Towards a Multi-Layer Architecture for Combination of Schema Matchers

Diego Ernesto R. Pessoa, Ana Carolina Salgado and Bernadette Farias Lóscio

Centro de Informática
Universidade Federal de Pernambuco (UFPE)
Recife, Brazil
{derp,acs,bfl}@cin.ufpe.br

1 Introduction

Schema matching is the process of establishing correspondences between elements of distinct schemas. These correspondences are generated by using a single similarity measure or by combining different ones [6]. Until now, even after more than 25 years of research, there is not a perfect schema matching solution [3]. Although current systems perform well for some knowledge domains, in some cases, they produce inconsistent or erroneous results. This makes the existing approaches more dependent on human interactions, whichâĂŃ can lead to imprecise results.

Schema matching is relevant for many modern applications. For instance, if one wants to get global statistics of product sales in different e-commerce stores, it would be necessary to integrate data from different data sources, including those that their schemas adopt distinct logical designs. Especially in the case of schema matching between heterogeneous and autonomous data sources on large-scale environments, the large number of correspondences that can be identified between the targeted schemas often do not represent a valuable outcome. Similarly, the wide number of existing matchers makes harder to choose the most suitable to use. So, our approach handles the identification of correspondences in scenarios with large number of schemas. We propose a flexible architecture capable of combining different matching approaches following application requirements.

An application requirement is a tuple $R = \langle D, S, M \rangle$, where D is a set of desirable features from a Data Source (e.g. response-time, language, freshness), S is a set of targeted elements from the Schemas (e.g. the *Customer* entity or the *Price* attribute) and M is a set of parameters to the matchers execution (e.g. only consider element-based matchers, maximum runtime of n seconds).

The intuition of our approach is that there is a need for a solution capable of stating which set of matchers fits some application requirements for a large number of schemas. In this context, we propose a multi-layer architecture for combination of schema matchers, which regards the main steps of traditional Schema Matching approaches [2]. This paper is organized as follows. In Section 2 we present our architecture and describe its respective components. In Section 3 we discuss the next steps of our ongoing research and future work.

2 A Multi-Layer Architecture for Combination of Schema Matchers

Following the principles claimed by Madhavan et al. [7], we consider to reduce uncertainty in all steps of the Schema Matching process. In this sense, we are proposing a general architectureâĂŃ composed of three layers.

Figure 1 presents a brief overview of the proposed multi-layer architecture for combination of Schema Matchers. This architecture consists of three layers: 1) Pre-Match Layer; 2) Matching Layer and 3) Post-Match Layer. In the Pre-Match Layer, we first collect the application requirements, then a set of suitable data sources is selected. The schemas of these selected data sources are submitted to a normalization step, standardizing their schema elements. Metadata from data sources (e.g. schemas, knowledge domain) are also collected to be used as background knowledge. Such information is the input of the Matching Layer, which can establish the most suitable matchers for each scenario. Finally, the Post-Match Layer contains components aiming to evaluate the obtained results (correspondences). This evaluation can be accomplished via manual user intervention or even applying Machine Learning techniques. We will describe them in details in what follows.

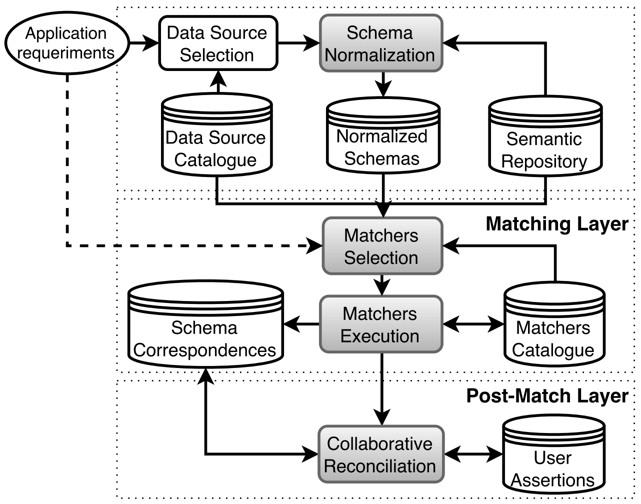

Fig. 1. Multi-Layer Architecture for Combination of Schema Matchers

2.1 Pre-Match Layer

The *Pre-Match Layer* aims to prepare the environment to the next steps of the process, collecting a set of useful information. To support the execution of schema matching tasks, we assume the existence of a *Data Source Catalogue*. This catalogue is used to manage a set of heterogeneous sources. The registry, standardization and data storage in an efficient way are some of the catalogue features. We clarify that the *Data Source Catalogue* is not in the scope of this work. The quantity of sources available from the *Data Source Catalogue* could lead to many comparisons. It becomes necessary to reduce this range. So, the Data Source Selection module identifies the most relevant sources to be in the pipeline, in accordance with the application requirements. Then, schema elements from selected sources are normalized by the *Schema Normalization* module according to terms from a Semantic Repository [1], which handles the combination of background knowledge from diverse semantic sources. The normalized schemas are stored for further use.

2.2 Matching Layer

The *Matching Layer* handles the selection, combination and execution of Schema Matchers. The outcome of this layer is a set of correspondences generated by a combination of matchers. These matchers are obtained from a *Matchers Catalogue*, which gather a set of existing matchers to be chosen. This task of choosing among many matchers is far from trivial [6]. Although several approaches combine different matchers, they still have some drawbacks. Some solutions adopt fixed heuristics (e.g. COMA++ [4]), which can do the selection of matchers unfeasible if the users do not have proper knowledge about the systems. On the other hand, some approaches adopt Machine Learning techniques. However, in large-scale environments it could be a hard task to set up a useful training set for supervised learning techniques (e.g. YAM [5]) or to express dependencies between user interactions over decision trees in active learning approaches (e.g. ALMa [8]), because it have poor statistical values (e.g. True, false). Looking for more flexibility, the *Matcher Selection* module will select different heuristics considering the application requirements, which will be combined and executed by the *Matchers Execution* module. Our intuition is that, considering large-scale environments, our approach can generate results that better fulfill user needs. To check this assumption, we are working on an experiment to compare our approach to other matchers combiners to integrate real-world applications.

2.3 Post-match Layer

The Post-Match Layer receives the generated set of correspondences to evaluate. This evaluation phase, also known as matching reconciliation, is supported by a human user. In our case, the application user must select the best correspondences among a set of top-k ranked items. This task is handled by the Collaborative Reconciliation module, which stores all decisions taken by application users in a repository of User Assertions. To this module, we are inspired in the method described by Zhang et al. [9], in which an approach based on crowdsourcing is established to accomplish the reconciliation task in a collaborative manner, reducing user participation in new tasks over the same sources.

3 Discussion and Future Work

In this short paper, we describe our ongoing research towards a multi-layer architecture to support the combination of schema matchers. We adopt flexible strategies that use application requirements to guide schema selection and matcher combination for a Schema Matching process. As future work, we plan to:

- detail and better formalize concepts used in the architecture, as for example the application requirements.
- generate a prototype to test combinations of candidate matchers and requirements from real-world applications, analyzing which solution is more suitable to each one.
- identify which rules are commonly used when the matcher combination is performed for a particular type of application.

The ultimate goal of this research is to build a system to execute Schema Matching tasks combining best matchers for a given application. The outcome of this matching process can serve in many tasks of data integration involving heterogeneous data sources, such as content delivery, database integration, query rewriting, duplicate data elimination and entity matching.

References

1. P. Arnold and E. Rahm. SemRep: A Repository for Semantic Mapping. *BTW*, pages 177–194, 2015.
2. Z. Bellahsene, A. Bonifati, and E. Rahm, editors. *Schema Matching and Mapping*. Data-Centric Systems and Applications. Springer, 2011.
3. P. a. Bernstein, J. Madhavan, and E. Rahm. Generic Schema Matching, Ten Years Later. *PVLDB*, 4(11):695–701, 2011.
4. H.-H. Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. *VLDB*, pages 610–621, 2002.
5. F. Duchateau, Z. Bellahsene, R. Coletta, R. J. Miller, Z. Bellahsene, and R. J. Miller. (Not) yet another matcher. In *Proceeding of the 18th ACM conference*, page 1537, New York, New York, USA, 2009. ACM Press.
6. A. Gal and T. Sagi. Tuning the ensemble selection process of schema matchers. *Information Systems*, 35(8):845–859, Dec. 2010.
7. J. Madhavan, S. R. Jeffery, S. Cohen, X. L. Dong, D. Ko, C. Yu, and A. Halevy. Web-scale Data Integration: You can only afford to Pay As You Go. pages 1–9, Dec. 2006.
8. D. Rodrigues, A. S. da Silva, R. Rodrigues, and E. dos Santos. Using active learning techniques for improving database schema matching methods. *IJCNN*, pages 1–8, 2015.
9. C. J. Zhang, L. Chen, H. V. Jagadish, and C. C. Cao. Reducing uncertainty of schema matching via crowdsourcing. *Proceedings of the VLDB Endowment*, 6(9):757–768, July 2013.