1
Introduction

1.1.
Motivation and problem definition

A *database conceptual schema*, or simply a *schema*, is a high level description of how database concepts are organized. For the sake of our initial discussion, it suffices to assume that database concepts are organized as classes of objects and their properties.

We may decompose the database schema matching problem into the problems of *vocabulary matching* and *concept mapping*. As in (Rahm and Bernstein 2001), a *vocabulary matching* between schemas *S* and *T* is a relationship, that we call *Mapping*, between concepts of *S* and concepts of *T* in such a way that, if two concepts *s* in *S* and *t* in *T* are related by *Mapping*, then *s* and *t*, in some sense, have related meanings. We note that one or more *source concepts* may be related to one or more *target concepts*. A *concept mapping from S into T* is a set of transformation rules that express concepts of *S* in terms of concepts of *T* such that it is possible to translate queries over *S* into queries over *T*.

Schema matching is a fundamental issue in many database applications, such as query mediation, database integration, catalog matching and data warehousing (Casanova et al. 2007). Database integration in the context of the Web is even more complex, because: (1) the number of data sources may be enormous; Madhavan et al. (2007) estimates that there can be 25 million Deep Web sources; (2) the data model used in the data sources may be very different from each other and the mediator does not have much control over the sources; (3) the sources (sellers and makers) may join or leave the mediated environment or change their schemas at will. Therefore, such applications require that the schemas of the data sources be dynamically mapped to the schema of the mediator. An example of a database integration application comes from an e-commerce scenario. A user interested in buying a new car informs the desired

models, makers, year and price interval, and the system returns a list of selling offers ordered by average price and model. The technical characteristics and available selling offers are grouped by car models. At the back end of this application, a mediator receives the information of the desired car models, queries different data sources of car sellers and makers, identifies the equivalent car models across all data sources, attaches all selling offers and technical characteristics and returns the complete set of information to the user.

Another example of schema matching occurs in a business scenario. Suppose that a company has just bought another company. The buying company would likely intend to integrate its databases with the databases of the other company. Since the databases were independently developed, they often have many differences in terminology and structure, what leads to a non trivial integration process. Traditionally, this task starts with the IT team from both companies manually pointing out semantic correspondences between concepts of the schemas, based on the documentation and their knowledge about the databases. Clearly, it is a laborious task, which consumes a lot of IT resources. Automatic or semi-automatic tools able to identify such correspondences, or at least indicate some of the semantic correspondences, would definitely boost up the process.

We introduce a variant of the general database schema matching problem, that we call the *catalogue matching problem*. A *catalogue* is a simple database that holds information about a set of objects, typically classified using terms taken from a given thesaurus. Catalogues are fairly common and can be found, for example, in e-commerce and GIS applications, such as on-line stores and gazetteers. The schema of a catalogue has a single class with a list of properties. The catalogue matching problem involves three subproblems: (1) to match the catalogue conceptual schemas; (2) to find a relationship between the terms of the catalogue thesauri; (3) to define a way of identifying when two objects from different catalogues represent the same real-world object. Note that the last two problems are usually not considered in database schema matching.

The work presented in this thesis was previously described in other publications by the author. Chapter 3 presents a catalogue matching approach, originally published in (Leme et al. 2008a) and (Leme et al. 2009a). Chapter 4 describes a matching approach for complex schemas originally published in

(Leme et al. 2009b). The research reported in these publications benefit from the study described in the technical report (Leme et al. 2008b), where different similarity models were evaluated for schema matching purposes.

1.2. Related work

Rahm and Bernstein (2001) present an early survey of schema matching techniques. Euzenat and Shvaiko (2007) contain an account of ontology matching techniques. Bernstein and Melnik (2007) list the requirements for model management systems that support schema mappings, to which the work reported in this thesis contributes.

According to these surveys, matching techniques can be classified as *schema-based*, *extensional* (or *instance-based*) and *hybrid* (Rahm and Bernstein 2001). In schema-based techniques, the evidences for generating mapping elements are extracted from the schema definitions, such as names, descriptions, datatypes, relationships and constraints, e.g. a property *Client.name* from a *source* schema may be equivalent to a property *Customer.fullName* from a *target* schema, because their names ("*name*" and "*fullName*") are syntactically similar.

In extensional (or instance-based) techniques, the evidences come from the data held by the concepts of the schemas. For example, if the properties *Class1.property1* and *Class2.property2* from a *source* and *target* databases respectively assume the values shown in Figure 1, then, although the property names are syntactically unrelated, the two properties seem to be equivalent because they have similar sets of values. Additionally, if there is previous knowledge that the property *Book.author* may assume values from the set in Figure 1, then it can be inferred that the previous two properties may be equivalent to *Book.author* because the sets of values are similar.

Class1.property1, Class2.property2 $\in$ *{"Guimarães Rosa", "Machado de Assis", "William Shakespeare"}*

Book.author $\in$ *{"Edgar Allan Poe", "Guimarães Rosa", "Machado de Assis", "William Shakespeare"}*

Figure 1. Sets of values assumed by the properties Class1.property1, Class2.property2, Book.author.

Finally, hybrid techniques combine evidences from schemas and instances to produce mappings.

Melnik et al. (2002) propose a schema-based technique for schema matching. The source and target schemas are modeled as a graph in such a way that each node represents a possible mapping of two schema concepts. The schema matching is the set of mappings with the most similar concepts. The similarity of each node depends on the characteristics of its concepts and the similarity of its neighbors. An iterative process propagates the similarities of the nodes to their neighbors until all similarities converge to a stable value.

Madhavan et al. (2001) model the source and target schemas as two graphs where each node represents a schema concept. The schema matching consists of the most similar pair of nodes. The similarity of each pair depends on the syntactic similarity of the names of the concepts and the structure of the subgraph below them.

Do and Rahm (2002) propose the COMA schema matching system as a platform to combine multiple matchers in a flexible way. It provides a large spectrum of individual matchers, in particular, a novel approach aiming at reusing results from previous match operations, and several mechanisms to combine the results of multiple matchings.

Doan et al. (2001) propose a system (LSD – Learning Source Description system) that employs and extends current machine-learning techniques to semi-automatically find mappings. It first asks the user to provide the semantic mappings for a small set of data sources and then uses these mappings, together with the sources, to train a set of learners. Each learner exploits a different type of information, either from the source schema or from their data. Once the learners have been trained, LSD first finds semantic mappings for a new data source by applying the learners and then combines their predictions using a meta-learner.

Wang et al. (2004) describe a technique based on query probing to match Web databases, which relies on human intervention to select a set of typical instances used in the probing. Brauner et al. (2007b) apply this idea to match geographical database Web services.

Brauner et al. (2008) describe a matching algorithm based on measuring the similarity between the property domains of distinct Web databases.

Madhavan et al. (2005) propose the use of a set of schemas and mappings to

help the schema matching algorithms. The authors use predictor algorithms that measure the similarity between schema elements, adopted in the PayGo architecture (Madhavan et al. 2007).

Bilke and Naumann (2005) propose a property matching technique based on duplicate instances. Brauner et al. (2007a) adopt the same idea to match two thesauri. The method first finds the K most similar pairs of tuples and then compares the values of pairs of properties. The matching candidates are chosen as those pairs of properties with more values in common. The process of finding duplicates considers that each tuple is represented as a single string. Two tuples are considered equivalent if their string representations are similar. However, if the number of properties in each database is drastically different, two equivalent tuples may have very different string representations, thereby resulting in a poorer matching.

Finally, Udrea et al. (2007) present the algorithm ILIADS (Integrated Learning In Alignment of Data and Schema) for ontology alignment. The method combines similarity clustering and incremental logical inference. For similarity calculation the algorithm takes into account lexical, schema structure and data information. In the clustering of ontology entities (classes, properties and instances), new axioms and logical consequences are created. For example, while merging two medical ontologies A and B, if the algorithm adds the axiom

`(A:E-Coli-Poisoning, owl:sameAs, B:E-Coli)`,

to the set of existing axioms

`(A:discoveredBy, owl:inverseOf, B:discoverer)`,
`(A:discoveredBy, owl:Type, owl:FunctionalProperty)`,
`(B:T.S.Esherich, B:discover, B:E-Coli)` and
`(A:E-Coli-Poisoning,A:discoveredBy, A:TheodorEscherich)`

It will produce the following implication:

`(A:TheodorEscherich, owl:sameAs, B:T.S. Esherich)`.

The existing schema-based techniques can match schemas with multiple classes and relationships. Indeed, they compare the interconnections between the

schema elements in different schemas for extracting evidences of matching candidates. Such techniques rest mainly on the principle of comparing the syntactic similarity of names and descriptions of the elements of the different schemas.

On the other hand, extensional techniques are grounded on the interpretation, traditionally accepted, that “terms have the same extension when true of the same things” (Quine 1968)’However, they can not handle complex schemas, i.e., schemas with multiple classes and relationships.

Moreover, the database integration problem requires that a single query be translated to different databases. Current techniques focus on the task of finding matching schema elements and do not provide means of effectively integrating data from multiple data sources.

In this thesis we advocate that extensional techniques are more robust than purely syntactical approaches because they get closer to the real schema semantics. We also point out that query translation is an important issue whose solution is intimately linked to the schema matching solution. We propose a schema matching approach mainly based on extensional techniques, and a method of generating query translation rules which uses the schema matching results.

1.3. Thesis contributions

In this thesis, we propose hybrid matching techniques based on instance values and on schema information, such as datatypes, cardinality and relationships. The techniques uniformly apply similarity functions to generate matchings and are grounded on the interpretation, traditionally accepted, that “terms have the same extension when true of the same things” (Quine 1968). In our context, two concepts match if they denote similar sets of objects. The techniques essentially differ on the nature of the sets to be compared and on the similarity functions adopted. For example and in a very intuitive way, two classes match if their sets of observed instances are similar, two terms from different thesauri match if the sets of instances they classify are similar, properties match if their sets of observed values are similar.

We address in Chapter 3 the catalogue schema matching problem. We

introduce a matching approach, based on the notion of similarity, which applies to pairs of thesauri and to pairs of lists of properties. We then describe matchings based on the co-occurrence of data and introduce variations that explore certain heuristics. Lastly, we discuss experimental results that evaluate the precision of the matchings introduced and that measure the influence of the heuristics.

In Chapter 4, we focus on the more complex problem of matching two schemas that belong to an expressive OWL dialect. We decompose the problem of OWL schema matching into the problem of vocabulary matching and the problem of concept mapping. We also introduce sufficient conditions guaranteeing that a vocabulary matching induces a correct concept mapping. Next, we describe an OWL schema matching technique based on the notion of similarity. We develop a similarity function based on the contrast model (Tversky and Gati 1978), which proved (Leme et al. 2008b) to efficiently capture the notion of similarity, and describe heuristics that lead to practical OWL matchings. Lastly, we describe experimental results that evaluate the precision of the OWL matchings introduced, measure the influence of the heuristics and compare the customized contrast model with three different similarity functions.

Unlike any of the instance-based techniques previously defined (see Section 1.2), the OWL schema matching process we describe uses similarity functions to induce vocabulary matchings in a non-trivial way. We also point out that the structure of OWL schemas may lead to incorrect concept mappings and indicate how to avoid such pitfalls.

The assumptions that the database schemas to be matched are described in OWL and that the data obtained from the databases is available as sets of RDF triples facilitate the construction of matching techniques, since schema elements and data instances are similarly defined (as RDF triples). However, the techniques introduced in the thesis can be directly applied to conceptual schemas described in other database models, such as the relational model. In conjunction, these assumptions permit us to concentrate on a strategy to unveil the semantics of the database schemas to be matched, without being distracted by syntactical peculiarities.

Contrasting with (Doan et al. 2001, Madhavan et al. 2005), we do not use machine learning techniques to acquire knowledge about matchings. Instead, we capture semantic similarity by adopting similarity functions and heuristics that

depend on the schema concepts. We consider this strategy to be more general because it can identify matching candidates which were not in the training corpus. For example, in (Doan et al. 2001) the system learns how to match a target schema with a reference schema, or a mediated schema, based on user decisions while manual matching a set of target schemas with the reference schema. Using this approach, the system can not identify matchings for schema elements which did not belong to the reference schema. However, just like machine learning techniques, we also depend on training corpora to calibrate thresholds in order to decide the matchings, once the similarity scores have been computed.

Contrasting with (Brauner et al. 2007b, Wang et al. 2004), which measure the similarity between concepts only by the commonalities between sets of values, we use similarity functions which take into account not only the commonalities, but also the differences between concepts. Such models proved to increase the precision of the matching process. In addition, we use similarity heuristics that operate at the level of data values, that is, they permit comparing data values based on their similarity, and not just on their exact equality.

We overcome the limitation of representing an instance by a string constructed out of all its property values, introduced in (Bilke and Naumann 2005) and by representing an instance by a string constructed out of the values only of those properties that match, in a first approximation. In fact, we adopt a three-step process: we first find a preliminary property matching; then, we find instances that match based on this preliminary property matching; finally, we use instance matchings to refine the preliminary property matchings.

In summary, unlike the techniques listed in Section 1.2, we propose hybrid matching techniques that are uniformly grounded on similarity functions to generate matchings between simple catalogue schemas and between more complex OWL schemas.

Finally, we introduce the idea of decomposing the problem of schema matching into the problems of vocabulary matching and concept mapping, which are often confused in the literature. We also show when a vocabulary matching induces a concept mapping which is correct with respect to the integrity constraints of the schema, which is also frequently overlooked in the literature (Leme et al. 2009b).

1.4. Thesis outline

This thesis is organized as follows. Chapter 2 provides background information on the RDF/OWL languages and on similarity functions. It also introduces a new similarity function based on the contrast model proposed by Tversky and Gati (1978).

Chapters 3 and 4, the core of the thesis, gradually introduce the matching techniques. Chapter 3 addresses the catalogue schema matching problem, which is a simpler, albeit recurrent problem in e-business. Chapter 4 enhances the matching technique in such a way that it can be applied to more complex schemas, with multiple classes, class hierarchies and relationships (object properties). Both chapters present experimental results measuring the performance of the matching techniques.

Finally, Chapter 5 discusses several directions for future research based on the work presented in this thesis and summarizes the main contributions.