Large-Scale Schema Matching

Erhard Rahm, Eric Peukert

Synonym

Large-Scale Ontology Alignment

Definition

Schema matching aims at identifying semantically corresponding elements in two or more schemas, e.g., database schemas or ontologies. Large-scale schema matching focusses on the challenging cases of matching large schemas or more than two schemas. Matching multiple (> 2) schemas is also known as holistic schema matching.

Overview

Schema matching aims at identifying semantic correspondences between metadata structures or models, such as database schemas, XML message formats, and ontologies. Solving such match problems is a key task in numerous application fields, in particular for data exchange and virtually all kinds of data integration. For example, in the two simple database schemas DB1.Student (Name, Major, Marks) and DB2.Grad-Student (LastName, FirstName, Major, Grades) simple (equivalence) correspondences would be DB1.Student ≈ DB2.Grad-Student, DB1.Major ≈ DB2.Major, and DB1.Marks ≈ DB2.Grades while DB1.Name is related to both DB2.LastName and DB2.FirstName (part-of correspondences 'DB2.LastName - DB1.Name' and 'DB2.FirstName - DB1.Name'). The inherent semantic heterogeneity in different schemas makes is difficult for automatic approaches to correctly determine the correspondences so that there is a need for user involvement. The complexity is especially high for Big Data applications that frequently involve *large-scale schema matching* where large schemas with thousands of elements or many data sources with different schemas have to be dealt with. In this entry, we will focus on the pairwise matching of large schemas while matching of multiple schemas is treated under *Holistic schema matching*, sometimes also called *collective schema matching* (Rahm 2016).

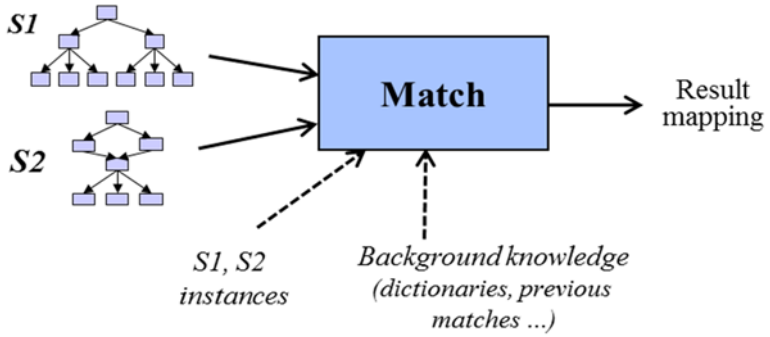

Fig. 1: Input and output of an automatic schema matching system

Most approaches for schema matching focus on 2-way or pairwise matching where two related input schemas are matched with each other. As shown in Fig. 1, automatic matching may also make use of instance data for the input schemas or background knowledge such as dictionaries or previous match results that simplify the identification of corresponding elements in the input schemas. The result of pairwise schema matching is usually an equivalence *mapping* containing the identified semantic correspondences, i.e. pairs of semantically equiv-

alent schema elements (e.g., schema attributes or ontology concepts). Some match approaches for ontologies also try to determine different kinds of semantic correspondences, such as is-a and part-of relationships between concepts (Euzenat and Shvaiko 2013, Arnold and Rahm 2014). Due to the typically high semantic heterogeneity of schemas, algorithms can determine approximate mappings only. The automatically determined mappings may thus require the inspection and adaptation by human domain experts (deletion of wrong correspondences, addition of missing correspondences) to obtain the correct mapping. Mapping results are useful input for further data integration tasks such as merging or integrating the respective schemas since a mapping indicates the elements that should be represented only once in the integrated result. In fact, several such mapping-based merge approaches have been proposed for both schemas and ontologies (Pottinger and Bernstein 2003, Raunich and Rahm 2014).
Schema matching has been a very active research area in the last decades, and numerous techniques and prototypes for automatic matching have been developed (Rahm and Bernstein 2001, Euzenat and Shvaiko 2013). Matches are usually identified by a combination of techniques (also called *matchers*) to determine the similarity of elements. These include

- linguistic matchers that focus on the similarity of element names based on string similarity measures or synonym information from background knowledge resources such as dictionaries,
- structural matchers that determine the similarity of elements based on the similarity of their schema neighbors, e.g., ancestors and/or descendants,
- instance matchers that determine the similarity of elements based on their associated instance data, e.g., attribute values or entities of a concept.

Despite the advances made, current match systems still struggle to deal with large-scale match tasks. In particular, achieving both good efficiency and good effectiveness are two major challenges. Further challenges for large-scale schema matching include benchmarking and evaluation as well as suitable approaches and interfaces for user involvement. The latter issues are further discussed in (Hung et al 2013) and are beyond the scope of this entry.

Key Research Findings

We focus on recent approaches to improve efficiency and effectiveness for large-scale pairwise schema (and ontology) matching.
The main performance (efficiency) problem for matching large schemas is the potentially huge search space if one has to compare every element of the first with every element of the second input schema. For schemas and ontologies with thousands or tens of thousands of elements this leads to a search space of many million comparisons. This cannot only result in long execution times for the evaluation of several matchers but will also make it difficult to find the relatively few correspondences under so many possible candidates. Hence, it is imperative for large-scale matching to reduce the search space. Furthermore, one can apply parallel matching to improve runtimes. Note that these techniques (reduction of the search space and parallel matching) are also common for large-scale entity matching (Kolb and Rahm 2013, Dong and Srivastava 2015).

Parallel matching is relatively straight-forward but has been applied in only few prototypes for schema matching, e.g., in Gomma (Groß et al 2010) and SPHeRe (Amin et al 2016). It can be used to speed-up single matchers, e.g. by partitioning the search space (set of element pairs from the input schemas) and evaluating different partitions in parallel on different processors. Furthermore, different matchers can be evaluated in parallel to each other.

Reducing the search space for matching large schemas has found more interest. One idea is to apply several matchers in sequence and to initially use fast matcher(s) (e.g., a string matcher on element names) with a low similarity threshold to already eliminate most element pairs before applying further matchers (Ehrig and Staab 2004, Peukert et al 2010). Several other approaches are based on a partitioning of the input schemas and restrict the search of similar schema elements to a subset of these partitions. Such *partition-based matching* can achieve a significant reduction of the search space and thus improved efficiency. Furthermore, matching the smaller partitions reduces the memory requirements compared to matching the full schemas. The partition-based match tasks can also be performed in parallel for additional performance improvements (Amin et al 2016).

There are many possible ways to perform partitioning and finding the most effective approaches is still an open research problem. COMA++ was one of the first systems to support partition-based schema matching by a so-called fragment matching (Do and Rahm 2007). Fragment matching works in two phases. In the first phase, the fragments of a specified type (e.g., subschemas such as relational tables) are determined and compared with each other to identify the most similar fragments from the other schema worth to be fully matched later. The search for similar fragments is some kind of light-weight matching, e.g., based on the similarity of the fragment roots. In the second phase, each pair of similar fragments is matched independently to identify correspondences between their elements.

The ontology matching system Falcon-AO (Hu et al 2008) uses a structural clustering to initially partition the ontologies into relatively small, disjoint blocks. Matching is then restricted to the most similar blocks from the two ontologies. To determine the block similarity, Falcon-AO utilizes so-called anchors, i.e., highly similar element pairs that are determined before partitioning by a combined name/comment matcher. Matching is then limited to pairs of blocks sharing at least one anchor. Further approaches for partition-based matching used different kinds of clustering for the input schemas/ontologies, e.g. hierarchical clustering in (Algergawy et al 2011).

The taxonomy matching system AnchorFlood implements a dynamic partition-based matching that avoids the a-priori partitioning of the ontologies (Hanif and Aono 2009). Like in Falcon-AO, the approach utilizes anchors (similar concept pairs) but takes them as a starting point to incrementally match elements in their structural neighborhood until no further matches are found or all elements are processed. Thus, the partitions (called segments) are located around the anchors and their size depends on the continued success of finding match partners for the considered elements.

Several approaches considered the case when one of the input schemas/ontologies is much smaller than the other one so that only the larger input is partitioned, e.g., (Zhong et al 2009, Gross et al 2012). In GOMMA (Gross et al 2012), a fast name matcher is initially applied between the small and larger ontology. Matching is then limited to the sub-ontologies from the larger ontology for which the number of correspondences exceeds a certain ratio of all correspondences.

Effectiveness (high match quality) is another key challenge that requires the correct and complete identification of semantic correspondences. This problem becomes more difficult for larger search spaces. Furthermore, the semantic heterogeneity is typically high for large-scale match tasks, e.g., the schemas may largely differ in their size and scope making it difficult to find all correspondences and to avoid false match candidates. In addition to the combined use

of several matchers the following techniques are especially promising to improve match effectiveness for large-scale schema matching (Rahm 2011):

- *Post-processing and repair of proposed correspondences*. The individual matchers typically generate a large number of possible correspondences from which the final ones need to be selected before they are included in the match result, e.g., for possible validation by a domain expert. The selection step should not only consider different ways to aggregate matcher results (union, intersection, weighted similarity values, etc.) and similarity thresholds but should also consider global constraints, e.g., if there should be at most 1 or a small number k of match partners in the other schema. So it is reasonable for 1:1 mappings to enforce so-called stable marriages or to only accept the mutually best match candidates, i.e., a correspondence c1-c1' is only accepted if c1' is the most similar element for c1 and vice versa (Max-Both selection in COMA).

  For semantically expressive ontologies, e.g. specified in the language OWL-DL, the determined mapping M between two ontologies may introduce inconsistencies so that axioms in the input ontologies may be violated if one combines O1 and O2 with the correspondences of M within a merged ontology. For example, for relations 'O1.Customer is-a O1.Person' and 'O2.Person is-a O2-Member' the equality correspondences 'O1.Person ≈ O2.Person' and 'O1.Customer ≈ O2.Member' would result in an inconsistency because 'O1.Person is-a O1.Customer' could then be derived. The task of *mapping repair* (Meilicke et al 2007) is to eliminate correspondences during post-processing that would cause such inconsistencies (e.g., 'O1.Customer ≈ O2.Member' in the example). Mapping repair is supported in some ontology matching systems, e.g., LogMap and AlComo. It has to be noted that mapping repair can reduce recall by eliminating correct correspondences. Furthermore, the input ontologies may follow partially conflicting way in the representation of a domain that cannot be resolved by automatic repair actions but only with manual interaction.

- *Utilization of background knowledge.* The information within the input schemas itself is insufficient for large-scale matching with heterogeneous inputs so that it becomes necessary to find correspondences with the help of general or domain-specific backgound knowledge in dictionaries and thesauri (e.g., Wordnet, UMLS etc.).

- *Reuse of previous match results.* Already matched schemas and the match results can be maintained in a repository for later reuse, in particular when modified versions of the schemas or other similar schemas have to be matched. Reusing such already found correspondences is a special form of utilizing background knowledge that holds high promise but also requires a complex infrastructure to find the reusable correspondences and schema mappings. The corpus-based match approach of (Madhavan et al 2005) uses a domain-specific corpus of schemas and match mappings and focuses on the reuse of element correspondences. They augment schema elements with matching elements from the corpus and assume that two schema elements match if they match with the same corpus element(s).
  COMA was the first match tool to reuse entire schema mappings and to support indirect matching by composing already existing mappings (Do and Rahm 2002). For example, a mapping between schemas S1 and S3 can be obtained by combining preexisting S1-S2 and S2-S3 mappings involving another schema S2. Such derived mappings can also be combined with the results of regular matchers. The reuse and composition approach of COMA also allows the adaptation of an old mapping S1-S2 after one of the schemas evolves, e.g., S2 to S2', by composing the previous match result with the

newly determined mapping S2-S2' between the old and new version of S2. The reuse and composition approach has also been incorporated and extended within the GOMMA prototype for ontology matching (Gross et al 2011).

- *Semi-automatic tuning of match strategies*. In most current systems the match strategy needs to be manually specified and configured, in particular the choice of matchers and the method to combine matcher results as well as critical parameters such as the weights of different matchers and similarity thresholds. Obviously, these decisions have a significant impact on match effectiveness but are difficult to make even for expert users. Semi-automatic tuning approaches aim at reducing this problem either by using learning-based or rule-based approaches. While supervised machine learning methods are very successful for semi-automatically configuring entity matching strategies, they are less applicable for learning schema matching configurations due to the difficulty to provide a sufficiently high number of correct schema mappings for training.

  Several match systems including RIMOM analyze the schemas to be matched and determine their linguistic and structural similarity. These similarity characteristics are then used to select matchers or to weight the influence of different matchers in the combination of matcher results. (Peukert et al 2012) proposes a rule-based approach to tune entire match workflows based on features of the input schemas and of intermediate matcher results. There are different kinds of rules representing expert knowledge, in particular to select the matchers based on schema features and to combine matcher results based on schema and mapping features.

Future Directions

Despite the huge amount of research on schema matching, current products for data exchange and data integration typically include only simple linguistic approaches for automatically finding correspondences but still rely to a large degree on manual match decisions (Rahm 2011). As a result, they are ill suited for large-scale match problems pointing to the need of bringing more proposed match approaches into products and practical application.

There is still a need for more research. In particular, it would be worthwhile to comparatively evaluate proposed approaches for mapping repair, reusing previous match results, and for self-tuning match strategies. Based on the observations we envision possibilities to improve previous approaches in these areas.

Further topics for future research include improved user interfaces for controlling schema matching processes and providing expert feedback as well as holistic schema matching for multiple (> 2) schemas.

References

- Amin MB, Khan WA, Hussain S, Bui DM, Banos O, Kang BH, Lee S (2016) Evaluating Large-Scale Biomedical Ontology Matching Over Parallel Platforms. IETE Techn. Review, 33(4), 415-427
- Arnold P, Rahm E (2014) Enriching Ontology Mappings with Semantic Relations. Data and Knowledge Engineering 93: 1–18
- Algergawy A, Massmann S, Rahm E (2011) A clustering-based approach for large-scale ontology matching. Proc. ADBIS conf., Springer LNCS 6909

- Do HH, Rahm E (2002) COMA – A System for Flexible Combination of Sche-ma Matching Approaches. Proc. VLDB conf., pp 610–621
- Do HH, Rahm E (2007) Matching large schemas: Approaches and evaluation. Inf. Syst. 32(6): 857-885
- Dong XL, Srivastava D (2015) Big Data Integration. Morgan & Claypool
- Ehrig M, Staab S (2004) Quick ontology matching. Proc. ICSW, Springer LNCS 3298
- Euzenat J, Shvaiko P (2013) Ontology Matching. 2nd edition, Springer
- Gross A, Hartung M, Kirsten T, Rahm E (2010) On Matching Large Life Science Ontologies in Parallel. Proc. DILS conf., Springer LNCS 6254
- Gross A, Hartung M, Kirsten T and Rahm E (2011) Mapping Composition for Matching Large Life Science Ontologies. Proc. ICBO, 109-116
- Gross A, Hartung M, Kirsten T, Rahm E (2012) GOMMA results for OAEI 2012. Proc. 7th Ontology Matching workshop. CEUR-WS 946
- Hanif MS, Aono M (2009) An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. J. Web Sem. 7(4): 344-356
- Hu W et al (2008) Matching large ontologies: A divide-and-conquer-approach. Data Knowl. Eng. 67(1): 140-160
- Hung N.Q.V., Tam N.T., Miklós Z., Aberer K. (2013) On Leveraging Crowdsourcing Techniques for Schema Matching Networks. In: Proc. DASFAA, LNCS 7826 Springer
- Kolb L, Rahm E (2013) Parallel entity resolution with Dedoop. Datenbankspektrum, Springer
- Madhavan J, Bernstein PA, Doan A, Halevy AY (2005) Corpus-based Schema Matching. Proc. IEEE ICDE conf.
- Meilicke C, Stuckenschmidt H, Tamilin A (2007) Repairing Ontology Mappings. AAAI, 1408-1413
- Peukert E, Berthold H, Rahm E (2010) Rewrite Techniques for Performance Optimization of Schema Matching Processes. Proc. EDBT conf.
- Peukert E, Eberius J, Rahm E (2012) A self-configuring schema matching system. Proc. IEEE ICDE conf., 306-317
- Pottinger RA, Bernstein PA (2003) Merging models based on given correspondences. Proc. VLDB conf., 862–873
- Rahm E (2011) Towards Large-Scale Schema and Ontology Matching. In: Schema Matching and Mapping, Springer
- Rahm E (2016) The case for holistic data integration. In: Proc. ADBIS conf., Springer LNCS 9809
- Rahm E, Bernstein PA (2001) A Survey of Approaches to Automatic Schema Matching. VLDB Journal 10(4):334–350
- Raunich S, Rahm E (2014) Target-driven merging of taxonomies with ATOM. Information Systems, 42:1–14
- Zhong Q, Li H et al (2009) A Gauss function based approach for unbalanced ontology matching. Proc. ACM SIGMOD conf.

Cross-References:

Holistic schema matching,
Large-scale entity resolution