RIO: Minimizing User Interaction in Debugging of Aligned Ontologies *

Patrick Rodler, Kostyantyn Shchekotykhin, Philipp Fleiss, and Gerhard Friedrich

Alpen-Adria Universität, Klagenfurt, 9020 Austria
firstname.lastname@aau.at

Abstract. Efficient ontology debugging is a cornerstone for many activities in the context of the Semantic Web, especially when automatic tools produce (parts of) ontologies such as in the field of ontology matching. The best currently known interactive debugging systems rely upon meta information in terms of fault probabilities, which can speed up the debugging procedure in the good case, but can also have negative impact in the bad case. Unfortunately, assessment of meta information is only possible a-posteriori. Hence, as long as the actual fault is unknown, there is always some risk of suboptimal interactive diagnoses discrimination. As an alternative, one might prefer to rely on a no-risk strategy. In this case, however, possibly well-chosen meta information cannot be exploited, resulting again in inefficient debugging actions. In this work we present a reinforcement learning strategy that continuously adapts its behavior depending on the performance achieved and minimizes the risk of using low-quality meta information. Therefore, this method is suitable for application scenarios where reliable a-priori fault estimates are difficult to obtain. Using faulty ontologies produced by ontology matchers, we show that the proposed strategy outperforms both active learning and no-risk approaches on average w.r.t. required amount of user interaction.

1 Introduction

The foundation for widespread adoption of Semantic Web technologies is a broad community of ontology developers which is not restricted to experienced knowledge engineers. Instead, domain experts from diverse fields should be able to create ontologies incorporating their knowledge as autonomously as possible. The resulting ontologies are required to fulfill some minimal quality criteria, usually consistency, coherency and no undesired entailments, in order to grant successful deployment. However, the correct formulation of logical descriptions in ontologies is an error-prone task which accounts for a need for assistance in ontology development in terms of ontology debugging tools. Things get even worse when independent standalone ontologies describing related domains are unified to a single ontology (called aligned ontology) by adding a set of suitable correspondences (called alignment) between signature elements of the different ontologies. This task is addressed in the field of ontology matching where researchers aim to produce automated tools for the generation of correspondences. Applying such tools, however, often results in inconsistent/incoherent aligned ontologies even though input ontologies (considered separately) do not violate any quality criteria. Moreover, these aligned ontologies may exhibit a very complex fault structure as a consequence of (1) adding many links between the single ontologies at once and since

* This research is funded by Austrian Science Fund (Project V-Know, contract 19996).

(2) the actual fault may be located in the produced alignment and/or in one or more of the single ontologies, e.g. if a correct correpondence between two concepts "activates" a fault in one of the single ontologies. In this vein, many different sources of inconsistency/incoherence could arise (due to (1)) each of which may comprise parts of each single ontology as well as of the alignment (due to (2)). In this work we present an interactive approach dealing with the general problem of locating a fault throughout the entire aligned ontology, not just in the alignment, as addressed by state-of-the-art systems in ontology matching such as CODI [10] or LogMap [5]. Comparison of our method with these systems is inappropriate since they use greedy diagnosis techniques (e.g. [8]), whereas our approach is complete.

Usually, ontology debugging tools [3, 4] use model-based diagnosis [11] to identify sets of faulty axioms, called diagnoses, that need to be modified or deleted in order to meet the imposed quality requirements. The major challenge inherent in the debugging task is often a substantial number of alternative diagnoses. This problem has been addressed in [12] by proposing an active learning debugging method which queries the user (e.g. a domain expert) for additional information about the intended ontology.

In a debugging scenario involving a faulty ontology developed by one user, the meta information might be extracted from the logs of previous sessions, if available, or specified by the user based on their experience w.r.t. own faults. However, in scenarios involving automatized systems producing (parts of) ontologies as in ontology matching, the choice of reasonable meta information is rather unclear. If, on the one hand, an active learning method is used relying on a guess of the meta information, this might result in an overhead w.r.t. user interaction of more than 2000%. If one wants to play it safe, on the other hand, by deciding not to exploit any meta information at all, this might also result in substantial extra time and effort for the user. So, in the light of current state-of-the-art one is spoilt for choice between debugging strategies with high potential but also high risk, or methods with no risk but also no potential.

In this work we present an ontology debugging approach with high potential and low risk, which allows to minimize user interaction throughout a debugging session on average, without depending on high-quality meta information. By virtue of its reinforcement learning capability, our approach is optimally suited for debugging aligned ontologies, where only vague or no meta information is available. On the one hand, our method takes advantage of the given meta information as long as good performance is achieved. On the other hand, it gradually gets more independent of meta information if suboptimal behavior is measured. The method constantly improves the quality of meta information and adapts a risk parameter based on the new information obtained by querying the user. This means that, in case of good meta information, the performance of our method will be close to the performance of the active learning method, whereas, in case of bad meta information, the achieved performance will approach the performance of the risk-free strategy. So, our approach can be seen as a risk optimization strategy (RIO) which combines the benefits of active learning and risk-free strategies. Experiments on two datasets of faulty ontologies produced by ontology matching systems show the feasibility, efficiency and scalability of RIO. The evaluation of these experiments will manifest that, on average, RIO is the best choice of strategy for both good and bad meta information with savings in terms of user interaction of up to 80%.

The problem specification, basic concepts and a motivating example are provided in Section 2. Section 3 explains the suggested approach and gives implementation details. Evaluation results are described in Section 4. Section 5 concludes.

2 Basic Concepts and Motivation

Ontology debugging deals with the following problem: Given is an ontology $\mathcal{O}$ which does not meet postulated requirements R.[1] $\mathcal{O}$ is a set of axioms formulated in some monotonic knowledge representation language, e.g. OWL. The task is to find one of generally many alternative subsets of axioms in $\mathcal{O}$, called target diagnosis $\mathcal{D}_t \in \mathcal{O}$, that needs to be altered or eliminated from the ontology such that the resulting ontology meets the given requirements and has the intended semantics. The general debugging setting we consider also envisions the opportunity for the user to specify some background knowledge $\mathcal{B}$, i.e. a set of axioms which are known to be correct. Moreover, we allow definition of a set P of positive (entailed) and a set N of negative (non-entailed) test cases, where each test case is a set of axioms.

More formally, ontology debugging can be defined in terms of conditions a target ontology must fulfill, which leads to the definition of a diagnosis problem instance, for which we search for solutions, i.e. diagnoses:

Definition 1 (Target Ontology, Diagnosis Problem Instance). *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ denote an ontology consisting of a set of terminological axioms $\mathcal{T}$ and a set of assertional axioms $\mathcal{A}$, P a set of positive test cases, N a set of negative test cases, $\mathcal{B}$ a set of background knowledge axioms, and R a set of requirements to an ontology. Then an ontology $\mathcal{O}_t$ is called* target ontology *iff all the following conditions are fulfilled:*

$$\forall\, r \in R:\ \mathcal{O}_t \cup \mathcal{B} \textit{ fulfills } r$$
$$\forall\, p \in P:\ \mathcal{O}_t \cup \mathcal{B} \models p$$
$$\forall\, n \in N:\ \mathcal{O}_t \cup \mathcal{B} \not\models n$$

The tuple $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$ is called a diagnosis problem instance *iff $\mathcal{B} \cup (\bigcup_{p \in P} p) \not\models n$ for all $n \in N$ and $\mathcal{O}$ is not a target ontology, i.e. $\mathcal{O}$ violates at least one of the conditions above.*

Definition 2 (Diagnosis). *We call $\mathcal{D} \subseteq \mathcal{O}$ a* diagnosis *w.r.t. a diagnosis problem instance $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$ iff there exists a set of axioms $EX_{\mathcal{D}}$ such that $(\mathcal{O} \setminus \mathcal{D}) \cup EX_{\mathcal{D}}$ is a target ontology. A diagnosis $\mathcal{D}$ is* minimal *iff there is no $\mathcal{D}' \subset \mathcal{D}$ such that $\mathcal{D}'$ is a diagnosis. A diagnosis $\mathcal{D}$ gives complete information about the correctness of each axiom $ax_k \in \mathcal{O}$, i.e. all $ax_i \in \mathcal{D}$ are assumed to be faulty and all $ax_j \in \mathcal{O} \setminus \mathcal{D}$ are assumed to be correct. The set of all minimal diagnoses is denoted by* $\mathbf{D}$.

The identification of an extension $EX_{\mathcal{D}}$, accomplished e.g. by some learning approach, is a crucial part of the ontology repair process. However, the formulation of a complete extension is outside the scope of this work where we focus on computing diagnoses. Following the approach suggested in [12], we approximate $EX_{\mathcal{D}}$ by the set $\bigcup_{p \in P} p$.
Example: Consider the OWL ontology $\mathcal{O}$ encompassing the following terminology $\mathcal{T}$:

$ax_1 : PhD \sqsubseteq Researcher$ $\quad$ $ax_4 : Student \sqsubseteq \neg DeptMember$
$ax_2 : Researcher \sqsubseteq DeptEmployee$ $\quad$ $ax_5 : PhDStudent \sqsubseteq PhD$
$ax_3 : PhDStudent \sqsubseteq Student$ $\quad$ $ax_6 : DeptEmployee \sqsubseteq DeptMember$

[1] Throughout the paper we consider debugging of inconsistent and/or incoherent ontologies, i.e. whenever not stated explicitly we assume $R = \{\text{consistency, coherency}\}$.

and an assertional axiom $\mathcal{A} = \{PhDStudent(s)\}$. Then $\mathcal{O}$ is inconsistent since it describes a PhD student as both a department member and not.

Let us assume that the assertion $PhDStudent(s)$ is considered as correct and is thus added to the background theory, i.e. $\mathcal{B} = \mathcal{A}$, and both sets P and N are empty. Then, the set of minimal diagnoses $\mathbf{D} = \{\mathcal{D}_1 : [ax_1], \mathcal{D}_2 : [ax_2], \mathcal{D}_3 : [ax_3], \mathcal{D}_4 : [ax_4], \mathcal{D}_5 : [ax_5], \mathcal{D}_6 : [ax_6]\}$ for the given problem instance $\langle \mathcal{T}, \mathcal{A}, \emptyset, \emptyset \rangle$. $\mathbf{D}$ can be computed by a diagnosis algorithm such as the one presented in [3].

With six diagnoses for six ontology axioms, this example might already give an idea that in many cases the number of diagnoses $\mathbf{D}$ can get very large. Without any prior knowledge, each of the diagnoses in $\mathbf{D}$ is equally likely to be the target diagnosis $\mathcal{D}_t$, that is selected by a user in order to formulate the intended ontology $\mathcal{O}_t := (\mathcal{O} \setminus \mathcal{D}_t) \cup EX_{\mathcal{D}_t}$. Identification of $\mathcal{D}_t$ can be accomplished by means of queries [12]. Thereby, the fact is exploited that ontologies $\mathcal{O} \setminus \mathcal{D}_i$ and $\mathcal{O} \setminus \mathcal{D}_j$ resulting in application of different diagnoses $\mathcal{D}_i, \mathcal{D}_j \in \mathbf{D}$ $(\mathcal{D}_i \neq \mathcal{D}_j)$ entail different sets of logical axioms.

Definition 3 (Query). *A set of logical axioms X_j is called a* query *iff there exists a set of diagnoses $\emptyset \subset \mathbf{D}' \subset \mathbf{D}$ such that X_j is entailed by each ontology in $\{\mathcal{O}_i^* \mid \mathcal{D}_i \in \mathbf{D}'\}$ where $\mathcal{O}_i^* := (\mathcal{O} \setminus \mathcal{D}_i) \cup \mathcal{B} \cup \bigcup_{p \in P} p$. Asking a query X_j to a user means asking them $(\mathcal{O}_t \models X_j?)$. The set of all queries w.r.t. $\mathbf{D}$ is denoted by $\mathbf{X_D}$.*[2]

Each query X_j partitions the set of diagnoses $\mathbf{D}$ into $\langle \mathbf{D}_j^P, \mathbf{D}_j^N, \mathbf{D}_j^\emptyset \rangle$ such that $\mathbf{D}_j^P = \{\mathcal{D}_i \mid \mathcal{O}_i^* \models X_j\}$, $\mathbf{D}_j^N = \{\mathcal{D}_i \mid \mathcal{O}_i^* \cup X_j \text{ is inconsistent}\}$ and $\mathbf{D}_j^\emptyset = \mathbf{D} \setminus (\mathbf{D}_j^P \cup \mathbf{D}_j^N)$. If the answering of queries by a user u is modeled as a function $a_u : \mathbf{X} \to \{yes, no\}$, then the following holds: If $a_u(X_j) = yes$, then X_j is added to the positive test cases, i.e. $P \leftarrow P \cup \{X_j\}$, and all diagnoses in $\mathbf{D}_j^N$ are *rejected*. Given that $a_u(X_j) = no$, then $N \leftarrow N \cup \{X_j\}$ and all diagnoses in $\mathbf{D}_j^P$ are *rejected*. For the example ontology $\mathcal{O}$, we could, e.g., ask the user the query $X_1 := (\mathcal{O}_t \models \{DeptEmployee(s), Student(s)\}?)$ with the associated partition $\langle \mathbf{D}_1^P, \mathbf{D}_1^N, \mathbf{D}_1^\emptyset \rangle = \langle \{\mathcal{D}_4, \mathcal{D}_6\}, \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_5\}, \emptyset \rangle$. A negative answer would then eliminate $\{\mathcal{D}_4, \mathcal{D}_6\}$.

Definition 4 (Diagnosis Discrimination). ***Given*** *the set of diagnoses $\mathbf{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ w.r.t. $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$ and a user u,* ***find*** *a sequence $(X_1, \ldots, X_q)$ of queries $X_i \in \mathbf{X}$ with minimal q, such that $\mathbf{D} = \{\mathcal{D}_t\}$ after assigning $X_{i(i=1\ldots,q)}$ each to either P iff $a_u(X_i) = yes$ or N iff $a_u(X_i) = no$.*[3]

A set of queries for a given set of diagnoses $\mathbf{D}$ can be generated as shown in Algorithm 1. In each iteration, for a set of diagnoses $\mathbf{D}^P \subset \mathbf{D}$, the generator gets a set of logical descriptions X that are entailed by each ontology $\mathcal{O}_i^*$ where $\mathcal{D}_i \in \mathbf{D}^P$ (function GETENTAILMENTS). When we speak of entailments, we always address the output computed by the classification and realization services of a reasoner [1, p.323 ff.]. These axioms X are then used to classify the remaining diagnoses in $\mathbf{D} \setminus \mathbf{D}^P$ in order to obtain the partition $\langle \mathbf{D}^P, \mathbf{D}^N, \mathbf{D}^\emptyset \rangle$ associated with X. Then, together with its partition, X is added to the set of queries $\mathbf{X}$. Note that in real-world applications, investigation of all possible subsets of the set $\mathbf{D}$ might be infeasible. Thus, it is common to approximate

[2] For the sake of simplicity, we will use $\mathbf{X}$ instead of $\mathbf{X_D}$ throughout this work because the $\mathbf{D}$ associated with $\mathbf{X}$ will be clear from the context.

[3] Since the user u is assumed fixed throughout a debugging session and for brevity, we will use a_i equivalent to $a_u(X_i)$ in the rest of this work.

Algorithm 1: Query Generation

Input: diagnosis problem instance $\langle \mathcal{O}, \mathcal{B}, P, N \rangle$, set of diagnoses $\mathbf{D}$
Output: a set of queries and associated partitions $\mathbf{X}$

foreach $\mathbf{D}^P \subset \mathbf{D}$ **do**
  $X \leftarrow$ `getEntailments`$(\mathcal{O}, \mathcal{B}, P, \mathbf{D}^P)$;
  if $X \neq \emptyset$ **then**
    foreach $\mathcal{D}_r \in \mathbf{D} \setminus \mathbf{D}^P$ **do**
      if $\mathcal{O}_r^* \models X$ **then** $\mathbf{D}^P \leftarrow \mathbf{D}^P \cup \{\mathcal{D}_r\}$;
      else if $\mathcal{O}_r^* \cup X$ *is inconsistent* **then** $\mathbf{D}^N \leftarrow \mathbf{D}^N \cup \{\mathcal{D}_r\}$;
      else $\mathbf{D}^\emptyset \leftarrow \mathbf{D}^\emptyset \cup \{\mathcal{D}_r\}$;
    $\mathbf{X} \leftarrow \mathbf{X} \cup \left\langle X, \mathbf{D}^P, \mathbf{D}^N, \mathbf{D}^\emptyset \right\rangle$
return $\mathbf{X}$;

the set of all minimal diagnoses by a set of *leading diagnoses*. This set comprises a predefined number n of minimal diagnoses.

The query generation algorithm returns a set of queries $\mathbf{X}$ that generally contains a lot of elements. Therefore the authors in [12] suggested two query selection strategies.

Split-in-half strategy, selects the query X_j which minimizes the scoring function $sc_{split}(X_j) = \left||\mathbf{D}_j^P| - |\mathbf{D}_j^N|\right| + |\mathbf{D}_j^\emptyset|$, i.e. this strategy prefers queries which eliminate half of the diagnoses independently of the query outcome.

Entropy-based strategy, uses information about prior probabilities for a user to make a fault in an axiom [12]. The fault probabilities of axioms $p(ax_i)$ can in turn be used to determine fault probabilities of diagnoses $\mathcal{D}_i \in \mathbf{D}$. The strategy is then to select the query which minimizes the expected entropy of the set of leading diagnoses $\mathbf{D}$ after the query is answered. This means that the expected uncertainty is minimized and the expected information gain is maximized. According to [7], this is equivalent to choosing the query X_j which minimizes the scoring function $sc_{ent}(X_j) = \sum_{a_j \in \{yes,no\}} p(a_j) \log_2 p(a_j) + p(\mathbf{D}_j^\emptyset) + 1$. After each query X_j, the diagnosis probabilities are updated according to the Bayesian formula [12].

A diagnosis discrimination procedure can use either of the strategies to identify the target diagnosis $\mathcal{D}_t$. The result of the evaluation in [12] shows that entropy-based query selection reveals better performance than split-in-half in most of the cases. However, split-in-half proved to be the best strategy in situations when only vague priors are provided, i.e. the target diagnosis $\mathcal{D}_t$ has rather low prior fault probability. Therefore selection of prior fault probabilities is crucial for successful query selection and minimization of user interaction.

Example (continued): In our example, if the user specifies $p(ax_{i\,(i=1,\ldots,4)}) = 0.001$, $p(ax_5) = 0.1$ and $p(ax_6) = 0.15$. Given $\mathcal{D}_t := \mathcal{D}_2$, the no-risk strategy sc_{split} (*three queries*) is more suitable than sc_{ent} (*four queries*) because the fault probabilities disfavor $\mathcal{D}_2$. If $\mathcal{D}_t := \mathcal{D}_6$, then the entropy-based strategy requires only *two queries* while it takes split-in-half *three queries* due to favorable fault probabilities.

We learn from this example that the best choice of discrimination strategy depends on the quality of the meta information in terms of prior fault probabilities. In cases where adequate meta information is not available and hard to estimate, e.g. Ontology Matching, the inappropriate choice of strategy might cause tremendous extra effort for the user interacting with the debugging system.

3 Risk Optimization Strategy for Query Selection

The proposed Risk Optimization Algorithm (RIO) extends entropy-based query selection strategy with a dynamic learning procedure that learns by reinforcement how to select optimal queries. Moreover, it continually improves the prior fault probabilities based on new knowledge obtained through queries to a user. The behavior of our algorithm can be co-determined by the user. The algorithm takes into account the user's doubt about the priors in terms of the initial cautiousness c as well as the cautiousness interval $[\underline{c}, \overline{c}]$ where $c, \underline{c}, \overline{c} \in [c_{\min}, c_{\max}] := [0, \lfloor |\mathbf{D}|/2 \rfloor / |\mathbf{D}|]$, $\underline{c} \leq c \leq \overline{c}$ and $\mathbf{D}$ contains at most n leading diagnoses (see Section 2). The interval $[\underline{c}, \overline{c}]$ constitutes the set of all admissible cautiousness values the algorithm may take during the debugging session. High trust in the prior fault probabilities is reflected by specifying a low minimum required cautiousness $\underline{c}$ and/or a low maximum admissible cautiousness $\overline{c}$. If the user is unsure about the rationality of the priors this can be expressed by setting $\underline{c}$ and/or $\overline{c}$ to a higher value. Intuitively, $\underline{c} - c_{\min}$ and $c_{\max} - \overline{c}$ represent the minimal desired difference in performance to a high-risk (entropy) and no-risk (split-in-half) query selection, respectively.

The relationship between cautiousness c and queries is formalized by the following definitions:

Definition 5 (Cautiousness of a Query). *We define the* cautiousness $caut(X_i)$ *of a query X_i as follows:*

$$caut(X_i) := \frac{min\left\{|\mathbf{D}_i^P|, |\mathbf{D}_i^N|\right\}}{|\mathbf{D}|} \in \left[0, \frac{\left\lfloor \frac{|\mathbf{D}|}{2} \right\rfloor}{|\mathbf{D}|}\right]$$

A query X_i is called braver *than query X_j iff $caut(X_i) < caut(X_j)$. Otherwise X_i is called* more cautious *than X_j. A query with highest possible cautiousness is called* no-risk query.

Definition 6 (Elimination Rate). *Given a query X_i and the corresponding answer $a_i \in \{yes, no\}$, the* elimination rate $e(X_i, a_i)$ *is defined as follows:*

$$e(X_i, a_i) = \begin{cases} \frac{|\mathbf{D}_i^N|}{|\mathbf{D}|} & \text{if } a_i = yes \\ \frac{|\mathbf{D}_i^P|}{|\mathbf{D}|} & \text{if } a_i = no \end{cases}$$

The answer a_i to a query X_i is called favorable *iff it maximizes the elimination rate $e(X_i, a_i)$. Otherwise a_i is called* unfavorable.

So, the cautiousness $caut(X_i)$ of a query X_i is exactly the minimal elimination rate, i.e. $caut(X_i) = e(X_i, a_i)$ given that a_i is the unfavorable query result. Intuitively, the user-defined cautiousness c is the minimum proportion of diagnoses in $\mathbf{D}$ which should be eliminated by the successive query. For braver queries the interval between minimum and maximum elimination rate is larger than for more cautious queries. For no-risk queries it is minimal.

Definition 7 (High-Risk Query). *Given a query X_i and cautiousness c, then X_i is called a* high-risk query *iff $caut(X_i) < c$, i.e. the cautiousness of the query is lower than the algorithm's current cautiousness value c. Otherwise, X_i is called* non-high-risk query. *By $HR_c(\mathbf{X}) \subseteq \mathbf{X}$ we denote the set of all high-risk queries w.r.t. c. For given cautiousness c, the set of all queries $\mathbf{X}$ can be partitioned in high-risk queries and non-high-risk queries.*

Given a user's answer a_s to a query X_s, the cautiousness c is updated depending on the elimination rate $e(X_s, a_s)$ by $c \leftarrow c + c_{adj}$ where $c_{adj} := 2\,(\overline{c} - \underline{c})\,adj$ denotes the cautiousness adjustment factor. The factor $2\,(\overline{c} - \underline{c})$ is a scaling factor that simply regulates the extent of the cautiousness adjustment depending on the interval length $\overline{c} - \underline{c}$. The more crucial factor in the formula is adj which indicates the sign and magnitude of the cautiousness adjustment.

$$adj := \frac{\left\lfloor \frac{|\mathbf{D}|}{2} - \epsilon \right\rfloor}{|\mathbf{D}|} - e(X_s, a_s)$$

where $\epsilon \in (0, \frac{1}{2})$ is a constant which prevents the algorithm from getting stuck in a no-risk strategy for even $|\mathbf{D}|$. E.g., given $c = 0.5$ and $\epsilon = 0$, the elimination rate of a no-risk query $e(X_s, a_s) = \frac{1}{2}$ resulting always in $adj = 0$. The value of ϵ can be set to an arbitrary real number, e.g. $\epsilon := \frac{1}{4}$. If $c + c_{adj}$ is outside the user-defined cautiousness interval $[\underline{c}, \overline{c}]$, it is set to $\underline{c}$ if $c < \underline{c}$ and to $\overline{c}$ if $c > \overline{c}$. Positive c_{adj} is a penalty telling the algorithm to get more cautious, whereas negative c_{adj} is a bonus resulting in a braver behavior of the algorithm.

The RIO algorithm, described in Algorithm 2, starts with the computation of minimal diagnoses. GETDIAGNOSES function implements a combination of hitting-set (HS-Tree) [11] and QuickXPlain [6] algorithms as suggested in [12]. Using uniform cost search, the algorithm extends the set of leading diagnoses $\mathbf{D}$ with a maximum number of most probable minimal diagnoses such that $|\mathbf{D}| \leq n$.

Then the GETPROBABILITIES function calculates the fault probabilities $p(\mathcal{D}_i)$ for each diagnosis $\mathcal{D}_i$ of the set of leading diagnoses $\mathbf{D}$ according to [12]. In order to take into account all information gathered by querying an oracle so far the algorithm adjusts fault probabilities $p(\mathcal{D}_i)$ as follows: $p_{adj}(\mathcal{D}_i) = (1/2)^z\, p(\mathcal{D}_i)$, where z is the number of precedent queries X_k for which $\mathcal{D}_i \in \mathbf{D}_k^{\emptyset}$. Afterwards the probabilities $p_{adj}(\mathcal{D}_i)$ are normalized. Note that z can be computed from P and N which comprise all query answers. This way of updating probabilities is exactly in compliance with the Bayes Formula [12]. Based on the set of leading diagnoses $\mathbf{D}$, GENERATEQUERIES generates all queries according to Algorithm 1. GETMINSCOREQUERY determines the best query $X_{sc} \in \mathbf{X}$ according to sc_{ent}. That is, $X_{sc} = \arg\min_{X_k \in \mathbf{X}}(sc_{ent}(X_k))$. If X_{sc} is a non-high-risk query, i.e. $c \leq caut(X_{sc})$ (determined by GETQUERYCAUTIOUSNESS), X_{sc} is selected. In this case, X_{sc} is the query with maximum information gain among all queries $\mathbf{X}$ and additionally guarantees the required elimination rate specified by c.

Otherwise, GETALTERNATIVEQUERY selects the query $X_{alt} \in \mathbf{X}$ $(X_{alt} \neq X_{sc})$ which has minimal score sc_{ent} among all least cautious non-high-risk queries L_c. That is, $X_{alt} = \arg\min_{X_k \in L_c}(sc_{ent}(X_k))$ where $L_c = \{X_r \in \mathbf{X} \setminus HR_c(\mathbf{X}) \mid \forall X_t \in \mathbf{X} \setminus HR_c(\mathbf{X}) : caut(X_r) \leq caut(X_t)\}$. If there is no such query $X_{alt} \in \mathbf{X}$, then X_{sc} is selected. Given the positive answer of the oracle, the selected query $X_s \in \{X_{sc}, X_{alt}\}$ is added to the set of positive test cases P or, otherwise, to the set of negative test cases N. In the last step of the main loop the algorithm updates the cautiousness value c (function UPDATECAUTIOUSNESS) as described above.

Before the next query selection iteration starts, a stop condition test is performed. The algorithm evaluates whether the most probable diagnosis is at least $\sigma\%$ more likely than the second most probable diagnosis (ABOVETHRESHOLD) or none of the leading diagnoses has been eliminated by the previous query, i.e.GETELIMINATIONRATE returns zero for X_s. In case that one of the stop conditions is fulfilled, the presently most likely diagnosis is returned (MOSTPROBABLEDIAG).

4 Evaluation

The main points we want to show in this evaluation are: On the one hand, independently of the specified meta information, RIO exhibits superior average behavior compared to entropy-based method and split-in-half w.r.t. the amount of user interaction required. On the other hand, we want to demonstrate that RIO scales well and that the reaction time measured is well suited for an interactive debugging approach.

As data source for the evaluation we used problematic real-world ontologies produced by ontology matching systems.[4] This has the following reasons: (1) Matching results often cause inconsistency and/or incoherency of ontologies. (2) The (fault) structure of different ontologies obtained through matching generally varies due to different authors and matching systems involved in the genesis of these ontologies. (3) For the same reasons, it is hard to estimate the quality of fault probabilities, i.e. it is unclear which of the existing query selection strategies to chose for best performance. (4) Available reference mappings can be used as correct solutions of the debugging procedure.

Matching of two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ is understood as detection of correspondences between matchable elements of these ontologies. An ontology matching operation determines an *alignment* M_{ij}, which is a set of correspondences, i.e. tuples of the form $\langle x_i, x_j, r, v\rangle$, where $x_i \in Q(\mathcal{O}_i)$, $x_j \in Q(\mathcal{O}_j)$, $Q(\mathcal{O})$ is the set of matchable elements of an ontology $\mathcal{O}$, r is a semantic relation and $v \in [0, 1]$ is a confidence value. We call $\mathcal{O}_{iMj} := \mathcal{O}_i \cup M_{ij} \cup \mathcal{O}_j$ the *aligned ontology* for $\mathcal{O}_i$ and $\mathcal{O}_j$. In our approach the elements of $Q(\mathcal{O})$ are restricted to atomic concepts and roles and $r \in \{\sqsubseteq, \sqsupseteq, \equiv\}$ under the natural alignment semantics [8]

Example (continued): Imagine that our example ontology $\mathcal{O}$ evolved from matching two standalone ontologies $\mathcal{O}_1 := \{ax_1, ax_2\}$ and $\mathcal{O}_2 := \{ax_3, ax_4\}$ resulting in the alignment $M_{12} = \{ax_5, ax_6\}$. If we recall the set of diagnoses for $\mathcal{O}$ consisting of all single axioms in $\mathcal{O}$, we realize that the fault we are trying to find may be located either in $\mathcal{O}_1$ or in $\mathcal{O}_2$ or in M_{12}. Existing approaches to alignment debugging usually consider only the produced alignment as problem source. Our approach, on the contrary, is designed to cope with the most general setting: Any subset $S \subseteq \mathcal{O}_{1M2}$ of axioms of the aligned ontology can be analyzed for faults whereas $\mathcal{O}_{1M2} \setminus S$ can be added to the background axioms $\mathcal{B}$, if known to be correct. In this way, the search space for

[4] Thanks to Christian Meilicke for the supply of the test cases used in the evaluation.

Algorithm 2: Risk Optimization Algorithm (RIO)

Input: diagnosis problem instance $\langle \mathcal{O}, \mathcal{B}, P, N\rangle$, fault probabilities of diagnoses DP, cautiousness $C = (c, \underline{c}, \overline{c})$, number of leading diagnoses n to be considered, acceptance threshold σ
Output: a diagnosis $\mathcal{D}$

$P \leftarrow \emptyset$; $N \leftarrow \emptyset$; $\mathbf{D} \leftarrow \emptyset$;
repeat
  $\mathbf{D} \leftarrow$ `getDiagnoses`$(\mathbf{D}, n, \mathcal{O}, \mathcal{B}, P, N)$;
  $DP \leftarrow$ `getProbabilities`$(DP, \mathbf{D}, P, N)$;
  $\mathbf{X} \leftarrow$ `generateQueries`$(\mathcal{O}, \mathcal{B}, P, \mathbf{D})$;
  $X_s \leftarrow$ `getMinScoreQuery`$(DP, \mathbf{X})$;
  if `getQueryCautiousness`$(X_s, \mathbf{D}) < c$ **then** $X_s \leftarrow$ `getAlternativeQuery`$(c, \mathbf{X}, DP, \mathbf{D})$;
  if `getAnswer`$(X_s) =$ *yes* **then** $P \leftarrow P \cup \{X_s\}$;
  else $N \leftarrow N \cup \{X_s\}$;
  $c \leftarrow$ `updateCautiousness`$(\mathbf{D}, P, N, X_s, c, \underline{c}, \overline{c})$;
until (`aboveThreshold`$(DP, \sigma) \vee$ `eliminationRate`$(X_s) = 0$);
return `mostProbableDiag`$(\mathbf{D}, DP)$;

diagnoses can be restricted elegantly depending on the prior knowledge about $\mathcal{D}_t$, which can greatly reduce the complexity of the underlying diagnosis problem.

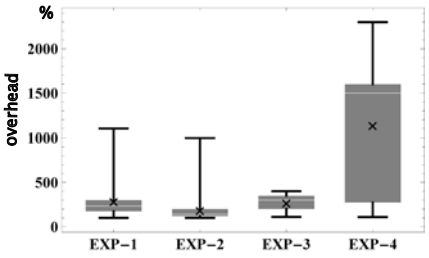
In [13] it was shown that existing debugging approaches suffer from serious problems w.r.t. both scalability and correctness of results when tested on a dataset of incoherent aligned OWL ontologies. Since RIO is an *interactive* ontology debugging approach able to query and incorporate additional information into its computations, it can cope with cases unsolved in [13]. In order to provide evidence for this and to show the feasibility of RIO – simultaneously to the main goals of this evaluation – we decided to use a superset of the dataset[5] used in [13] for our tests. Each incoherent aligned ontology $\mathcal{O}_{iMj}$ in the dataset is the result of applying one of the ontology matching systems COMA++, Falcon-AO, HMatch or OWL-CTXmatch to a set of six ontologies $Ont = \{\text{CRS}, \text{PCS}, \text{CMT}, \text{CONFTOOL}, \text{SIGKDD}, \text{EKAW}\}$ in the domain of conference organization. For a given pair of ontologies $\mathcal{O}_i \neq \mathcal{O}_j \in Ont$, each system produced an alignment M_{ij}. On the basis of a manually produced reference alignment $\mathcal{R}_{ij} \subseteq M_{ij}$ for ontologies $\mathcal{O}_i, \mathcal{O}_j$ (cf. [9]), we were able to fix a target diagnosis $\mathcal{D}_t$ for each incoherent $\mathcal{O}_{iMj}$. In cases where $\mathcal{R}_{ij}$ suggested a non-minimal diagnosis, we defined $\mathcal{D}_t$ as the minimum cardinality diagnosis which was a subset of $M_{ij} \setminus \mathcal{R}_{ij}$. In one single case, $\mathcal{R}_{ij}$ proved to be incoherent because an obviously valid correspondence $Reviewer_1 \equiv reviewer_2$ turned out to be incorrect. We re-evaluated this ontology and specified a coherent $\mathcal{R}_{ij}$. Yet this makes evident that, in general, people are not capable of analyzing alignments without adequate tool support.

In our experiments we set the prior fault probabilities as follows: $p(ax_k) := 0.001$ for $ax_k \in \mathcal{O}_i \cup \mathcal{O}_j$ and $p(ax_m) := 1 - v_m$ for $ax_m \in M_{ij}$, where v_m is the confidence of the correspondence underlying ax_m. Note that this choice results in a significant bias towards diagnoses which include axioms from M_{ij}. Based on these settings, in the first experiment (EXP-1), we simulated an interactive debugging session employing split-in-half (SPL), entropy (ENT) and RIO algorithms, respectively, for each ontology $\mathcal{O}_{iMj}$. Throughout all experiments, we performed module extraction before each test run, which is a standard preprocessing method for ontology debugging approaches. All tests were executed on a Core-i7 (3930K) 3.2Ghz, 32GB RAM and with Ubuntu Server 11.04 and Java 6 installed. The user-chosen parameters were set as follows: $|\mathbf{D}| := 9$ (proved to be a good trade-off between computational complexity for query generation and approximation of all minimal diagnoses), $\sigma := 85\%$, $c := 0.25$ and $[\underline{c}, \overline{c}] := [c_{\min}, c_{\max}] = [0, \frac{4}{9}]$. For the tests we considered the most general setting, i.e. $\mathcal{D}_t \subset \mathcal{O}_{iMj}$. So, we did not restrict the search for $\mathcal{D}_t$ to M_{ij} only, simulating the case where the user has no idea whether any of the input ontologies $\mathcal{O}_i, \mathcal{O}_j$ or the alignment M_{ij} or a combination thereof is faulty. In each test run we measured the number of required queries until $\mathcal{D}_t$ was identified. In order to simulate the case where the fault includes at least one axiom $ax \in \mathcal{O}_{iMj} \setminus M_{ij}$, we implemented a second test session with altered $\mathcal{D}_t$. In this experiment (EXP-2), we precalculated a maximum of 30 most probable minimal diagnoses, and from these we selected the diagnosis with the highest number of axioms $ax_k \in \mathcal{O}_{iMj} \setminus M_{ij}$ as $\mathcal{D}_t$ in order to simulate more unsuitable meta information. All the other settings were left unchanged. The queries generated in the tests were answered by an automatic oracle by means of the target ontology $\mathcal{O}_{iMj} \setminus \mathcal{D}_t$. The average metrics for the set of aligned ontologies $\mathcal{O}_{iMj}$ per matching system were as follows: $312 \leq |\mathcal{O}_{iMj}| \leq 377$ and $19.1 \leq |M_{ij}| \leq 28.4$.

[5] http://code.google.com/p/rmbd/downloads

	EXP-1	EXP-2	EXP-3	EXP-4
$q_{\text{SPL}} < q_{\text{ENT}}$	11%	37%	0%	29%
$q_{\text{ENT}} < q_{\text{SPL}}$	82%	56%	100%	71%
$q_{\text{SPL}} = q_{\text{ENT}}$	7%	7%	0%	0%
$q_{\text{RIO}} < \min$	4%	26%	29%	71%
$q_{\text{RIO}} \leq \min$	74%	74%	100%	100%

(a)



(b)

Fig. 1. (a) Percentage rates indicating which strategy performed best/better w.r.t. the required user interaction, i.e. number of queries. EXP-1 and EXP-2 involved 27, EXP-3 and EXP-4 seven debugging sessions each. q_{str} denotes the number of queries needed by strategy str and $\min$ is an abbreviation for $\min(q_{\text{SPL}}, q_{\text{ENT}})$. **(b)** Box-Whisker Plots presenting the distribution of overhead $(q_w - q_b)/q_b * 100$ (in %) per debugging session of the worse strategy $q_w := \max(q_{\text{SPL}}, q_{\text{ENT}})$ compared to the better strategy $q_b := \min(q_{\text{SPL}}, q_{\text{ENT}})$. Mean values are depicted by a cross.

In order to analyze the scalability of RIO, we used the set of ontologies from the ANATOMY track in the Ontology Alignment Evaluation Initiative[6] (OAEI) 2011.5, which comprises two input ontologies $\mathcal{O}_1$ (Human, 11545 axioms) and $\mathcal{O}_2$ (Mouse, 4838 axioms). The size of the alignments generated by 12 different matching systems was between 1147 and 1461 correspondences. Note that the aligned ontologies output by five matching systems, i.e. CODI, CSA, MaasMtch, MapEVO and Aroma, could not be analyzed in the experiments. This was due to a consistent output produced by CODI and the problem that the reasoner was not able to find a model within acceptable time (2 hours) in the case of CSA, MaasMtch, MapEVO and Aroma. Similar reasoning problems were also reported in [2]. Given the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the output M_{12} of a matching system, and the correct reference alignment $\mathcal{R}_{12}$, we first fixed $\mathcal{D}_t$ as follows: Both ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ as well as the correctly extracted alignments $M_{12} \cap \mathcal{R}_{12}$ were placed in the background knowledge $\mathcal{B}$. The incorrect correspondences $M_{12} \setminus \mathcal{R}_{12}$ were analyzed by the debugger. In this way, we identified a set of diagnoses, where each diagnosis is a subset of $M_{12} \setminus \mathcal{R}_{12}$. From this set of diagnoses, we randomly selected one diagnosis as $\mathcal{D}_t$. Then we started the actual experiments: In EXP-3,[7] in order to simulate reasonable prior fault probabilities, a debugging session with parameter settings as in EXP-1 was executed. In EXP-4, we altered the settings in that we specified $p(ax_k) := 0.01$ for $ax_k \in \mathcal{O}_i \cup \mathcal{O}_j$ and $p(ax_m) := 0.001$ for $ax_m \in M_{ij}$, which caused the target diagnosis, that consisted solely of axioms in M_{ij}, to get assigned a relatively low prior fault probability.

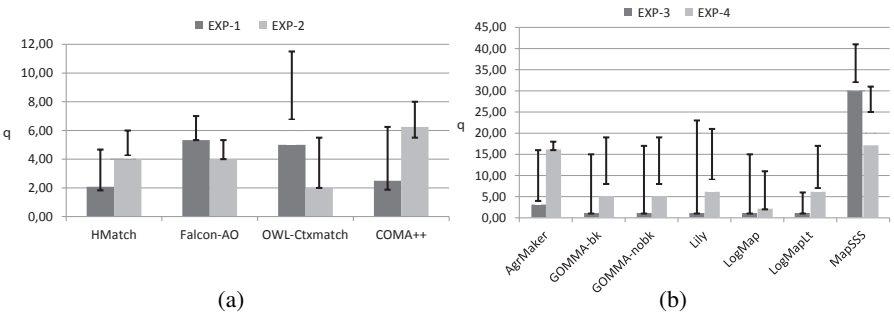
Results of both experimental sessions, ⟨EXP-1,EXP-2⟩ and ⟨EXP-3,EXP-4⟩, are summarized in Figure 2(a) and Figure 2(b), respectively. For the ontologies produced by each of the matching systems and for the different experimental scenarios, the figures show the (average) number of queries asked by RIO and the (average) differences to the number of queries needed by the per-session better and worse strategy of SPL and ENT, respectively. The results illustrate clearly that the average performance achieved by RIO was always substantially closer to the better than to the worse strategy. In both EXP-1 and EXP-2, throughout 74% of 27 debugging sessions, RIO worked as efficiently as the best strategy (Figure 1(a)). In more than 25% of the cases in EXP-2, RIO even

[6] http://oaei.ontologymatching.org

[7] For all details w.r.t. ⟨EXP-3,EXP-4⟩, see http://code.google.com/p/rmbd/wiki/ OntologyAlignmentAnatomy.

outperformed both other strategies; in these cases, RIO could save more than 20% of user interaction on average compared to the best other strategy. In one scenario involving OWL-CTXmatch in EXP-1, it took ENT 31 and SPL 13 queries to finish, whereas RIO required only 6 queries, which amounts to an improvement of more than 80% and 53%, respectively. In ⟨EXP-3,EXP-4⟩, the savings achieved by RIO were even more substantial. RIO manifested superior behavior to both other strategies in 29% and 71% of cases, respectively. Not less remarkable, in 100% of the tests in EXP-3 and EXP-4, RIO was at least as efficient as the best other strategy. Table 1, which provides the average number of queries per strategy, demonstrates that, overall, RIO is the best choice in all experiments. Consequently, RIO is suitable for both good meta information as in EXP-1 and EXP-3, where $\mathcal{D}_t$ has high probability, and poor meta information as in EXP-2 and EXP-4, where $\mathcal{D}_t$ is a-priori less likely. Additionally, Table 1 illustrates the (average) *overall* debugging time assuming that queries are answered instantaneously and the reaction time, i.e. the average time between two successive queries. Also w.r.t. these aspects, RIO manifested good performance. Since the times consumed by either of the strategies in ⟨EXP-1,EXP-2⟩ are almost negligible, consider the more meaningful results obtained in ⟨EXP-3,EXP-4⟩. While the best reaction time in both experiments was achieved by SPL, we can clearly see that SPL was significantly inferior to both ENT and RIO concerning the user interaction required and the overall time. RIO revealed the best debugging time in EXP-4, and needed only 2.2% more time than the best strategy (ENT) in EXP-3. However, if we assume the user being capable of reading and answering a query in, e.g., half a minute on average, which is already quite fast, then the overall time savings of RIO compared to ENT in EXP-3 would already account for 5%. Doing the same thought experiment for EXP-4, using RIO instead of ENT and SPL would save 25% and 50% of debugging time on average, respectively. All in all, the measured times confirm that RIO is well suited as *interactive* debugging method.

For SPL and ENT strategies, the difference w.r.t. the number of queries per test run between the better and the worse strategy was absolutely significant, with a maximum of 2300% in EXP-4 and averages of 190% to 1145% throughout all four experiments, measured on the basis of the better strategy (Figure 1(b)). Moreover, results show that the different quality of the prior fault probabilities in {EXP-1,EXP-3} compared to {EXP-2,EXP-4} clearly affected the performance of the ENT and SPL strategies (see first two rows in Figure 1(a)). This perfectly motivates the application of RIO.



(a) (b)

Fig. 2. The bars show the avg. number of queries (q) needed by RIO, grouped by matching tools. The distance from the bar to the lower (upper) end of the whisker indicates the avg. difference of RIO to the queries needed by the per-session better (worse) strategy of SPL and ENT, respectively.

	EXP-1			EXP-2			EXP-3			EXP-4		
	debug	react	q	debug	react	q	debug	react	q	debug	react	q
ENT	1860	262	3.67	1423	204	5.26	**60928**	12367	5.86	74463	5629	11.86
SPL	**1427**	**159**	5.70	**1237**	**148**	5.44	104910	**4786**	19.43	98647	**4781**	18.29
RIO	1592	286	**3.00**	1749	245	**4.37**	62289	12825	**5.43**	**66895**	8327	**8.14**

Table 1. Average time (ms) for the entire debugging session (debug), average time (ms) between two successive queries (react), and average number of queries (q) required by each strategy.

5 Conclusion

We have shown problems of state-of-the-art interactive ontology debugging strategies w.r.t. the usage of unreliable meta information. To tackle this issue, we proposed a learning strategy which combines the benefits of existing approaches, i.e. high potential and low risk. Depending on the performance of the diagnosis discrimination actions, the trust in the a-priori information is adapted. Tested under various conditions, our algorithm revealed an average performance superior to two common approaches in the field w.r.t. required user interaction. In our evaluation we showed the utility of our approach in the important area of ontology matching, its scalability and adequate reaction time allowing for continuous interactivity.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, Applications. Cambridge Press (2003)
2. Ferrara, A., Hage, W.R.V., Hollink, L., Nikolov, A., Shvaiko, P.: Final results of the Ontology Alignment Evaluation Initiative 2011. In: Evaluation (2011)
3. Friedrich, G., Shchekotykhin, K.: A General Diagnosis Method for Ontologies. In: Gil, Y., Motta, E., Benjamins, R., Musen, M. (eds.) The Semantic Web - ISWC 2005, 4th International Semantic Web Conference. pp. 232–246. Springer (2005)
4. Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. Proc of the 7th International Semantic Web Conference ISWC 2008 5318, 323–338 (2008)
5. Jiménez-Ruiz, E., Grau, B.C.: Logmap: Logic-based and scalable ontology matching. In: The Semantic Web - ISWC 2011. pp. 273–288. Springer (2011)
6. Junker, U.: QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In: Proceedings of the 19th National Conference on AI, 16th Conference on Innovative Applications of AI. vol. 3, pp. 167–172. AAAI Press / The MIT Press (2004)
7. de Kleer, J., Williams, B.C.: Diagnosing multiple faults. Artif. Intell. 32(1), 97–130 (1987)
8. Meilicke, C., Stuckenschmidt, H.: An Efficient Method for Computing Alignment Diagnoses. In: Proceedings of the 3rd International Conference on Web Reasoning and Rule Systems. pp. 182–196. Springer-Verlag (2009)
9. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Reasoning Support for Mapping Revision. Journal of Logic and Computation 19(5), 807–829 (2008)
10. Noessner, J., Niepert, M., Meilicke, C., Stuckenschmidt, H.: Leveraging Terminological Structure for Object Reconciliation. The Semantic Web: Research and Applications pp. 334–348 (2010)
11. Reiter, R.: A Theory of Diagnosis from First Principles. Artif. Intell. 32(1), 57–95 (1987)
12. Shchekotykhin, K., Friedrich, G., Fleiss, P., Rodler, P.: Interactive ontology debugging : two query strategies for efficient fault localization. Web Semantics: Science, Services and Agents on the World Wide Web 12-13, 88–103 (2012)
13. Stuckenschmidt, H.: Debugging OWL Ontologies - A Reality Check. In: Proceedings of the 6th International Workshop on Evaluation of Ontology-based Tools and the Semantic Web Service Challenge (EON). pp. 1–12. Tenerife, Spain (2008)