows an example generic semantic mapping, where schemas S and T represent 'class' and 'course' respectively. The schemas consist of elements related by *Has-a* or *Is-a* relationships. The generic mapping S_T specifies a rich set of semantic relationships between the elements of S and T, e.g., 'college' of T '*Has-A*' 'dept' of S, while 'instructor' of S '*Is-A*' 'faculty' of T, a common set of relationships [21]. Compared with the equivalence relationships (1-to-1 or complex) considered in previous literature, this relationship classification is semantically richer and more expressive.

This paper constructs such generic semantic mappings. It relies on correspondences between the elements of both schemas, which may be produced by current schema matching techniques. Specifically, we search for a global optimal match assignment from the pool of possible assignments, solving conflicts among the selected matches, and identifying complex relationships between the schema elements, e.g., the '*Has-A*' relationships in Figure 1. Constructing a generic semantic mapping is fundamentally difficult for several reasons:

- The number of initial correspondences to be considered is larger than that in finding simple equivalences, since many *similar* but not *equivalent* correspondences can potentially be of *Is-a*, *Has-a* or *Associates* relationships;
- Identifying implicit relationships in matches is difficult, and exacerbated by requiring data model independence.
- Many schema matching techniques stop at finding the correspondences, but one needs to go a step further to construct a complete mapping, i.e., finding a global optimal mapping assignment.

Mapping construction, like schema matching, inherently cannot be fully automatic. The importance of user feedback is recognized in schema matching research [7, 26]; however, no systematic modeling of user interaction for mapping construction is available to date. One goal of our work is to limit interaction to critical points to help focus user attention and minimize user effort.

This paper describes a prototype system, *SeMap*, which creates a generic, semantic mapping that overcomes the above problems. We choose a graph-based representation similar to that in model management [2], which is expressive enough to accommodate both schemas of many types and other meta-data, such as ontologies.

Specifically, we make the following contributions:

- An architecture to semi-automatically construct generic semantic mappings based on initial correspondences
- Effective extraction of implicit relationships from initial matches based on various types of semantic evidence;
- A novel probabilistic framework that uniformly incorporates uncertainty and semantic constraints and expresses match selection as a mathematical optimization problem;
- Effective modeling of user interaction to help focus user attention and minimize user effort, by detecting critical points where feedback is maximally useful;
- A prototype system embodying the innovations above and a set of experiments illustrating the correctness and effectiveness of our approach. We show that our approach is at least as effective as previous approaches for solving the simpler problem of finding correspondences, even though we are solving a harder problem.

The paper is organized as follows: Section 2 surveys related work. Section 3 formally defines mapping construction and gives an overview of *SeMap*'s architecture. Section 4 describes our mapping construction approach in detail. Section 5 presents the experimental analysis of our approach. Section 6 concludes and presents future work.

2. RELATED WORK

Semi-automatically creating semantic mappings has attracted intensive research in both the database (schema matching and mapping construction) and AI (ontology alignment) communities. In this section, we survey related work in three parts: Section 2.1 surveys techniques for finding mappings based on the relationships they find; Section 2.2 describes techniques used in schema matching and mapping construction, and Section 2.3 discusses ontology alignment.

2.1 Relationship Classification

The relationship types created by matching techniques can be divided into three categories: *equivalent* relationships, *set-theoretic* relationships and *generic* relationships.

Most schema matching, mapping construction, and ontology alignment algorithms to date aim to discover equivalence relationships (e.g., [7, 5, 6, 13, 16, 18, 26]). Equivalence correspondences can be a 1-to-1 match (e.g., 'course' = 'class'), or a complex match (e.g., 'name' = concat('first-name' +'last-name')). Due to the complexity of creating multi-arity (1-to-n or even n-to-m) matches, most schema matching work to date has focused on discovering 1-to-1 equivalence correspondences between schema elements [7, 6, 13, 16, 18, 26]. iMAP [5] identifies 1-to-n correspondences by reformulating schema matching as a search problem. To search effectively, it employs a set of searchers, each discovering specific type of equivalent relationship, e.g., the name of one element in one schema corresponds to the concatenation of that of two elements in the second schema.

While these methods return equivalent correspondences, it is likely that the matches identified by these techniques may be more accurately represented by both equivalent correspondences and the semantically richer relationships that *SeMap* finds, such as the relationship between 'TA' and {'grad TA', 'ugrad TA'} in Figure 1.

Set-theoretic relationship classification regards each schema element as a set and specifies each relationship as one of *equivalence, subsumption, intersection, disjointness* and *incompatibility*. Rizopoulos [23] identifies inter-set relationships by bidirectionally comparing the containment of data instances and meta-data, of different schema elements. Unfortunately, this approach requires that data instances associated with the two schemas are in the same universe (i.e., contain the same values), which is not commonly true in many applications, such as web data integration.

Generic relationships refer to non-equivalent relationships, such as the *Has-a* and *Is-a* relationships in this paper. Few works find generic relationships between schema elements. Giunchiglia and Yatskevich [10] proposed schema-based approach, based on containment relationships of element names. However, their approach works only for identifying *Is-a* relationships — it cannot be generalized to any other relationship type — and requires tree-structured schemas. D. Embley et al. [9] use a domain-specific ontology to find the relationships of *Merge/Split* and *Superset/Subset*. Such ontologies are usually hard to obtain in real scenarios. Our work

assumes a flexible framework: it leverages the semantic information available in most schemas (e.g., element names) to produce fairly good results, even though additional resources can be easily incorporated within to improve the quality of results. However, *SeMap* can be extended to use such ontologies to improve the quality of results.

2.2 Schema Matching Techniques

Schema matching, mapping construction, and ontology alignment research use a plethora of techniques to semi-automatically find semantic matches. These techniques can be classified into rule-based and learning-based solutions [7]. Rule-based approaches [10, 16, 18] exploit schema-level information (e.g., element name, data type) to discover similar schema elements. For example, *Cupid* [16] is a rule-based matching system that categorizes elements based on element names, data types and domains; *Similarity flooding* [18] measures pairwise similarity by propagating similarity from fixed points according to the schema structures.

As pointed out in [8], rule-based techniques have both desirable features and drawbacks. On the positive side, they are computationally efficient, assume no external semantic resources, and thus are applicable for a large range of applications. However, since hard-coded rules cannot effectively extract hidden semantic information from noisy raw data, such as data instance of schema elements, these techniques may lose valuable semantic information.

Motivated by the drawbacks of rule-based matching methods, learning-based solutions have been proposed. These methods outperform the rule-based solutions, though with the cost of a training phase. For example *LSD* [7] employs Naive Bayes learning over data instances and also exploits XML structure information; *iMAP* [5] uses the description of elements in addition to other schema information.

In addition to schema level and data instance level information, several types of external resources have also been considered to improve the matching quality. For example, assuming a domain-specific ontology is available, one technique is to first map schemas/ontologies into the ontology, then construct the matches based on the relationships inherent in that ontology [9]. Some recent works exploit similarity with other schemas or past matching results to improve current ones [7, 6, 16]. However, it is not always practical to have such external resources, particularly since such these resources must be domain-specific to be effective.

Finally, other works exist for deriving schema mapping by assembling schema matches, with performance guarantees. For example, [3] proposes a reduction from the assembling problem to Independent Set. These papers also do not consider generic relationships of the type that we consider.

2.3 Ontology Alignment Techniques

To the best of our knowledge, though an ontology may have complex relationships, most previous ontology alignment work focuses on finding semantically equivalent concepts or one specific type of relationship, e.g., *Is-a* in *FCA-Merge* [25]. We briefly present some typical work ontology alignment work; see [13] for a more complete survey.

OntoMorph [4] translates symbolic knowledge between different knowledge representations through user-provided transformation rules. *Prompt* [19] proposed an ontology alignment mechanism that finds corresponding concepts by refining an initial mapping (pairs of anchors) given by either users or simple linguistic matching approaches. *Prompt*'s philosophy is similar to that of *Similarity Flooding* [18]. *FCA-Merge* [25] is an alignment technique that depends on external resources to find *Is-a* relationships between concepts. However, since the formal context is built upon the generalization/specialization hierarchy of the concepts, this approach could not be extended to other relationships, such as *Has-a*. Moreover, FCA-Merge requires domain-specific documents, which is not always feasible.

In summary, to the best of our knowledge, most previous schema matching, mapping construction, and ontology alignment work focuses on finding one-to-one equivalence relationships between two schemas. Little work is done in identifying multiple types of complex relationships. In the following sections, we present *SeMap*, a prototype mapping construction system, which is designed to find generic semantic correspondences.

3. PROBLEM FORMULATION

The overall goal of our schema matching and mapping construction system, *SeMap*, is to identify generic semantic mappings between two schemas. Specifically, *SeMap* finds mappings where: (1) the matches may be 1-to-1, 1-to-n or n-to-1 (2) the relationship types may be non-equivalence.

In *SeMap*, in addition to *Equivalent* relationships, we consider *Has-a*, *Is-a*, and *Associates* (i.e., is related to in some other fashion, e.g., "Bob" created element "foo"). *Has-a* and *Is-a* are illustrated in Figure 1. These relationships are common across many semantic models [21], and also appear in generic schema manipulation research [2].

An example of a generic semantic mapping is shown in Figure 1, where two schemas represent the concept of '`class`' / '`course`' in different ways. In addition to possibly containing standard equivalence relationships, the mapping contains complex correspondences, such as '`TA`' of schema `S` is a generalization of '`undergrad TA`' and '`grad TA`' of schema `T`. As well, '`department`' of schema `S` is shown to be 'part of' of the '`college`' of schema `T`.

Because we are searching for a data model neutral solution, we adopt the terminology from Model Management [2], and say that we take as input two *models*[1]. A model is a complex design artifact, such as a relational schema, XML schema, or an ontology [2]. A model can be represented as a directed labeled graph (V, E). Specifically, V is the set of nodes, each denoting an element of the schema, (e.g., attributes in relational table, XML schema type definitions). E is the set of binary, directed typed edges over V.

A *mapping*, Map_{S-T}, is a formal description of the semantic relationships between schemas S and T. A mapping is itself a model consisting of a set of *mapping elements* $\mathcal{E}$, and a set of relationships $\mathcal{R}$ on $\mathcal{E}$. The elements of the two schemas are related through *mapping elements*. Each mapping element $e \in \mathcal{E}$ is like any other element in a model. In addition to being the origin or destination of any kind of relationship found in a model, i.e., $\mathcal{R}$, each $e \in \mathcal{E}$ can be the origin of one or more mapping relationships, $M(e, s)$, where $s \in S \cup T$, which specifies that the mapping element e is a semantically equivalent representation of s. Therefore, for $s_1, s_2 \in S \cup T$ s.t. there exist $M(e, s_1)$ and $M(e, s_2)$, then s_1 corresponds to s_2. Figure 1 shows an example of two models with a mapping in between them.

[1]We use *schema* and *model* inter-changeably.

Given this rich mapping structure, generic semantic relationships — not just simple correspondences — can be expressed in in a data model independent fashion as follows: two semantically equivalent elements are represented by one mapping element. The relationship of two mapping elements indicates the relationship between their corresponding schema elements. For example, in Figure 1, the mapping element m_1 corresponds to the elements 'class' and 'course' representing the same concept; the relationship between m_4 and m_5 indicates 'instructor' 'is-a' 'faculty'.

Given the definition of model and mapping, we are now ready to formally define the goal of *SeMap*: *given two models, S and T, find generic semantic relationships required to create the mapping S_T between S and T.*

There may be some optional inputs to the matching process, specifically (1) an initial mapping S_T' which provides an initial set of correspondences and needs to be refined by the process; and (2) external semantic resources r used by the matching process, e.g., domain-specific thesauri.

3.1 Semantic Resources

The semantic resources used by matching techniques can be categorized as *internal resources* (Section 3.1.1), which are contained in the input schemas or associated data instances, and *external resources* (Section 3.1.2), which come from outside the schemas or data instances.

3.1.1 Internal Resources

The semantic resources of the input schemas include both element-level information, which refers to the information stored at each schema element (e.g., element name, data type, structure) and structure-level information, which refers to the information contained in the relationships between schema elements (e.g., relationship type, constraints). We consider the following element-level information:

- **Element name (label):** The name (label) provides a first layer of semantic evidence.
- **Element type:** Elements that contain data are usually associated with a type. Semantically similar schema elements usually have the same or compatible data types.
- **Data instances:** As discussed in Section 2, data instances provide valuable information that cannot be found in schemas. For example, consider element 'phone' of type String. Looking at its data instances may show that its exact format is 'xxx-xxx-xxxx', which is not reflected in its type. The distribution of data instances is also useful in identifying similar schema elements, especially when the element names are obscure, e.g., A_1 and B_2 [14].

SeMap takes advantage of each of these element-level information sources above. However, it does not require that any of them exist, except that each element must be named.

In addition to the above *element-level* information, *SeMap* also considers two types of *structure-level* information:

- **Relationship Type:** The type of edge between schema elements can be leveraged. If two elements are semantically similar, elements having the same relationship with them are likely to be semantically related.
- **Constraints:** Each edge can have constraints, including (1) cardinality, and (2) key properties of elements, (e.g., unique, primary)

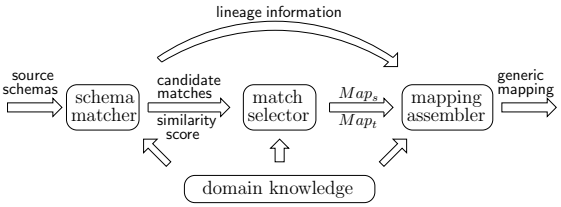

Figure 2: *SeMap*'s basic system architecture

3.1.2 External Resources

Previous work on matching techniques has shown that internal semantic evidence is usually insufficient for achieving high quality matching results; some additional external resources should be leveraged to improve the matching quality.

SeMap, considers two types of external resources:

- **Thesaurus:** *SeMap* uses WordNet as its thesaurus. WordNet organizes more than 80,000 words into sets of synonyms (synsets). Synsets are organized into a hierarchy based on their semantic relationships (e.g., "is a specialization of"). WordNet is considered one of the most powerful tools in computational linguistics, and is used in several matching applications [10].
- **Ontology:** Ontologies, especially domain-specific ontologies, are powerful tools in discovering similar elements and identifying implicit relationships. However they are not always obtainable.

3.2 Approach Overview

This section presents an overview of our generic matching system, *SeMap*. *SeMap* takes as input two schemas, S and T, along with optional external resources, and produces, S_T, a generic mapping between S and T.

To identify generic semantic relationships between schema elements, *SeMap* must not only find correspondences, but also extract the implicit relationship types. Figure 2 shows *SeMap*'s basic architecture. The first phase, *schema matching* (Section 3.2.1), identifies the candidate matches (correspondences). Note that most previous work focuses on finding correspondences of *Equivalent* relationships, while *SeMap* also finds the correspondences of other relationships, which significantly increases the difficulty. The second step, *match selection* (Section 3.2.2), selects a subset of candidate matches to form the complete mapping. This phase uses a novel probabilistic framework that incorporates both match uncertainty and domain constraints, and implements match selection as a constrained optimization problem. Finally, the third phase, *mapping assembling* (Section 3.2.3), determines the implicit relationship types of the selected matches. As will be shown, although the relationship types are determined in the mapping assembler, all system components require augmentation to assist the mapping assembler in its task. Again, to the best of our knowledge, previous work focuses either on finding *Equivalent* relationships or one specific type of relationship, (i.e., *Is-a*), making *SeMap* the first system to exploit different types of semantic information to identify multiple generic semantic relationships.

3.2.1 Schema Matcher

The schema matcher takes two schemas as input and generates a set of initial matches showing the correspondence between the elements of the schemas, i.e., it constructs the

input to the mapping construction problem. For each initial match, it also produces its similarity score indicating its uncertainty and lineage information recording how it is identified. This lineage information is required to identify generic semantic relationships in the mapping assembler.

The schema matcher consists of a set of base matchers, as in many composite approaches (see [20] for a survey). A base matcher looks at some aspect of the model/schema and generates a series of candidate matches. Each candidate match shows a 1-to-1 or 1-to-n correspondence of schema elements. *SeMap* currently includes a *label matcher*, *sense matcher*, *type matcher*, *structure matcher* and *data instance matcher* as its base matchers. Additionally, *SeMap*'s flexible architecture makes it easy to include other base matchers.

The generic semantic match may be either 1-to-1 or 1-to-n (i.e., one source element corresponds to multiple target elements). However, since the focus of our work is to identify the relationship between schema elements, rather than finding the exact transformation rules, we consider 1-to-n matches as n 1-to-1 matches, which significantly simplifies the searching complexity. More detailed discussion of schema matcher is described in Section 4.1.

3.2.2 Match Selector

The match selector is responsible for assigning each schema element to a match and outputs the best match assignments, Map_s and Map_t, for the elements of S and T respectively. For example, as shown in Figure 1, Map_s may be: 'class' : 'course', 'professor' : 'faculty', 'dept' : 'college', 'instructor' : 'faculty', and 'TA' : 'grad TA', while Map_t may consist of: 'course' : 'class', 'college' : 'dept', 'ugrad TA' : 'TA', 'grad TA' : 'TA' and 'faculty' : 'professor'. These two mappings are then merged to form the final mapping as shown in Figure 1.

The match selector searches for the best global assignments from the set of candidate matches.

The functionality of the match selector is similar to the constraint handler described in iMAP [7], which applies a set of domain constraints to select a subset of candidate matches. However, we propose a novel statistical model to incorporate both match uncertainty and domain constraints in the same framework, and express the match selection as a constrained optimization problem, for which an effective solution is available.

User interaction can greatly improve prediction accuracy. *SeMap* applies active learning to identify the points in selecting the matches where user interaction is maximally useful, so that user effort can be significantly reduced. Section 4.2.2 describes the match selector in greater detail.

3.2.3 Mapping Assembler

The mapping assembler combines Map_s and Map_t, identifies the relationships embedded in the selected matches, and assembles them into a generic semantic mapping that includes richer semantic relationships. For example, in our ongoing example, consulting the lineage information that 'TA' is a substring of that of 'grad TA', results in identifying the elements as being in an *Is-a* relationship.

The mapping assembler combines the best match assignments Map_s and Map_t to form a final mapping. Then for each match in the final mapping, the mapping assembler extracts related semantic evidence from lineage information to identify the implicit relationship type. A detailed discussion of the mapping assembler is described in Section 4.3.

4. SEMAP SYSTEM

In this section, we present the technical details of our schema matching and mapping construction system, *SeMap*. As shown in Section 3, the overall architecture of *SeMap* consists of three main parts, the schema matcher (Section 4.1), the match selector (Section 4.2), and the mapping assembler (Section 4.3), which are responsible for finding correspondences (candidate matches), selecting a subset of candidate matches to form a mapping, and identifying the implicit relationships respectively. To illustrate how *SeMap* produces generic semantic matches, we show the matching process of the example in Figure 1.

4.1 Schema Matcher

Previous research shows that effective schema matching requires a combination of base techniques (e.g., linguistic matching and structure matching) [20]. Hence *SeMap* uses a composite approach to detect potential correspondences.

4.1.1 Base Matchers

The components of a schema matcher are a set of pre-existing matchers that exploit available information, and a framework — such as COMA [6] and iMAP [5] — to incorporate them. Discovering initial matches can be modeled as a search using the base matchers, each of which exploits a meaningful subset of the space [5]. Base matchers can be classified as either *element-level* matchers, or *structure-level* matchers. The former computes mapping elements between individual nodes, and the latter computes mapping elements between subgraphs. Since the base matchers are not *SeMap*'s focus, we only briefly describe the matchers that *SeMap* uses and refer the reader to [20, 17, 10] for more thorough treatment. The selection of this specific set is based the semantic evidence available in most types of schemas. The base matchers that are considered in *SeMap* include:

- **sense matcher:** The sense matcher populates the tokens of each element with their senses (atomic meanings of a word or expression) in WordNet. An element's senses are the union of the senses of all its tokens. By comparing their senses, one can evaluate the similarity of the corresponding elements. Since *SeMap* lacks external knowledge of the context, it considers all senses of a word.
- **label matcher:** The label-based matcher finds semantically related elements by evaluating the syntactic similarity of their labels (names). *SeMap* applies a standard set of techniques to improve label matching, which includes case normalization, soundex elimination (e.g., '4U' and 'for you'), tokenization, and stopword elimination.
- **type matcher:** The type of schema elements carries the information of data type (e.g, string), value domains (e.g., range of [1, 12]), and key characteristics (e.g., unique). The type matcher determines the similarity of schema elements based on such semantic information.
- **data instance matcher:** Instances contain valuable information on frequency and other data characteristics, e.g., phone numbers have similar formats. Specifically, *SeMap* considers the data's format and distribution.
- **structure matcher:** *SeMap*'s structure matcher is *Similarity Flooding* [18]. Based on an initial mapping generated by other techniques, similarity flooding propagates the similarity of mapped elements to adjacent ones which have similar relationship to the mapped elements.

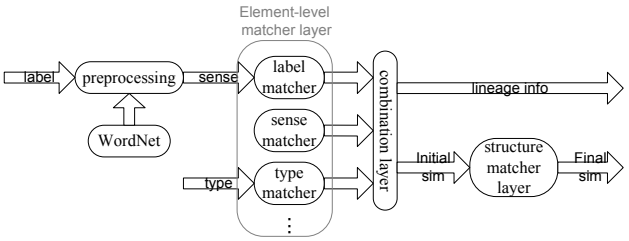

Figure 3: *SeMap*'s schema matcher architecture consists of three layers: base matchers, a combination layer and a structure matcher.

4.1.2 Similarity Score and Lineage Information

In addition to the set of initial matches (correspondences), the schema matcher also provides the following information: (1) similarity score. Each candidate match m is associated with a similarity score $\text{Sim}(m) \in [0, 1]$, indicating the belief about its uncertainty, with 1 meaning perfectly certain; (2) lineage information. For each initial match, *SeMap* records the flow of information, such as assumptions and domain knowledge. From this lineage information, *SeMap* traces how this match is generated. The mapping assembler (Section 4.3) uses the lineage information to discover the generic relationships that other schema matchers do not find.

Similarity evaluation has been discussed in previous work [7, 5, 6]. *SeMap*'s lineage information recording is similar in spirit to the explanation module in [5], which provides explanations for user questions posed on the generated matches, such as explaining existing match, absent match, or match ranking. However *SeMap* uses this information to distinguish different semantic relationships as described in the following sections.

4.1.3 Schema Matcher Architecture

As in previous approaches, *SeMap* combines the base matchers to improve matching quality. The schema matcher architecture is illustrated in Figure 3. It consists of three major layers: *element-level matchers*, a *combination layer* and a *structure-level matcher*, along with a preprocessing step.

The combination phase integrates the similarity scores produced by the element-level matchers to form a unified score. The combination scheme can vary, as long as the final result is well normalized, i.e., in the range of $[0, 1]$. *SeMap* follows a simple linear combination scheme. Each element-level matcher em is associated with a weight w_{em}, and the similarity score Sim_{em} produced by em is damped by w_{em} when computing the unified score. Formally, the similarity score $\text{Sim} = \sum_{em} w_{em}\text{Sim}_{em}$, where $\sum_{em} w_{em} = 1$. The weight parameters must be tuned in order to achieve the optimal result, which is not the focus of this paper. For an automatic learning approach, see [24].

The unified similarity scores generated in the second phase are fed into the structure-level matcher as an initial mapping. The structure matching is then performed to produce the final similarity score for each pair of elements.

4.2 Match Selector

The match selector is the second major part of *SeMap*'s architecture (Figure 2), and is application specific in our current implementation. Given the pool of initial matches and associated similarity scores, the match selector searches for a global optimal match assignment that satisfies a set of domain constraints, e.g., in the case of Figure 1, the user may specify that each class has only one instructor. The constraints are specified manually for each application, based on users' domain knowledge. In the current implementation, we only consider the cardinality constraints, e.g., the restriction on the number of elements an element can be matched with.

It also contains a user interaction module which exploits user feedback to improve the mapping quality. In this section, we introduce a novel probabilistic framework that allows *SeMap* to uniformly express the match uncertainty and domain constraints. Match selection can then be transformed as an optimization problem (Section 4.2.2). Within this framework, we reduce the need of user interaction and focus user attention by identifying critical points where user feedback is maximally helpful (Section 4.2.3).

Most prior works studied 1-to-1 correspondences. They either first apply constraints to narrow the pool of possible mappings, and then transform match selection to a stable marriage problem [18], or assume that the initial matches are mutually independent, and seek a trade-off between uncertainty and constraint satisfaction [7]. Discovering semantic relationships requires considering more complex matches, so the match selector must be more complex.

Next we present a novel probabilistic framework incorporating both the similarity scores and the semantic constraints. This allows us in turn to model match selection as an optimization problem. Additionally, this framework reduces the need for user interaction and focuses user attention by identifying critical points where user feedback is maximally helpful. Section 4.2.1 introduces the representation of this probabilistic framework; Section 4.2.2 discusses the match selection problem and its solution within this framework, and Section 4.2.3 presents the user interaction scheme.

4.2.1 Representation

This section shows how to incorporate match uncertainty and semantic constraints in a probabilistic framework. Our representation is inspired by [15], which proposes using a probabilistic model to express this uncertainty. We extend this framework to support domain constraints. Hence our work contributes to schema matching in addition to mapping construction.

Formally, each schema element e is associated with a set of initial matches $\mathcal{M}_e$, and can be assigned to a match $m \in \mathcal{M}_e$. The probability of assigning e to match $m^* \in \mathcal{M}_e$ is defined as $P(e \leftarrow m^*) = \frac{\text{Sim}(m^*)}{\sum_{m \in \mathcal{M}_e} \text{Sim}(m)}$, where $\text{Sim}(m)$ is the similarity score of match m provided by the schema matcher. Intuitively, this represents the preference for matches with low uncertainty. It is easy to verify that this model is well normalized: $\sum_{m \in \mathcal{M}_e} P(e \leftarrow m) = 1$.

Each mapping consists of a set of matches, some of which may violate the user-specified domain constraints. Specifically, we associate each constraint c with a *penalty function* $\Pi_c(\mathcal{M})$, which counts the number of violations of c in the assignment $\mathcal{M}$. Each constraint c is also assigned a weight α_c, a user-defined metric of the strictness of constraint. In our current implementation, α_c is set to 1 (i.e., strict) by default. The weights can be hard coded or learned from known mappings [16]. Given a set of constraints $\mathcal{C}$, the probability of a set of elements $\mathcal{E}$ taking match assignment $\mathcal{M}$ (where

$e_i \in \mathcal{E}$ is assigned to $m_i \in \mathcal{M}$) is defined as:

$$P(\mathcal{E} \leftarrow \mathcal{M}|\mathcal{C}) = \frac{1}{Z}[\prod_{e_i \in \mathcal{E}} P(e_i \leftarrow m_i)]^{(\sum_{c \in \mathcal{C}} -\alpha_c \Pi_c(\mathcal{M}))} \quad (1)$$

where Z is a normalization constant, in order to guarantee that $P(\mathcal{E} \leftarrow \mathcal{M}|\mathcal{C}) \in [0,1]$.[2] In this equation, the total likelihood of the match assignment for $\mathcal{E}$ is captured in $\prod_{e_i \in \mathcal{E}} P(e_i \leftarrow m_i)$, while the exponential part represents the penalty of violating the constraints. The more constraints that are violated, the lower the probability is.

4.2.2 Bidirectional search

Since the joint probability of the selected matches measures the uncertainty of the mapping, for a set of schema elements, $\mathcal{E}$, and constraints, $\mathcal{C}$, match selection amounts to finding the match assignment, $\mathcal{M}$, that maximizes the probability $P(\mathcal{E} \leftarrow \mathcal{M}|\mathcal{C})$.

It is infeasible to consider all possible combinations from the domain of $\mathcal{E}$ to find the optimal match assignment. For example if each $e \in \mathcal{E}$ is associated with k initial matches, one potentially has to consider $|\mathcal{E}|^k$ combinations. It is highly likely that $\mathcal{E}$ comprises a series of disjoint subset $\mathcal{E}_1, \mathcal{E}_2, \ldots$, which are mutually independent. For example, *SeMap* considers the source and target schemas as the bipartitions of a bipartite graph, where vertices are linked by matches. Vertices that are connected belong to the same subset, which can divide the graph into disjoint subsets. Hence we optimize these independent parts separately, that is $\max_{\mathcal{M}} P(\mathcal{E} \leftarrow \mathcal{M}|\mathcal{C}) \equiv \max_{\mathcal{M}_1, \mathcal{M}_2, \ldots} \prod_i P(\mathcal{E}_i \leftarrow \mathcal{M}_i|\mathcal{C})$

However, it is possible that $\mathcal{E}_i \in \mathcal{E}$ can still be quite large, requiring more efficient solutions, including several effective graphical model optimization algorithms proposed in the machine learning community [12], e.g., *graph-cuts*, and heuristic searching methods, e.g., A^*. *SeMap* applies the A^* [11] to this optimization problem.

The discussion above focuses on match selection for an arbitrary set of elements $\mathcal{E}$. In the context of complex match, which is necessary to find semantic relationships, the perspectives from source and target schemas could be significantly different. For example, in Figure 1, starting from Schema T, one may discover that '**grad TA**' is a specialization of '**TA**', but miss that '**TA**' consists of both '**grad TA**' and '**ugrad TA**'. This necessitates matching elements for both source and target elements simultaneously, i.e., a "bidirectional" perspective.

To simplify the problem, *SeMap* distinguishes the perspectives of source and target schemas, i.e., treat the match selection for source and target elements separately. Thus, instead of searching the best matches for source elements $\mathcal{E}_s$ or target elements $\mathcal{E}_t$ separately, as in previous work, e.g., iMAP [5], *SeMap* runs the optimization algorithms for both source elements $\mathcal{E}_s$ and target elements $\mathcal{E}_t$, i.e., *bidirectional search*. The result of the bidirectional search are two sets of matches Map_s and Map_t. An example of Map_s and Map_t is shown in Figure 4. Map_s and Map_t are then be merged to form a final complete mapping (Section 4.3). For the possible conflicts in Map_s and Map_t, the one with higher similarity score and less constraints violation is selected. Details of the merging process are given in (Section 4.3).

4.2.3 Modeling user interaction

[2]Formally, $Z = \sum_{\mathcal{M}} \prod_{e_i \in \mathcal{E}} P(e_i \leftarrow m_i)^{(\sum_{c \in \mathcal{C}} -\alpha_c \Pi_c(\mathcal{M}))}$

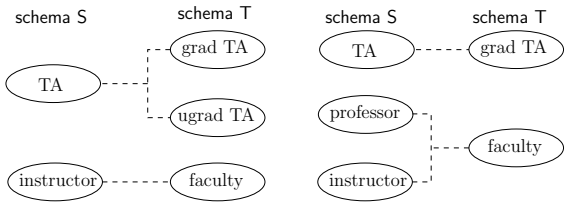

Figure 4: Partial match assignments from the perspectives of source and target schemas respectively.

Capturing user feedback is crucial for improving mapping quality, particularly for the more complex matches that *SeMap* considers. Though user feedback is used in error correction and mapping refinement [27], modeling user interaction is under-studied. This section shows how we extended *SeMap*'s probabilistic framework to model user interaction.

The key to modeling user interaction is identifying points where feedback is maximally helpful, so that user workload is minimized. We present an *active learning* solution to simulating the effect of a user's selection of a candidate match for schema elements. This is an extension of the approach in [26], where active learning is applied to learning the optimal parameters in matching web interfaces. In *SeMap*, elements are ranked based on their potential information value; user feedback is asked for those with the highest values.

A natural measurement of information value is *entropy*. Intuitively, entropy measures randomness or uncertainty. The higher the uncertainty, the larger the entropy value. Formally, for a variable x with distribution $P(x)$, its entropy $\mathbb{H}(x)$ is defined as $-\sum_x P(x) \log P(x)$. In match selection, the goal is to have a single (possibly empty) match assignment for each schema element. For example, in Figure 4, the assignment for **TA** in schema S is {**gradTA**, **ugradTA**}. The higher the uncertainty of the selection, the higher the entropy. However, this does not consider an assignment's influence on match assignment of other schema elements. This deficiency of entropy leads to another metric, *mutual information* for variables x and y is: $\mathrm{I}(x,y) = \mathbb{H}(x) - \mathbb{H}(x|y) = \mathbb{H}(y) - \mathbb{H}(y|x)$ In our scenario, consider the assignment of two elements e and e' as two variables; the mutual information $I(e,e')$ indicates the reduction of the uncertainty of the assignment of e given the assignment of e'. For example, in Figure 1, since the assignments of **professor** and **instructor** affect each other significantly, it can be expected that they have a high mutual information value.

Hence *SeMap* seeks the element having the maximum mutual information with other elements, i.e., the one whose match assignment will best identify matches for others. Formally, *SeMap* seeks the most informative element e that maximizes $\sum_{e' \in \mathcal{E}} I(e,e')$. The joint probability of $P(e,e'|\mathcal{C})$ can be calculated according to Equation 1, which will yield the most informative element.

Once the most informative element, e, has been selected, it should be disambiguated by the user. This is the key tenant of active learning: the system should prompt the user to disambiguate cases where the user's input will provide the most information to the system. Once the user's selection for e is obtained, the assignment needs to be updated accordingly. Based on the new belief about e, the joint probability of variables is updated accordingly. Note that only the joint probability of the subset of $\mathcal{E}$ involving e needs to be up-

dated. The interaction process repeats until a satisfactory threshold is reached. The effectiveness of this user interaction modeling is empirically proved in our experiments: for most datasets, providing correct match assignment for only 15-20% of elements led to an globally optimal assignment.

4.3 Mapping Assembler

The bidirectional search (Section 4.2.2) produces two sets of matches Map_s and Map_t, representing the correspondences from the perspectives of source and target schemas respectively. The next, and final, phase of *SeMap* is the mapping assembler. In the mapping assembler, Map_s and Map_t are combined to form a final generic semantic mapping. Specifically, this process selects an optimal set of matches from both mappings (Section 4.3.1); identifies the relationship implicit in the selected matches (Section 4.3.2); and assembles these matches together to form a final, generic semantic mapping (Section 4.3.3).

4.3.1 Combining Map_s and Map_t

By using our novel bidirectional search to create Map_s and Map_t, we have considerably narrowed the search space to be examined in determining semantic relationships (such as the ones in Figure 1). To merge Map_s and Map_t to form a final mapping, we present a heuristic approach, which our preliminary tests have shown to work effectively in practice.

Let $\mathcal{M}$ denote the union of Map_s and Map_t. For each match $m \in \mathcal{M}$, the reward, $\mathrm{R}(m)$ of including m in the final mapping is $\mathrm{Sim}(m)^{(\sum_{c\in\mathcal{C}}\sum_{m'\in\mathcal{M}}-\alpha_c\Pi_c(m,m'))}$where $Sim(m)$ is the similarity score of match m provided by the schema matcher, and $\alpha_c\Pi_c(m,m')$ is the penalty of including both matches m and m', which indicates the conflict between m and m'. Intuitively, this reward function takes account of the similarity score of the match, the constraints it violates together, and other matches in $\mathcal{M}$. *SeMap* then selects a subset of matches with the highest ranking and filters those matches with reward $\mathrm{R}(m)$ lower than certain threshold ϵ.

4.3.2 Identifying relationships

One key step of *SeMap* is identifying the relationship implicit in the selected matches in order to create generic semantic relationships. To the best of our knowledge, there is little prior work on this problem. *SeMap* uses a novel rule-based method to identify the implicit relationships. Because matches are identified by competing semantic evidence, a uniform solution is hard to obtain. Instead, for each type of semantic evidence, we define specific rules to extract the embedded generic relationship. In what follows, we show how to use semantic evidence to identify the four specific relationships, *Equivalent*, *Has-a*, *Is-a* and *Associates*, where the first three relationships have the obvious meanings, and *Associates* means that the elements are related in some weaker fashion (e.g., "Bob" created element "foo").

We classify the semantic evidence into four categories, *schema-level*, *semantic-level*, *instance-level*, and *ontology*. We describe each category in more detail below.

Schema-level evidence includes label, type, and structure information. Generally speaking, schema-level information alone is insufficient to determine the embedded generic relationship. However, it provides support for the results claimed by other semantic evidence. *SeMap* uses the lineage information about why a match was selected (Section 4.1.2) in the following heuristic rules:

- Label. Elements in an *Equivalent* relationship may have similar names; Elements in *Is-a* or *Has-a* relationships tend to have labels with prefix/suffix relationships; For example, given that 'grad-TA' has a suffix of 'TA', 'grad-TA' is likely to be a part or a specification of 'TA'
- Type. Two elements with *Equivalent* or *Is-a* relationships probably share a data type. If one element has a data type as a subcomponent of that of the other, it is likely that they have a *Has-a* relationship
- Structure. The hierarchical structure usually embeds the *part-of* (*Has-a*) relationship, for example, the non-leaf node 'time' may have leaf-nodes 'day', 'mon' and 'year'. While two elements with *Is-a* or *Equivalent* relationships tend not have such structure (one non-leaf, one leaf)

Semantic-level evidence is captured by the lineage information produced by the sense matcher. *SeMap* infers the embedded semantic relationships for two elements X and Y as follows:

- If X and Y share an identical sense, it is highly likely that they are semantically similar or equivalent;
- If any senses of X act as the hypernym of Y (i.e., X is a (kind of) Y) then they may be in an *Is-a* relationship;
- If a sense of X is a hyponym of Y (i.e., X is a (kind of) Y), the two elements are likely in an *Is-a* relationship;
- If any senses X appear as the holonym or meronym Y (i.e., X is a part of Y or Y is a part of X), they may be in a *Has-a* relationship;

For example, in Figure 1, the element 'professor' has a hypernym of 'faculty member', while 'department' has an meronym of 'academic institution', which is a sense of 'college'.

Instances (i.e., data) give entity-level clues about the relationship between schema elements. Hence instance-level evidence usually precisely characterizes the content of schema elements. By studying the similarity of the data instances of schema elements, *SeMap* discovers relationships that are difficult to identify on the schema-level:

- If two elements have a similar data distribution, they are likely to be *Equivalent*;
- If the instance of one element x subsumes that of another element y, it is likely that x *Has-a* y as its member;
- If the instances of x and y intersect, it is possible that x *Associates* with y.

A domain-specific ontology provides alternative representations of concepts in the domain, and their possible relationships. If available, *SeMap* will use this great source of relationship information. However, if a domain-specific ontology is not available, *SeMap* can function without it. To combine the possibly conflicting results suggested by various semantic evidence, *SeMap* associates each kind of evidence with a weight, and a unified conclusion is obtained by linearly combining the results. Specifically *SeMap* considers ontology > instance > sense > schema information.

For example, in Figure 1, for the match of 'professor' and 'faculty', the sense evidence suggests *Is-a* relationship, the type evidence indicates *Equivalent* or *Is-a* equally, and the label and structure evidence provide no advice. Assuming the four types of evidence have weights 0.4, 0.3, 0.2 and 0.1 respectively, the voting result correctly identifies the match 'professor':'faculty' as an *Is-a* relationship.

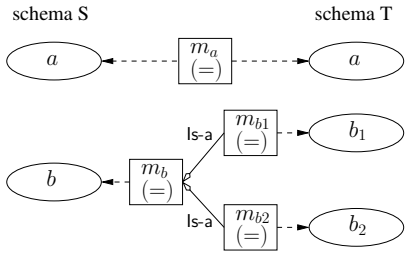

Figure 5: Mapping assembling for different match types Each 1-to-1 equivalence match corresponds to one mapping element, while each element of a complex match is associated with one mapping element.

4.3.3 Mapping Assembling

Finally, given the semantic relationships in Section 4.3.2, *SeMap* must create the final mapping. In order to ensure that the final mapping meets the specifications in the problem definition (Section 3), we apply the following rules to create mapping elements: (1) one mapping element is sufficient to represent a 1-to-1 equivalent match in the mapping and (2) one mapping element is created for each element of the match in a 1-to-1 non-equivalent match or complex match. Figure 5 illustrates both cases. While it is possible to identify the relationships between the mapping elements belonging to different matches in order to form a more complete mapping, it is beyond the scope of this paper.

5. EXPERIMENTAL ANALYSIS

To evaluate the effectiveness of our schema matching and mapping construction system, we applied *SeMap* to several real-world domains. The experiments were performed with two main goals:

- To evaluate *SeMap*'s matching accuracy, including both the correspondences measured in most previous schema matching systems, and the detected relationship types.
- To measure the relative contribution of different system components to the result. Specifically, we measured the performance gain from (1) different base matchers (2) the match selector, including user interaction.

Section 5.1 describes the experimental settings, including (1) the dataset used in experiments; (2) the expert mappings; (3) the metrics to evaluate matching results and (4) the experimental methodology. Section 5.2.1 presents the evaluation of matching accuracy. Section 5.2.2 shows the contribution of each system component to the final results.

5.1 Experimental Setting

5.1.1 Data Set

We evaluated *SeMap* on both synthetic and real datasets. The synthetic dataset is a version of the example shown in Figure 1 that has been expanded to cover more concepts. The real-life datasets are from two domains: Real Estate (i.e., houses for sale) and Course Information (i.e., courses at different universities). Each of the domains contains two data sets, which we refer to as "Real Estate 1" and "Real Estate 2" and "Course Info 1" and "Course Info 2". All real datasets are imported from the Illinois Semantic Integration Archive [1]. The real-life datasets are associated with data instances, which we exploited in addition to schema-level information. For the synthetic dataset we evaluated our approach on schema-level information only.

domain	schema	# leaf/ non-leaf	# rels.	depth
Real Estate	Homeseekers	25/3	27	3
	Texas	31/3	33	3
	Yahoo	23/2	24	3
Course Info	Reed	12/3	14	3
	Rice	11/4	14	4
	UWM	15/4	18	4
Synthetic	course	5 /1	5	2
	class	5/1	5	2

Table 1: Characteristics of the input schemas.

domain	schema S/T	# relationship			
		EQ.	*Is-a*	*Has-a*	*total*
Real Estate	Homeseekers/ Texas	18	6	12	36
	Homeseekers/ Yahoo	20	0	11	31
Course Info	Reed/U of WA	11	0	7	18
	Reed/WSU	18	1	8	27
Synthetic	course/class	2	4	3	9

Table 2: Characteristics of the expert mappings.

Since the real-life schemas are XML DTDs, we converted them into our model representation, leaving the structure and terms the same. The only change was that since each edge in the model representation has a relationship type, but there were no typed edges in the DTDs, we set all the DTD relationships as *Has-a* by default. The synthetic dataset was natively in the model format, and thus validates how the system would perform without this preprocessing.

5.1.2 Expert Mappings

In preparing the real-life datasets, we picked three schemas from each which had complex structure, and among which complex relationships can exist. The characteristics of these schemas — after the preprocessing described above to specify the relationship types within the input schemas — are shown in Table 1, including the number of elements (leaf and non-leaf), the number of relationships, and the maximum depth of the tree.

For each pair of schemas in the same domain, we created an expert mapping to judge *SeMap*'s results against. Several domains (Course Info 1 and Real Estate 2) included expert mappings, though without the semantically richer correspondence types. To avoid bias, we based our expert mappings on these existing expert mappings. The expert mappings' characteristics are shown in Table 2, including the total number of matches, the number of matches of each specific relationship type, and the percentage of elements involved in the mapping for both source and target schemas. The expert mappings contained no *Associates* relationships.

5.1.3 Evaluation Metrics

As in most previous schema matching work, we evaluated our approach via three metrics: recall, precision, and F-measure [22]. Precision P ($P = \frac{\#detected}{\#result}$) represents the

percentage of correctly identified matches over all matches identified by the system; Recall R ($R = \frac{\#detected}{\#mapping}$) is the percentage of correctly identified matches over all the matches in the given expert mapping. Recall and precision are inversely related, hence it is desirable to have one measurement to incorporate both recall and precision. F-measure F ($F = \frac{2PR}{P+R}$) equally weighs recall R and precision P.

As discussed above, *SeMap* finds both correspondences and implicit relationships. A correct match thus means (1) the correspondence is correct, and (2) the relationship is the same as in the expert mapping. So we measured the matching accuracy in two ways: (1) the total number of correct correspondences detected; (2) the number of correct matches for each type of relationships.

5.1.4 Experimental Methodology

For each domain, we performed three sets of experiments. We first evaluated *SeMap*'s matching accuracy. We then evaluated the relative contribution of each component and user interaction to the final mapping. This allowed us to tune the weights of *SeMap*'s label, type, and sense matchers. We found that the system was moderately sensitive to these parameters (Section 4.1.3), and list automatically tuning these parameters as future work. The specific settings for different datasets are as follows: (1) For the Real Estate dataset, the parameters are set as $\lambda_l = 0.3$, $\lambda_t = 0.5$, and $\lambda_s = 0.2$, reflecting that in the Real Estate dataset, the name, type and sense are all important to identifying the implicit relationship, however due to a number of abbreviations, e.g., 'ac' for 'air conditioner', the type information is especially important to finding the hidden correspondences; (2) For the Course Info dataset, $\lambda_l = 0.4$, $\lambda_t = 0.4$, and $\lambda_s = 0.2$ — the name and type convey equally significant semantic information, and the sense carries less importance weight due to a small quantity of synonyms, hyponyms, etc. in this dataset; (3) For the synthetic dataset, $\lambda_l = 0.4$, $\lambda_t = 0.3$, and $\lambda_s = 0.3$.

5.2 Experimental Results

We now present *SeMap*'s performance on synthetic and real datasets. We first consider *SeMap* without any user interaction. We show the overall matching accuracy and then analyze the contribution from different types of semantic evidence. Finally we measure the performance gain from incorporating user interaction into the match selector phase.

5.2.1 Matching Accuracy

Figure 6 shows *SeMap*'s accuracy. As discussed above, we measured both the accuracy of identified matches and identified relationships, and did not use any user interaction to improve results. The five bars (from left to right) show the matching accuracy for the three relationship types *Equivalent*, *Has-a*, and *Is-a* (correct match and correct relationship), accuracy across all relationship types (correct match and correct relationship), and total percent of correct matches (ignoring the relationship type) respectively.

The results show that *SeMap* achieved a high average matching accuracy not only in detecting the correct correspondences (total number of correct matches), but also in detecting the implicit relationships. Taking F-measure as an example, the percentage of correct correspondences ranges from 69% to 100%. Averaging across all experiments, the accuracy of detected relationships is 79%, 69% and 64% for

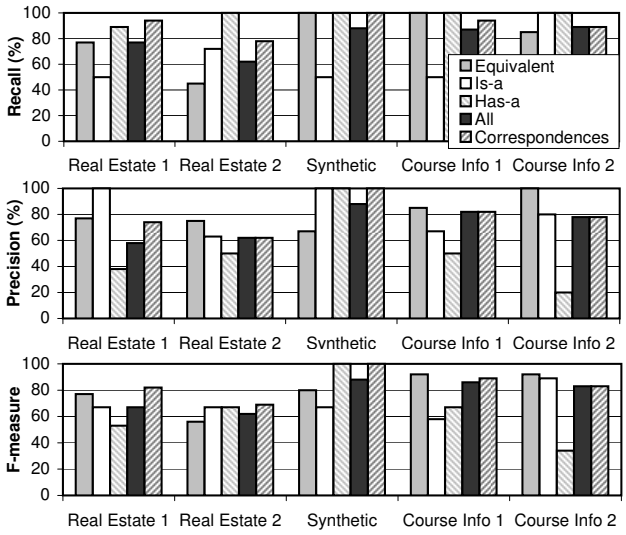

Figure 6: The recall, precision and F-measure for *Equivalent*, *Has-a*, *Is-a* relationships, all relationships, and correspondences for *SeMap*'s matching accuracy.

Equivalent, *Has-a*, and *Is-a* respectively, and 70% overall. The accuracy of correspondences detected is comparable to the results claimed in iMAP [5] (On Real Estate: 77-100% for 1-to-1 matches and 50-86% for 1-to-n matches), and that produced by [9] (on Real Estate and Course Info: 73% recall and 67% precision without a domain ontology, and 94% recall and 90% precision with a domain ontology — *SeMap* did not use a domain ontology, so these results are comparable). Thus, the schema match compares well to other results, even considering that *SeMap* only states a result is correct if the relationship type is correct as well, and hence is solving a more difficult problem.

Looking at these results in further detail shows that the matching accuracy highly depends on the domain. For example, for the two Real Estate datasets, *SeMap* has roughly the same overall accuracy as is the Course Info datasets. However for specific relationship types, some differences exist: for Real Estate, *SeMap* performs better in identifying *Equivalent* relationships in Real Estate 1, but worse for *Has-a* relationships. While for Course Info datasets, *SeMap* has higher accuracy in finding *Has-a* relationships in Course Info 1, but lower accuracy for *Is-a* relationships. This is caused by the different characteristics of the schemas. For example, the two schemas of Real Estate 1 have a more similar representation of the same concepts than that of Real Estate 2, thus it is easier for *SeMap* to detect *Equivalent* relationships in the first dataset. However, *SeMap* has quite low precision (about 40% for Real Estate 1 and 20% for Course Info 2) in identifying *Has-a* relationships.

Figure 7 provides a more detailed analysis of the composition of the matches identified by *SeMap*, i.e., of the matches identified as each relationship type, how many are (1) correct correspondence and correct relationship (2) correct correspondence but incorrect relationship (3) incorrect correspondence (non-match). In both Real Estate 1 and Course Info 2, incorrect correspondences are the main cause of the low precision in identifying *Has-a* relationship matches. The accuracy of *SeMap* after barring incorrect correspondences

schemas	relship.	composition			
		EQ.	*Is-a*	*Has-a*	*non-match*
Real Estate 1	*Equivalent*	13	0	0	4
	Is-a	0	2	0	0
	Has-a	4	2	8	7
Real Estate 2	*Equivalent*	9	0	0	3
	Is-a	0	5	0	3
	Has-a	3	0	7	4
Synthetic	*Equivalent*	2	1	0	0
	Is-a	0	1	0	0
	Has-a	0	0	4	0
Course Info 1	*Equivalent*	11	1	0	1
	Is-a	0	2	0	0
	Has-a	0	0	0	0
Course Info 2	*Equivalent*	12	0	0	0
	Is-a	0	4	0	1
	Has-a	0	0	1	4

Figure 7: Error analysis of the resulting mappings.

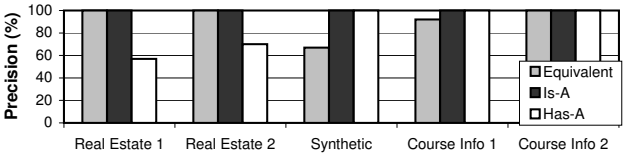

Figure 8: Precision for *Equivalent*, *Has-a* and *Is-a* relationships after pruning incorrect matches.

is shown in Figure 8, where the precision reaches near 100% in most datasets. Since the synthetic dataset is very small, *SeMap*'s low precision in this case corresponds to only one misclassified match. With more powerful schema matching techniques or domain knowledge, the number of incorrect matches can be significantly reduced, but that is outside the scope of this paper.

On average *SeMap* has higher accuracy in identifying *Is-a* relationships than *Has-a* relationships. This is because the thesaurus employed in *SeMap* (WordNet) returns comprehensive information of meronym/holonym relationships between two words. The accuracy of identifying *Has-a* relationships can improved by (1) pruning senses in using the thesaurus and (2) lowering the weight of the sense matcher; however this may affect finding other types of relationships. Both of these are future work, particularly understanding the interaction of the parameters.

5.2.2 Component Contribution

Next we studied the relative contribution of different types of semantic evidence. Specifically, we tested element label (name), element type and element sense, which are available in most schemas. In each test, we left out one type of semantic evidence and used the remaining two.

The two plots of Figure 9 show the F-measure of identified matches (correspondences) and identified relationships respectively. In almost all cases, each type of semantic evidence contributes to the overall performance. The exception is Real Estate 2, where the F-measure of relationships identified by a complete system is worse than that by a system without type evidence. This is because semantic evidence can conflict in determining the implicit relationships, again pointing to needing careful parameter tuning.

5.2.3 Incorporating User Feedback

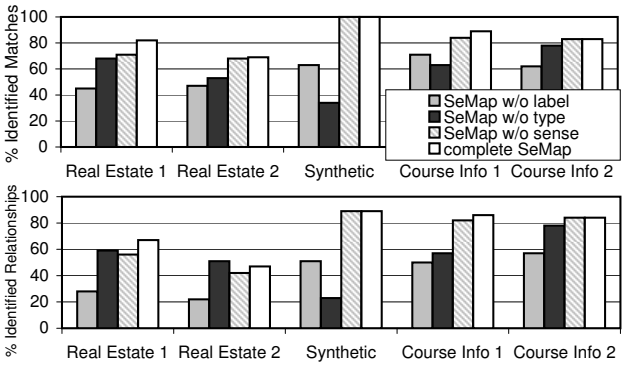

Figure 9: Relative contribution of different types of semantic evidence to the matching results of *SeMap*. The two plots (from up to down) show the F-measure of identified matches (correspondences) and identified relationships respectively.

We also studied the performance gain by incorporating user feedback (Section 4.2.3). Each candidate match is associated with an uncertainty estimate, based on its similarity score and domain constraints. At each iteration, the schema element whose candidate matches have the largest mutual entropy with other elements is selected, and the user is asked to provide the correct match assignment for this element. The procedure repeats until a threshold is reached.

We measured the number of correct matches required to be provided by the user before a perfect set of matches is reached. Note that we only tested on the accuracy of identifying the correspondences, but not extracting the implicit relationships. The F-measure of correct correspondences versus the amount of user interaction needed (percentage of expert matches provided over the total number of matches) is shown in Figure 10 (the synthetic dataset is skipped, since 100% correct correspondences is reached without any user interaction). In Real Estate 1, Real Estate 2 and Course Info 2, F-measure reaches its maximum value when about 20% of expert matches is provided. Over Course Info 1, about 10% of expert matches lead to a perfect set of matches. This result suggests that *SeMap* can effectively incorporate user interaction; it needs only a few equality constraints provided by users to achieve high-accuracy matches. Since F-measure incorporates both precision and recall, it is not always possible to achieve 100% recall, due to the limit of the current matching techniques that *SeMap* takes as input.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach of identifying generic, semantic relationships between the elements of two models (e.g., database schemas, ontologies, web interfaces) based on initial match information provided by current schema matching techniques. Our main contributions include (1) we point out the importance of the problem of identifying generic semantic relationships between schema elements; (2) we designed an architecture for semi-automatically constructing generic semantic mappings based on initial correspondence information; (3) we created a novel probabilistic framework that transforms match selection to a well-defined mathematical optimization problem; (4) we effectively modeled user interaction to minimize user effort by detecting

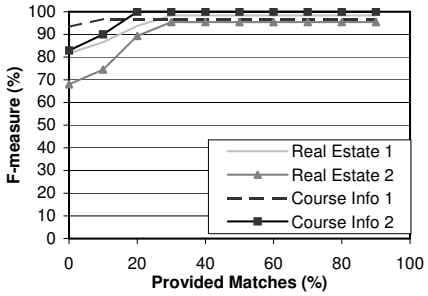

Figure 10: F-measure of correct correspondences vs. percentage of expert matches provided over the total number of matches.

critical points where feedback is maximally useful; (5) we proposed an effective solution to extracting relationship implicit in matches based on various types of semantic evidence; (6) we implemented a prototype system embodying the innovations above and a set of experiments to illustrate the effectiveness of our approach.

We envision several future directions. The first is to incorporate our system into a model management system [2], and explore the new possibility in meta-data management brought by generic, semantically rich mappings. Second, we would like to enhance our current prototype by adding in more matching techniques, and considering more types of semantic evidence. Third, more domain constraints (e.g., frequency, contiguity, nesting [7]) should be added to enhance *SeMap*'s match selector. Another direction would be to decouple the schema matching from the discovery of generic relationships — this would allow base matchers to be added without changing the heuristics, but would require modifying the semantic evidence used to identify the relationships. Finally, in judging which relationship is best represented by the input correspondences, *SeMap* takes into account additional information (e.g., lineage information) that the matchers themselves do not consider. As a result of this, if *SeMap* is unable to suggest an appropriate relationship, it may indicate that the input correspondence is wrong. One future direction is to redirect such concerns to improve the quality of the input match.

7. REFERENCES

[1] Illinois semantic integration archive. *http://pages.cs.wisc.edu/~anhai/wisc-si-archive/*, 2007.

[2] P. Bernstein. Applying model management to classical meta data problems. In *CIDR*, 2003.

[3] P. Bohannon, W. Fan, M. Flaster, and P. Narayan. Information preserving xml schema embedding. In *VLDB*, 2005.

[4] H. Chalupsky. Ontomorph: A translation system for symbolic knowledge. In *KR*, 2000.

[5] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: discovering complex semantic matches between database schemas. In *SIGMOD*, 2004.

[6] H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, 2002.

[7] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *SIGMOD*, 2001.

[8] A. Doan and A. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005.

[9] D. Embley, L. Xu, and Y. Ding. Automatic direct and indirect schema mapping: experiences and lessons learned. *SIGMOD Record*, 33(4):14–19, 2004.

[10] F. Giunchiglia and M. Yatskevich. Semantic matching. *Knowledge Engineering Review*, 18(3):265–280, 2004.

[11] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics (SSC)*, pages 100–108, 1968.

[12] F. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

[13] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *Knowledge Engineering Review*, 18(1):1–31, 2003.

[14] J. Kang and J. Naughton. On schema matching with opaque column names and data values. In *SIGMOD*, 2003.

[15] J. Madhavan. Learning mappings between models of data. *http://citeseer.ist.psu.edu/636517.html*, 1999.

[16] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, 2001.

[17] A. Maedche, B. Motik, N. Silva, and R. Volz. Mafra - a mapping framework for distributed ontologies. In *Knowledge Engineering and Knowledge Management Ontologies and the Semantic Web*, 2002.

[18] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[19] N. Noy and M. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *AAAI*, 2000.

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

[21] X. Renguo, T. Dillon, J. Rahayu, E. Chang, and N. Gorla. An indexing structure for aggregation relationship in oodb. In *DEXA*, 2000.

[22] C. Van Rijsbergen. *Information Retrieval, 2nd edition*. 1979.

[23] N. Rizopoulos. Automatic discovery of semantic relationships between schema elements. In *International Conference on Enterprise Information Systems*, 2004.

[24] M. Sayyadian, Y. Lee, A. Doan, and A. Rosenthal. Tuning schema matching software using synthetic scenarios. In *VLDB*, 2005.

[25] G. Stumme and A. Maedche. FCA-MERGE: Bottom-up merging of ontologies. In *IJCAI*.

[26] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *SIGMOD*, 2004.

[27] L. Yan, R. Miller, L. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. In *SIGMOD*, 2001.