Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Minimizing User Involvement for Accurate Ontology Matching Problems

Anika Schumann and Freddy Lécué
IBM Research - Ireland
Damastown Industrial Estate, Dublin, Ireland
{(firstname.lastname)@ie.ibm.com}

Abstract

Many various types of sensors coming from different complex devices collect data from a city. Their underlying data representation follows specific manufacturer specifications that have possibly incomplete descriptions (in ontology) alignments. This paper addresses the problem of determining accurate and complete matching of ontologies given some common descriptions and their pre-determined high level alignments. In this context the problem of ontology matching consists of automatically determining all matching given the latter alignments, and manually verifying the matching results. Especially for applications where it is crucial that ontologies are matched correctly the latter can turn into a very time-consuming task for the user. This paper tackles this challenge and addresses the problem of computing the minimum number of user inputs needed to verify all matchings. We show how to represent this problem as a reasoning problem over a bipartite graph and how to encode it over pseudo Boolean constraints. Experiments show that our approach can be successfully applied to real-world data sets.

Introduction

Handling variations in meaning in description interpretation or ambiguity is an open problem (Agrawal et al. 2008). Either it cannot be solved automatically or the returned solution is not guaranteed to be correct. However, for several applications correctness is crucial. Consider, for example, the matching problem in smart buildings that consists in identifying among thousands of sensor descriptions the ones that are needed as inputs to energy efficiency approaches (Roth et al. 2005). If the matching is done incorrectly the energy saving approaches do not operate correctly and cannot prevent energy waste.

This paper presents an approach for semi-automatically solving matching problems for applications such as the building one for which correctness is crucial. Contrary to existing works (Pavel and Euzenat 2013) in ontology matching we assume that common descriptions and representations are shared among the ontologies to be matched but not all mappings are known. Such problems occur in many real-world scenarios where different systems manufacturers (e.g., in the context of building, energy or water management systems) use state-of-the-art representation / vocabularies for defining specific features of their systems. In most cases manufacturers implement their own ontologies for internal representation and rarely update the core state-of-the-art vocabulary. Therefore each and every domain (e.g., building, energy or water) has multitude variance of their knowledge representation, which differ from one manufacturer to another. Integrating systems from different providers and manufacturers is then a difficult problem. We study this problem of ontology matching given some pre-determined common representation and high-level alignments, i.e. alignments of descriptions from where underlying alignments can be discovered. In our context the two ontologies to be aligned have (some, not all) common structures and having some alignments implies (indirectly) that some underlying (i.e., sub or super) descriptions will be also aligned.

Given these high level alignments, that can be computed using for example the method of (Stoilos, Stamou, and Kollias 2005), our goal is to find the matching of the specific descriptions that are guaranteed to be correct which is a manual, complex and hence time consuming task.

Suppose we would verify ontology matching results by asking the user for each alignment of the result set whether it is correct or not and put his response on record. Our aim is to obtain exactly this record of responses but without the need of having to involve the user for each alignment verification. The problem here is: Which alignments need to be verified by the users such that the user response for the remaining ones can be automatically inferred?

Our approach complements existing approaches in mapping verification but cannot be compared to them: Mapping verification approaches such as (Cruz, Stroe, and Palmonari 2012; Jiménez-Ruiz et al. 2013; Shi et al. 2009) determine iteratively *individual mappings* that need to be verified in order to *improve the matching result*. In contrast, we determine *simultaneously the set of mappings* that need to be verified in order to *verify the complete matching result*. For our problem this simultaneous computation of the mapping set is necessary in order to indeed minimize user involvement rather than verifying the mapping result in a greedy way.

Figure 1 illustrates our problem setting. In this example we have the three descriptions A, X, and Y that are part of the common domain vocabulary, and descriptions $A1$, $A2$, B, $B1$, $B2$, $B3$, and $B4$ that are part of the manufacturer

specification. There is an alignment between the high level descriptions A and B and our aim is to find the alignments for descriptions $A1$ and $A2$ that are guaranteed to be correct. For this task we make use of an ontology matching approach like (Stoilos, Stamou, and Kollias 2005) that employs variation metrics for computing similarity scores.

Instead of presenting the entire ontology matching result to users we iteratively ask them in each round to verify the correctness of a single mapping. Our aim is to minimize user involvement, i.e. the number of rounds needed for verifying the entire result. We approach it by encoding all conditions for automatically inferring user responses over pseudo Boolean constraints. Based on them we then define the objective function for our problem and solve it using an efficient maximum satisfiability (max-SAT) solver (Heras, Larrosa, and Oliveras 2008).

The paper is organized as follows: We start by introducing a use case and by providing an overview of our approach. Then we show how to compute and organize alignments which may need to be validated by the user. Our approach for minimizing the number of alignments that require user verification is presented in the following section. The paper concludes with an experimental evaluation.

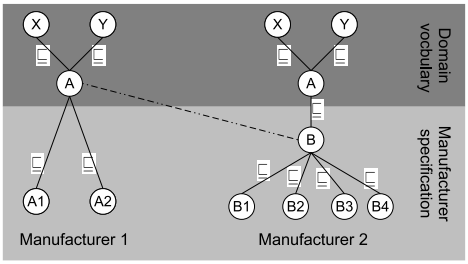

Figure 1: Two simple ontologies and an alignment.

Use Case: Smart Buildings

We start by presenting our building use case to which we will refer throughout the paper. Buildings are responsible for 40% of the energy in industrialized countries and 15% to 30% of it could be saved if energy efficiency approaches would be deployed more widely. The main obstacle for it lies in their high deployment costs (Roth et al. 2005). A significant part of it is due to the need of identifying among thousands of sensor readings in a building the ones that are required as inputs to the energy saving approaches and thus due to the need of solving a problem that can be viewed as an ontology matching problem.

For commercial buildings, related data is managed and stored in Building Management Systems (BMSs). When updating the BMS control strategy for new equipment additions, new data is labeled by most BMS vendor technicians in an ad-hoc manner despite some standardization efforts (ISO16484-3 2005). Therefore it cannot be assumed that ontology matching approaches can compute the correct matching without involving a user. As correctness is crucial for this task the matching problem is solved entirely manual today.

We illustrate our method for deploying the APAR (air-handling unit performance assessment rules) approach (Schein and Bushby 2006) since it is the state-of-the-art fault detection and diagnosis method for air handling units (AHUs), i.e. for one of the main energy consumers in buildings (Castro et al. 2003). Using the domain ontology in (Schumann, Ploennigs, and Gorman 2014) we will show how we can significantly reduce the requirements for user involvement.

Overview of the Approach

Figure 2 presents an overview of our approach. Its on-line part starts with the computation of the ontology matching result (OMR), noted $\mathbb{M}$. We then compute the smallest set of matchings that the user needs to verify such that the user responses for the remaining ones can be automatically inferred (step (2)). These matchings are iteratively presented to the user (loop (2)-(5)) as long as they are consistent with $\mathbb{M}$, i.e. as long as the user verifies that the alignments of the matching result are correct.

Once we have collected consistent feedback for all user requests the OMR is verified and the approach terminates. Each time the user provides feedback that is inconsistent with $\mathbb{M}$ we adapt the OMR accordingly, i.e. we remove the inconsistent alignment and check whether it can be replaced by an alignment whose similarity score is above a given threshold. If this is the case this alignment is added to the OMR (step (1)). Then we compute the smallest set of additional matching that needs to be verified by the user to validate the modified OMR (step 2).

The computation of the latter, i.e. the minimum user request set, is based on an alignment graph that represents all possible alignments between a source ontology (BMS labels for our use case) and a target ontology (APAR labels for our use case) and the dependencies among alignments.

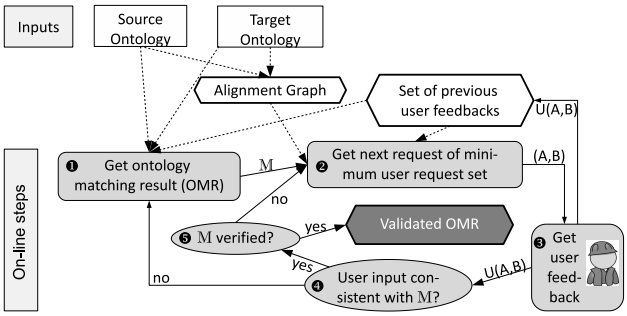

Figure 2: Overview of Approach. Dashed lines correspond to required inputs and solid lines to the flow of the approach.

Computing and Organizing User Requests

A user request corresponds to a matching $m = (A, B)$ between a concept A of the target ontology and a concept B of the source ontology. The user response states either that the matching is correct or that it is incorrect. Only matching whose correctness cannot be automatically inferred need to be presented to the user.

Automatically verifiable matchings correspond to those whose descriptions are semantically strictly more (resp. less) similar than that of a matching which has been previously verified by the user as correct (resp. incorrect). We will

$SupplyAT \equiv Tmp \sqcap \exists loc.Building \sqcap \exists supplyBy.Fan$
$ReturnAT \equiv Tmp \sqcap \exists loc.Area$
$ExtAT \equiv Tmp \sqcap \exists rely.Pressure \sqcap \exists supplyBy.Blower$
$\exists loc.Building \equiv \exists loc.Area \sqcap \exists coilUnit.CoilUnit$

Figure 3: Extract of $\mathcal{EL}^{++}$ Domain Ontology $\mathcal{T}$. AT denotes air temperature, Tmp refers to concept temperature.

exploit such dependencies among matching for computing minimum user request sets (see Section 4).

Now, we briefly review DLs basics and the logic we adopt. Then we formally state the conditions under which matchings can be verified automatically by defining the description matching problem and a subsumption-based hierarchy.

Semantic Representation

We use DL as a formal knowledge representation language for ontologies and illustrate our work with the DL $\mathcal{EL}^{++}$ (Baader, Brandt, and Lutz 2005) where satisfiability and subsumption are decidable. These properties are required for efficiently performing the semantic reasoning for our matching problem. Figure 3 shows an extract of the ontology for our use case expressed in $\mathcal{EL}^{++}$. For example, supply air temperature ($SupplyAT$) is a Temperature (Tmp), cached by a sensor, located in at least a Building ($\exists loc.Building$) which is supplied by at least a fan ($\exists supplyBy.Fan$).

We use the DL constructive reasoning *difference* (or subtraction operation) (Brandt, Küsters, and Turhan 2002) for comparing our $\mathcal{EL}^{++}$ DL-based descriptions that differ only from a few properties (among a dozen and more). While this is not intuitive in other ontology matching problems, it roots in contexts where sensor and physical device manufacturers base their data model on standard ontologies with some significant in-house extensions. Therefore mapping models from two manufacturers can be established by evaluating their difference, which is more appropriate in this context. A compact representation of the DL difference $X \backslash Y$ between two DL concepts X and Y defined through axioms in $\mathcal{T}$ (e.g. Figure 3) is:

$$X \backslash Y \doteq \min_{\preceq_d} \{Z | Z \sqcap Y \equiv X \sqcap Y\} \quad (1)$$

(1) refers to information required by X but not provided by Y. For instance, the difference $SupplyAT \setminus ReturnAT$ of the two $\mathcal{EL}^{++}$ concepts $SupplyAT$ and $ReturnAT$ consists in difference $(Tmp \sqcap \exists loc.Building \sqcap \exists supplyBy.Fan) \setminus (Tmp \sqcap \exists loc.Area)$. The concept $\exists loc.Building$ corresponds to $\exists loc.Area \sqcap \exists coilUnit.CoilUnit$. So their difference is: $\exists supplyBy.Fan \sqcap \exists coilUnit.CoilUnit$. The computation of (1) is elaborated with respect to the subdescription ordering $\preceq_d$ (Küsters 2001), which deals with syntactical redundancies. In other words the concept with the minimal number of properties and concepts in its description is returned.

Automatic Verification of Matchings

Matchings m of the form (A, B) can automatically be verified as incorrect if the two descriptions have no semantic similarity. Two descriptions from the two ontologies (to be aligned) can be unrelated, at least from a subsumption perspective. All other matchings, i.e. those *comparable* under $\sqsubseteq_\mathcal{T}$, could possibly be correct. They are solutions to the description matching problem that is defined for a description A of the target ontology and a set of descriptions $\mathbf{B}$ of the source ontology.

Definition 1 ***(Description Matching Problem - DMP)*** *Let $\mathcal{L}$ be a DL; A, $\mathbf{B}$ be respectively a concept and a set of concepts in $\mathcal{L}$, $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. A Description Matching Problem, denoted as DMP$\langle \mathcal{L}, \mathbf{B}, A, \mathcal{T} \rangle$, consists in retrieving concepts $\mathbf{B}' \subseteq \mathbf{B}$ such that $\forall B \in \mathbf{B}'$:*

$$\mathcal{T} \not\models A \sqcap B \sqsubseteq \bot \quad (2)$$

This ensures that the conjunction between concepts $B \in \mathbf{B}'$ and A is satisfiable, so semantically possible matching $m = (A, B)$ are retrieved and then incompatible concepts in B, (i.e. $\mathcal{T} \models A \sqcap B \sqsubseteq \bot$), are automatically discarded as incorrect.

In order to reduce the number of user feedbacks required for verifying which of the DMP solutions are indeed correct we compare them by first evaluating how concepts of B' are different from A (Definition 1) and then organizing their difference along a $\sqsubseteq_\mathcal{T}$-hierarchy. It captures the hierarchy of matchings $m = (A, B)$ i.e., all pairs (A', B') such that the difference between A and B is more specific than the difference of A' and B'. The latter ensures to capture more "loose" matching or "less specific" matching. In other words (A', B') is still a matching but there exists a more general matching (A, B) whose difference is more specific.

Definition 2 ***($\sqsubseteq_\mathcal{T}$-Hierarchy of Matchings)*** Let $\mathcal{L}$ be a DL; A, A', $\hat{\mathbf{B}}$ =DMP$\langle \mathcal{L}, \mathbf{B}, A, \mathcal{T} \rangle$, $\tilde{\mathbf{B}}$ =DMP$\langle \mathcal{L}, \mathbf{B}, A', \mathcal{T} \rangle$ be respectively two concepts and two sets of concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$. The hierarchy of a matching (A, B) with $B \in \hat{\mathbf{B}}$ is defined as:

$$H(A,B) = \{(A', B') | B' \in \tilde{\mathbf{B}} \text{ and } \mathcal{T} \not\models \mathbf{B} \equiv \tilde{\mathbf{B}} \text{ and } \mathcal{T} \models (A \backslash B) \sqsubseteq (A' \backslash B')\} \quad (3)$$

This definition ensures the comparison of matchings where the concept of "hierarchy of matches" is a set of relations between matches i.e., concept couples of the form $m = (A', B')$. (3) ensures that the concepts of a matching $m = (A, B') \in H(A, B)$ are semantically strictly more similar than those of matching $m' = (A, B)$. It implies that m has additional evidence for being correct. Thus, if the user has verified a matching m' as correct we can automatically infer that m is also correct. For example, consider the difference between matching $m = (SupplyAT, ReturnAT)$ and $m' = (SupplyAT, ExtAT)$. Here the concepts of matching m are strictly more similar than that of m' since:

$$\begin{aligned} &SupplyAT \backslash ExtAT \\ &\doteq \exists supplyBy.Fan \sqcap \exists loc.Building \\ &\doteq \exists supplyBy.Fan \sqcap \exists coilUnit.CoilUnit \sqcap \exists loc.Area \\ &\doteq SupplyAT \backslash ReturnAT \sqcap \exists loc.Area \\ &\sqsubseteq SupplyAT \backslash ReturnAT. \end{aligned}$$

Note that we take all matchings into account that could possibly be correct. Similarity scores are used only to obtain the initial ontology matching result. They are not used for discarding matchings.

Alignment Graph

As shown in Figure 2 we base our minimization approach on an alignment graph that represents the possible alignments, their dependencies, and further information that is typically available for matching problems in the manufacturer context and that can be exploited to further reduce the requirements for user input. This includes knowledge about descriptions that are guaranteed to have an alignment in the source ontology. Consider, for example, our use case where we do ontology matching to deploy an energy saving approach for AHUs. This implies that we have AHUs in our building and hence AHU descriptions in our ontology.

Definition 3 ***(Alignment Graph)*** *The alignment graph* $G = (\mathbf{A}, \mathbf{B}, M, \delta, H, \mathcal{T}, \mathbf{A}')$ *is a bipartite graph where:*

- $\mathbf{A}, \mathbf{B}$*: are two disjoint sets of vertices corresponding to the target and source ontology,*
- $M \subseteq \mathbf{A} \times \mathbf{B}$*: is the set of edges corresponding to the semantically possible matching between descriptions of the target and source ontology (Definition 1),*
- δ*: is the degree of a vertex, i.e. it corresponds to the number of edges incident to a vertex* ($\delta : \mathbf{A} \cup \mathbf{B} \mapsto \mathbb{N}^+$)*,*
- H *is the hierarchy of matchings (Definition 2),*
- $\mathcal{T}$ *is the set of axioms, and*
- $\mathbf{A}' \subseteq \mathbf{A}$ *is the set of vertices that is guaranteed to have an alignment with a description of the source ontology.*[1]

We also write $H(m)$ to denote the the set of subsumers of m. $M(v)$ denotes the set of matchings starting in vertex $v \in \mathbf{A} \cup \mathbf{B}$. Figure 4 illustrates an alignment graph. We will use the graph as a running example throughout the next section. Thus, for quick reference, we have also added identifiers A_1 to A_3 for the APAR descriptions, and B_1 to B_4 for the BMS descriptions. All of the APAR descriptions are guaranteed to have an alignment with a BMS description, i.e. $\{A_1, \ldots, A_3\} \in \mathbf{A}'$.

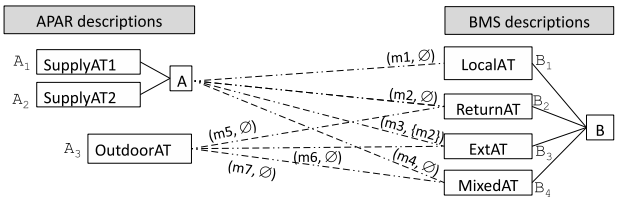

Figure 4: Example of an Alignment Graph: Each edge $m = (A, B)$ is labeled with a tuple consisting of the matching identifier m, and the set H. Only for ease of presentation $B_1 \ldots, B_4$ are connected to A rather than A_1 and A_2.

We presented a way to identify and organize all semantically possible matchings. The next section tackles the matching problem from a complementary angle and describes the computation of the correct matching set by taking user feedbacks into account.

[1] $\mathbf{A}'$ may be empty.

Minimizing User Involvement

We now address the problem of computing the correct label matching solution with minimum user requests. A naive approach would iteratively select arbitrary matchings for user validation until the ontology matching result is verified. We reduce the number of user consultations by exploiting $\sqsubseteq_{\mathcal{T}}$-hierarchy and further information that is typically available for matching problems in the manufacturer context. All these conditions for automatically verifying mappings we encode over pseudo Boolean constraints. This allows us to use efficient max-SAT solvers (Heras, Larrosa, and Oliveras 2008) for minimizing user involvement.

Computation of Minimum User Request Set

This section describes the computation of the minimum set of user inputs U which is required to validate the ontology matching result $\mathbb{M}$ that is retrieved using the approach in (Stoilos, Stamou, and Kollias 2005). Now the problem is not to determine whether a matching is correct or not but to determine whether the entire matching set $\mathbb{M}$ is correct or not.

To model our problem as max-SAT problem over Boolean variables we introduce for each matching m a set of Boolean temporal variables $\{t_m^0, \ldots, t_m^z\}$ where $t_m^0 = true$ implies that matching m was verified by users and $t_m^k = true$ implies that m was automatically verified in time step $k > 0$.

In order to generalize our problem formalization we introduce function $\eta : \mathbf{A} \mapsto \mathbb{N}^+$ that maps any description to the number of specific descriptions that are guaranteed to have an alignment with a description from the source ontology and that have the same high level description. If none of the descriptions is guaranteed to have such an alignment we set $\eta(A)$ to $\delta(A)$.

The general problem formalization is given in Problem 1. (4) states that we aim to minimize the truth assignments to variables t_m^0, so the number of required user inputs is minimized. Constraints (5) ensures that each matching is verified in exactly one time step and (6) states that an incorrect matching $m = (A, B) \notin \mathbb{M}$ can be automatically verified as false[2] in time step k, iff at least one of the conditions is satisfied:

i a matching that subsumes m was previously verified as false, i.e. $\bigvee_{l=0,\ \hat{m}:m\in H(\hat{m})\setminus\mathbb{M}}^{k-1} t_{\hat{m}}^l$ holds,

ii a matching starting in b was previously verified as correct, i.e. is in $\mathbb{M}$: $\bigvee_{l=0,\ \check{m}\in M(B)\cap\mathbb{M}}^{k-1} t_{\check{m}}^l$ holds,

iii given those matching that have already been verified as correct, m cannot be correct given that only $\eta(A)$ correct matchings exist for A. More precisely, the number n of previously verified correct matching that are not subsumed by m and that start in A is so large that m cannot be correct considering all those matchings that must be correct if m is correct, i.e. those in H(m). That is, if $\eta(A) - |H(m)| \leq n$ holds, and hence if $y_{k,m}$ holds subject to constraint (7). This condition is only satisfiable for applications with $\mathbf{A}' \neq \emptyset$.

[2] Since matchings could be automatically verifiable as correct based on matchings that have previously been verified as incorrect (see (9)) we also seek to verify false matchings automatically.

Problem 1 Minimum User Request Problem $MURP(G, \mathbb{M}, z)$

Constant $z \in \mathbb{N}$
Variables: $x_{k,m}$ and $y_{k,m}$ $\forall k \leq z,\ m = (A, B) \in \mathbb{M}$
for each $m \in M : t_m = \{t_m^0, \ldots, t_m^z\}$ where $t_m^j \in \{0, 1\}$

$$\text{Minimize: } \sum_{m \in M} t_m^0 \quad (4) \qquad \forall t_m \ \sum_{l=0}^{z} t_m^l = 1 \quad (5)$$

$$\forall k \in \{1, \ldots, z\},\ m = (A, B) \notin \mathbb{M}$$

$$\bar{t}_m^k \vee y_{k,m} \vee \bigvee_{l=0,\ \hat{m}: m \in H(\hat{m}) \setminus \mathbb{M}}^{k-1} t_{\hat{m}}^l \vee \bigvee_{l=0,\ \check{m} \in M(B) \cap \mathbb{M}}^{k-1} t_{\check{m}}^l \quad (6)$$

$$-1 \cdot y_{k,m} \cdot (\eta(A) - |H(m)|) + \sum_{l=0}^{k-1} \sum_{\check{m} \in M(A) \cap \mathbb{M} \setminus H(m)} t_{\check{m}}^l \geq 0 \quad (7)$$

$$\forall k \in \{1, \ldots, z\},\ m = (A, B) \in \mathbb{M}$$

$$\bar{t}_m^k \vee x_{k,m} \vee \bigvee_{l=0,\ \hat{m} \in H(m) \cap \mathbb{M}}^{k-1} t_{\hat{m}}^l \quad (8)$$

$$-1 \cdot x_{k,m} \cdot g + \sum_{l=0}^{k-1} \sum_{\acute{m} \in M(A) \setminus \mathbb{M} \text{ with } m \notin H(\acute{m})} t_{\acute{m}}^l \geq 0 \quad (9)$$

$$\text{where } g = \delta(A) - \eta(A) - |\{\hat{m} \text{ s.t. } m \in H(\hat{m})\}|$$

Otherwise, $\bar{t}_m^k$ must be true and hence m is not verifiable in time step k. Constraint (8) states that a correct matching $m = (A, B) \in \mathbb{M}$ can be automatically verified as true in time step k, iff at least one of the conditions is satisfied:

iv a matching that is subsumed by m was previously verified as true, i.e. $\bigvee_{l=0,\ \hat{m} \in H(m) \cap \mathbb{M}}^{k-1} t_{\hat{m}}^l$ holds,

v the number of previously verified incorrect matching that m does not subsume and that start in a is at least as large as the number of incorrect matching starting in a, i.e. if $x_{k,m}$ holds (see (9)).

Again, the latter condition is only satisfiable for applications with $\mathbf{A}' \neq \emptyset$.

Computing the Verified Matching Result

With the ability to compute minimum user request sets we can retrieve the verified solution to the ontology matching problem as shown in Procedure 1.[3] As already described in Section 2 it starts by computing the ontology matching result $\mathbb{M}$ (line 4) and based on that the minimum user request set RUI (line 5) which is computed via a max-SAT solver that also takes previous user feedbacks as input. Then, as long as the user feedback is in line with $\mathbb{M}$ we iteratively request feedback for an element in RUI (lines 7–10). Otherwise, we adapt the OMR accordingly, i.e. we remove the inconsistent alignment and check whether it can be replaced by an alignment whose similarity score is above a given threshold. If this is the case this alignment is added to $\mathbb{M}$. Then we recompute the minimum user request set by taking previous user feedbacks into account (lines 4–5). Our approach terminates once the user has provided feedback on all user requests (line 3). Then it returns the correct matching set that is composed only of those matchings that were either verified by the user or that can be automatically verified.

Procedure 1 $GetValidatedOMR(G, z)$

```
1: RUI ← GetElem(M)                          // required user inputs
2: U ← ∅                                     // user input set
3: while not(RUI ⊆ U) do
4:    M ← GetOntologyMatchingResult(G, U)
5:    RUI ← GetMaxSATsol(MURP(G, M, z), U)
6:    u ← GetElemIn(M)                       // user input
7:    while (M ⊨ u) and not(RUI ⊆ U) do
8:       r ← GetElement(RUI \ U)
9:       u ← GetUserInput(r)
10:      U ← U ∪ u
11: return M
```

Experimental Evaluation

We are not aware of any work that addresses our problem nor any existing experimental evaluations to which we could compare our work. While the OAEI test cases (http://oaei.ontologymatching.org/2013/) do contain benchmarks for evaluating interactive matching approaches they consider the problem of *improving* the matching result rather than *verifying* it. We do not present any new method for computing or improving matching results. The two problems of *improving* and *verifying* matching results are complementary but they are not comparable.

We therefore evaluated our approach for two use cases for which we had access to the correct matching as verified by a domain expert. The first one is from the building domain. We have evaluated our approach on a commercial site consisting of 9 buildings, all managed by a single BMS with 2,239 descriptions. In order to deploy the APAR rules to its 23 AHUs we need to find for each of the seven high level APAR descriptions[4] the corresponding BMS description, i.e. 139 descriptions.[5] The semantic analysis (Definition 1) determines $1,317$ matchings as potential solutions, and hence as matching set of our alignment graph G. To increase the computational efficiency of our approach we automatically partitioned G into 7 subgraphs with one high level APAR description (and all its potential matchings) each. As the APAR approach has already been applied to that building we had access to the correct ontology matching result. Thus, each time our approach requests a feedback from the user we compared our solution to the correct matching.

[3] As variables RUI and u require initialization they are set to an arbitrary element in M (resp. $\mathbb{M}$).

[4] The APAR approach is based on 20 variables, 7 of which correspond to BMS data points. The others are adjustable thresholds.

[5] Six high level APAR descriptions are each guaranteed to be aligned to 23 BMS descriptions and one APAR description (outdoor air temperature) is guaranteed to be aligned to one BMS description.

Experiments were run on a Linux Dual-Core PC with 2GB of RAM using the Pseudo-Boolean maxSAT solver MiniSAT+ (Een and Sorensson 2006). Off-line reasoning has been performed with CEL DL reasoner (http://lat.inf.tu-dresden.de/systems/cel/) on label descriptions, enriched using an $\mathcal{EL}^{++}$ ontology of the APAR domain (55 concepts, 39 properties; 19 concepts subsume the 36 remaining ones with a maximal depth of 3). Computing semantic matching solutions and their $\sqsubseteq_{\mathcal{T}}$-hierarchies requires 8.4 seconds.

The results are shown in Figure 5. When performing the matching without any semantics techniques the user had to consider $2,239 \times 7 = 15,673$ possible alignments. Utilizing the semantic descriptions and exploiting (2) of the description matching problem reduced the need for user input to 1,317 requests. When taking advantage of ontology matching approaches where the user was asked to verify alignments only if their quality was above a certain threshold only 145 alignments needed to be presented to the user. In contrast, our new approach allowed for the retrieval of the validated matching set by requesting user feedback for 28 alignments only.

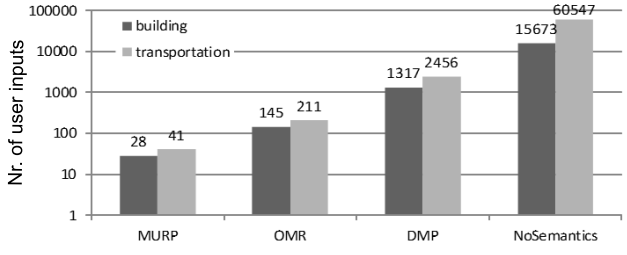

Figure 5: For the building and the transportation use case: Required user inputs for our approach (MURP), for the one considering only alignments that are likely to be correct (OMR), for the one considering only alignments if their corresponding descriptions are semantically similar (DMP), and for the one not utilizing any semantic techniques (NoSemantics).

On average each request was presented to the user every 2.5 seconds, with a maximum of 10.6. This was the case when we had to consider 20,033 clauses and 382 variables for computing the minimum user request set.

As a second use case we evaluated our approach in the context of transportation and particularly traffic sensors. In such a domain sensors do not follow any standard representation but rather specific characteristics that represent their functionalities. It is very common that sensor manufacturers provide sensor descriptions using non-standard vocabularies (to describe them). Similarly to the building domain manufacturers use a common basis of vocabularies and slightly diverge to represent the data that their sensors expose. We specifically notice this common behavior across cities where national manufacturers expose similar functionalities of sensors with changes in the data representation model. Adapting one system from one city to another one could be very expensive and time consuming because data to be ingested by the system does not necessarily fit. In order to ease data ingestion and limit system upgrading, we exploit our approach to guide transportation experts in aligning vocabularies of different sensors from different cities. In the end the same system could be used without tuning its set up. We exploit our approach in the context of transportation loop detector data from Dublin (Ireland) and Bologna (Italy), where both are described in English and refer to some concepts from a common ontology. The objective is to support the end-user in retrieving the appropriate alignment while minimizing the number of requests. The Dublin model consists of 1,234 concepts while the Bologna model consists of 890 concepts descriptions. The core common representation is about 4% of the Bologna model, which all fit the domain loop detector ontology model (123 concepts, 290 properties; 40 concepts subsume the 83 remaining ones with a maximal depth of 6). Computing the semantic matching solutions and their C-hierarchies requires 340.4 seconds. Based on these we could verify the ontology matching result by requesting user feedback for 41 alignments only as shown in Figure 5.

Conclusions, Related & Future Work

We have developed a novel approach for verifying ontology matching results without the need of presenting the complete result to the user. This required the development and integration of semantic reasoning, constraint modeling, and satisfiability solving techniques. Specifically we have used semantic reasoning techniques to compute dependencies among alignments and have shown how an efficient max-SAT solver can be used for minimizing the amount of user involvement. We have also successfully applied our approach to two real-world data sets and have shown that it can indeed reduce user involvement significantly.

To the best of our knowledge the presented approach is the first one that tackles the problem of minimizing the number of user requests for verifying *entire* ontology matching results. It is also the first one that addresses the problem of minimizing user inputs based on a bipartite graph and a set of constraints. From a constraint solving point of view other SAT solvers or mixed integer problem solvers could have also been considered. An evaluation of the different solvers was beyond the scope of the paper. The closest work to our method for computing optimal user requests are active learning approaches (Settles 2009). However, in contrast to the latter that seek to minimize the amount of user input needed for obtaining a good *estimate* of the *probability distribution* underlying the data, we seek to compute the minimum user requests for obtaining the matching set that is *guaranteed* to be correct.

Future work includes extending our approach to ontology matching problems where a complete validation of the matching correctness is not essential and hence were the trade-off between the benefits of having a validated result and the costs for getting it need to be considered. It would also be interesting to integrate crowdsourcing techniques such as Crowdmap (Sarasua, Simperl, and Noy 2012) to get the mappings verified. For applications that require a high level of domain expertise, such as the building one, we would need to resort to a crowd of experts.

References

Agrawal, R.; Ailamaki, A.; Bernstein, P. A.; Brewer, E. A.; Carey, M. J.; Chaudhuri, S.; Doan, A.; Florescu, D.; Franklin, M. J.; Garcia-Molina, H.; Gehrke, J.; Gruenwald, L.; Haas, L. M.; Halevy, A. Y.; Hellerstein, J. M.; Ioannidis, Y. E.; Korth, H. F.; Kossmann, D.; Madden, S.; Magoulas, R.; Ooi, B. C.; O'Reilly, T.; Ramakrishnan, R.; Sarawagi, S.; Stonebraker, M.; Szalay, A. S.; and Weikum, G. 2008. The claremont report on database research. *SIGMOD Rec.* 37(3):9–19.

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the el envelope. In *IJCAI*, 364–369.

Brandt, S.; Küsters, R.; and Turhan, A.-Y. 2002. Approximation and difference in description logics. In *KR*, 203–214.

Castro, N. S.; Schein, J.; Park, C.; Galler, M. A.; Bushby, S. T.; and House, J. M. 2003. Results from simulation and laboratory testing of air handling unit and variable air volume box diagnostic tools. Technical report, U.S. Department of Commerce.

Cruz, I. F.; Stroe, C.; and Palmonari, M. 2012. Interactive user feedback in ontology matching using signature vectors. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, 1321–1324. IEEE.

Een, N., and Sorensson, N. 2006. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation* 2:1–26.

Heras, F.; Larrosa, J.; and Oliveras, A. 2008. Minimaxsat: An efficient weighted max-sat solver. *Journal of Artificial Intelligence Research* 31:1–32.

ISO16484-3. 2005. Building automation and control systems – part 3: Functions.

Jiménez-Ruiz, E.; Meilicke, C.; Grau, B. C.; and Horrocks, I. 2013. Evaluating mapping repair systems with large biomedical ontologies. In *Proc. 26th Description Logics Workshop*.

Küsters, R. 2001. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes in Computer Science*. Springer.

Pavel, S., and Euzenat, J. 2013. Ontology matching: State of the art and future challenges. *IEEE Trans. on Knowl. and Data Eng.* 25(1):158–176.

Roth, K. W.; Westphalen, D.; Feng, M. Y.; Llana, P.; and Quartararo, L. 2005. Energy impact of commercial building controls and performance diagnostics: market characterization, energy impact of building faults and energy savings potential. Technical report, TIX LLC for U.S. Department of Energy.

Sarasua, C.; Simperl, E.; and Noy, N. F. 2012. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *Proceedings of the 11th International Semantic Web Conference*, ISWC '12, 525–541.

Schein, J., and Bushby, S. T. 2006. A hierarchical rule-based fault detection and diagnostic method for hvac systems. *HVAC&R RESEARCH* 1(1):111–125.

Schumann, A.; Ploennigs, J.; and Gorman, B. 2014. Towards automating the deployment of energy saving approaches in buildings. In *International Conference on Embedded Systems for Energy-Efficient Buildings (BuildSys-14)*, 164–167. ACM.

Settles, B. 2009. Active learning literature survey. Technical report, University of Wisconsin-ï£¡Madison.

Shi, F.; Li, J.; Tang, J.; Xie, G.; and Li, H. 2009. Actively learning ontology matching via user interaction. In *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, 585–600. Berlin, Heidelberg: Springer-Verlag.

Stoilos, G.; Stamou, G. B.; and Kollias, S. D. 2005. A string metric for ontology alignment. In *International Semantic Web Conference*, 624–637.