RIO: Minimizing User Interaction in Ontology Debugging

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 some meta information in terms of fault probabilities, which can speed up the debugging procedure in the good case, but can also have negative impact on the performance in the bad case. The problem is that assessment of the meta information is only possible a-posteriori. Consequently, 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 tool which pursues 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 a corpus of incoherent real-world ontologies from the field of ontology matching, we show that the proposed risk-aware query strategy outperforms both meta information based approaches and no-risk strategies on average in terms of 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. Usually, such tools [10,4,2,3] use model-based diagnosis [9] 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.

In [11] this issue is tackled by letting the user take action during the debugging session by answering queries about entailments and non-entailments of the desired ontology. These answers pose constraints to the validity of diagnoses and thus help to sort out

W. Faber and D. Lembo (Eds.): RR 2013, LNCS 7994, pp. 153–167, 2013.

incompliant diagnoses step-by-step. In addition, a Bayesian approach is used to continuously readjust the fault probabilities by means of the additional information given by the user. The user effort in this interactive debugging procedure is strongly affected by the quality of the initially provided meta information, i.e. prior knowledge about fault probabilities of a user w.r.t. particular logical operators. To get this under control, the selection of queries shown to the user can be varied correspondingly. To this end, two essential paradigms for choosing the next "best" query have been proposed, split-in-half and entropy-based.

In order to opt for the optimal strategy, however, the quality of the meta information, i.e. good or bad (which means high or low probability of the correct solution), must be known in advance. This would, however, implicate the pre-knowledge of the initially unknown solution. Entropy-based methods can make optimal profit from exploiting properly adjusted initial fault probabilities (high potential), whereas they can completely fail in the case of weak prior information (high risk). The split-in-half technique, on the other hand, manifests constant behavior independently of the probabilities given (no risk), but lacks the ability to leverage appropriate fault information (no potential). This matter of fact is witnessed by the evaluation we conducted, which shows that an unsuitable combination of meta information and query selection strategy can result in a substantial increase of more than 2000% w.r.t. number of queries to a user. So, there is a need to either (1) guarantee a sufficiently suited choice of prior fault information, or (2) to manage the "risk" of unsuitable method selection. The task of (1) might not be a severe problem in a debugging scenario involving a faulty ontology developed by a single expert, since the meta information might be extracted from the logs of previous sessions, if available, or specified by the expert based on their experience w.r.t. own faults. However, realization of task (1) is a major issue in scenarios involving automatized systems producing (parts of) ontologies, e.g. ontology alignment and ontology learning, or numerous users collaborating in modeling an ontology, where the choice of reasonable meta information is rather unclear. Therefore, we focus on accomplishing task (2).

The contribution of this paper is a new RIsk Optimization reinforcement learning method (RIO), which allows to minimize user interaction throughout a debugging session on average compared to existing strategies, for any quality of meta information (high potential at low risk). By virtue of its learning capability, our approach is optimally suited for debugging ontologies where only vague or no meta information is available. A learning parameter is constantly adapted based on the information gathered so far. 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.

Experiments on two datasets of faulty real-world ontologies show the feasibility, efficiency and scalability of RIO. The evaluation will indicate that, on average, RIO is the best choice of strategy for both good and bad meta information with savings as to 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. Related work is discussed in Section 5 Section 6 concludes.

2 Basic Concepts and Motivation

First we provide an informal introduction to ontology debugging, particularly addressing readers unfamiliar with the topic. Later we introduce precise formalizations. We assume the reader to be familiar with description logics [1].

Ontology debugging deals with the following problem: Given is an ontology $\mathcal{O}$ which does not meet postulated requirements R, e.g. $R = \{\text{coherency}, \text{consistency}\}$. $\mathcal{O}$ is a set of axioms formulated in some monotonic knowledge representation language, e.g. OWL DL. The task is to find a subset of axioms in $\mathcal{O}$, called diagnosis, that needs to be altered or eliminated from the ontology in order to meet the given requirements. The presented approach to ontology debugging does not rely upon a specific knowledge representation formalism, it solely presumes that it is logic-based and monotonic. Additionally, the existence of sound and complete procedures for deciding logical consistency and for calculating logical entailments is assumed. These procedures are used as a black box. For OWL DL, e.g., both functionalities are provided by a standard DL-reasoner.

A diagnosis is a hypothesis about the state of each axiom in $\mathcal{O}$ of being either correct or faulty. Generally, there are many diagnoses for one and the same faulty ontology $\mathcal{O}$. The problem is then to figure out the single diagnosis, called target diagnosis $\mathcal{D}^*$, that complies with the knowledge to be modeled by the intended ontology. In interactive ontology debugging we assume a user, e.g. the author of the faulty ontology or a domain expert, interacting with an ontology debugging system by answering queries about entailments of the desired ontology, called the target ontology $\mathcal{O}^*$. The target ontology can be understood as $\mathcal{O}$ minus the axioms of $\mathcal{D}^*$ plus a set of axioms needed to preserve the desired entailments, called positive test cases. Note that the user is not expected to know $\mathcal{O}^*$ explicitly (in which case there would be no need to consult an ontology debugger), but implicitly in that they are able to answer queries about $\mathcal{O}^*$.

A query is a set of axioms and the user is asked whether the conjunction of these axioms is entailed by $\mathcal{O}^*$. Every positively (negatively) answered query constitutes a positive (negative) test case fulfilled by $\mathcal{O}^*$. The set of positive (entailed) and negative (non-entailed) test cases is denoted by P and N, respectively. So, P and N are sets of sets of axioms, which can be, but do not need to be, initially empty. Test cases can be seen as constraints $\mathcal{O}^*$ must satisfy and are therefore used to gradually reduce the search space for valid diagnoses. Roughly, the overall procedure consists of (1) computing a predefined number of diagnoses, (2) gathering additional information by querying the user, (3) incorporating this information to prune the search space for diagnoses, and so forth, until a stopping criterion is fulfilled, e.g. one diagnosis $\mathcal{D}^*$ has overwhelming probability.

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 that are known to be correct. $\mathcal{B}$ is then incorporated in the calculations throughout the ontology debugging procedure, but no axiom in $\mathcal{B}$ may take part in a diagnosis. For example, in case the user knows that a subset of axioms in $\mathcal{O}$ is definitely sound, all axioms in this subset are added to $\mathcal{B}$ before initiating the debugging session. The advantage of this over simply not considering the axioms in $\mathcal{B}$ at all is, that the semantics of axioms in $\mathcal{B}$ is not lost and can be exploited, e.g., in query generation. $\mathcal{B}$ and $\mathcal{O} \setminus \mathcal{B}$ partition the original ontology into a set of correct and possibly incorrect axioms, respectively. In the debugging

session, only $\mathcal{O} := \mathcal{O} \setminus \mathcal{B}$ is used to search for diagnoses. This can reduce the search space for diagnoses substantially. Another application of background knowledge could be the reuse of an existing ontology to support successful debugging. For example, when formulating an ontology about medical terms, a thoroughly curated reference ontology $\mathcal{B}$ could be leveraged to find own formulations contradicting the correct ones in $\mathcal{B}$, which would not be found without integration of $\mathcal{B}$ into the debugging procedure.

More formally, ontology debugging can be defined in terms of a diagnosis problem instance, for which we search for solutions, i.e. diagnoses, that enable to formulate the target ontology:

Definition 1 (Diagnosis Problem Instance, Target Ontology). *Let $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ be an ontology with terminological axioms $\mathcal{T}$ and assertional axioms $\mathcal{A}$, $\mathcal{B}$ a set of axioms which are assumed to be correct (background knowledge), R a set of requirements to $\mathcal{O}$, P and N respectively a set of positive and negative test cases, where each test case $p \in P$ and $n \in N$ is a set of axioms. Then we call the tuple $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$ a* diagnosis problem instance (DPI). *An ontology $\mathcal{O}^*$ is called* target ontology *w.r.t. $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$ iff all the following conditions hold:*

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

Definition 2 (Diagnosis). *We call $\mathcal{D} \subseteq \mathcal{O}$ a* diagnosis *w.r.t. a DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$ iff $(\mathcal{O} \setminus \mathcal{D}) \cup (\bigcup_{p \in P} p)$ is a target ontology w.r.t. $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$. A diagnosis $\mathcal{D}$ w.r.t. a DPI is* minimal *iff there is no $\mathcal{D}' \subset \mathcal{D}$ such that $\mathcal{D}'$ is a diagnosis w.r.t. this DPI. The set of minimal diagnoses w.r.t. a DPI is denoted by* $\mathbf{MD}$.

Note that 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.

Example: Consider $\mathcal{O} := \mathcal{T} \cup \mathcal{A}$ with terminological axioms $\mathcal{T} := \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}_{12}$:

$\mathcal{O}_1$	$ax_1 : PhD \sqsubseteq Researcher$
	$ax_2 : Researcher \sqsubseteq DeptEmployee$
$\mathcal{O}_2$	$ax_3 : PhDStudent \sqsubseteq Student$
	$ax_4 : Student \sqsubseteq \neg DeptMember$
$\mathcal{M}_{12}$	$ax_5 : PhDStudent \sqsubseteq PhD$
	$ax_6 : DeptEmployee \sqsubseteq DeptMember$

and an assertional axiom $\mathcal{A} = \{PhDStudent(s)\}$, where $\mathcal{M}_{12}$ is an automatically generated set of axioms serving as semantic links between $\mathcal{O}_1$ and $\mathcal{O}_2$. The given ontology $\mathcal{O}$ is inconsistent since it describes s as both a $DeptMember$ 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 that no test cases are initially specified, i.e. the sets P and N are empty. For the resulting DPI $\langle \mathcal{T}, \mathcal{A}, \emptyset, \emptyset \rangle_{\{\text{coherence}\}}$ the set of minimal diagnoses $\mathbf{MD} = \{\mathcal{D}_1 : [ax_1], \mathcal{D}_2 : [ax_2], \mathcal{D}_3 : [ax_3], \mathcal{D}_4 : [ax_4], \mathcal{D}_5 : [ax_5], \mathcal{D}_6 : [ax_6]\}$. $\mathbf{MD}$ can be computed by a diagnosis algorithm such as the one presented in [2].

With six minimal diagnoses for only six ontology axioms, this example already gives an idea that in many cases $|\mathbf{MD}|$ can get very large. Note that generally the computation of *all* minimal diagnoses w.r.t. a given DPI is not feasible within reasonable time due to the complexity of the underlying algorithms. Therefore, in practice, especially in an interactive scenario where reaction time is essential, a set of *leading diagnoses* $\mathbf{D} \subseteq \mathbf{MD}$ is considered as a representative for $\mathbf{MD}$.[1] Concerning the optimal number of leading diagnoses, a trade-off between representativeness and complexity of associated computations w.r.t. $\mathbf{D}$ needs to be found.

Without any prior knowledge in terms of diagnosis fault probabilities or specified test cases, each diagnosis in $\mathbf{D}$ is equally likely to be the target diagnosis $\mathcal{D}^*$. In other words, for each $\mathcal{D} \in \mathbf{D}$ w.r.t. the DPI $\langle \mathcal{T}, \mathcal{A}, \emptyset, \emptyset \rangle_{\{\text{coherence}\}}$, the ontology $(\mathcal{O} \setminus \mathcal{D}) \cup (\bigcup_{p \in P} p)$ meets all the conditions defining a target ontology. However, besides postulating coherence the user might want the target ontology to entail that s is a student as well as a researcher, i.e. $\mathcal{O}^* \models t_1$ where $t_1 := \{Researcher(s), Student(s)\}$. Formulating t_1 as a positive test case yields the DPI $\langle \mathcal{T}, \mathcal{A}, \{t_1\}, \emptyset \rangle_{\{\text{coherence}\}}$, for which only diagnoses $\mathcal{D}_2, \mathcal{D}_4, \mathcal{D}_6 \in \mathbf{D}$ are valid and enable to formulate a corresponding $\mathcal{O}^*$. All other diagnoses in $\mathbf{D}$ are ruled out by the fact that $t_1 \in P$, which means they have a probability of zero of being the target diagnosis. If $t_1 \in N$, in contrast, this would imply that $\mathcal{D}_2, \mathcal{D}_4, \mathcal{D}_6$ had to be rejected.

So, it depends on the test cases specified by a user which diagnosis will finally be identified as target diagnosis. Also, the order in which test cases are specified, is crucial. For instance, consider the test cases $t_1 := \{PhD(s)\}$ and $t_2 := \{Student(s)\}$. If $t_1 \in P$ is specified before $t_2 \in N$, then $t_1 \in P$ is redundant, since the only diagnosis agreeing with $t_2 \in N$ is $\mathcal{D}_3$ which preserves also the entailment t_1 in the resulting target ontology $\mathcal{O}^* = (\mathcal{O} \setminus \mathcal{D}_3) \cup \emptyset$ without explicating it as a positive test case.

Since it is by no means trivial to get the right – in the sense of most informative – test cases formulated in the proper order such that the number of test cases necessary to detect the target diagnosis is minimized, interactive debugging systems offer the functionality to automatize selection of test cases. The benefit is that the user can just concentrate on "answering" the provided test cases which means assigning them to either P or N. We call such automatically generated test cases queries. The theoretical foundation for the application of queries is the fact that $\mathcal{O} \setminus \mathcal{D}_i$ and $\mathcal{O} \setminus \mathcal{D}_j$ for $\mathcal{D}_i \neq \mathcal{D}_j \in \mathbf{D}$ entail different sets of axioms.

Definition 3 (Query, Partition). *Let* $\mathbf{D}$ *be a set of minimal diagnoses w.r.t. a DPI* $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$ *and* $\mathcal{O}_i^* := (\mathcal{O} \setminus \mathcal{D}_i) \cup \mathcal{B} \cup (\bigcup_{p \in P} p)$ *for* $\mathcal{D}_i \in \mathbf{D}$*. Then a set of axioms* $X_j \neq \emptyset$ *is called a* query *w.r.t.* $\mathbf{D}$ *iff* $\mathbf{D}_j^P := \{\mathcal{D}_i \in \mathbf{D} \mid \mathcal{O}_i^* \models X_j\} \neq \emptyset$ *and* $\mathbf{D}_j^N := \{\mathcal{D}_i \in \mathbf{D} \mid \exists r \in R : \mathcal{O}_i^* \cup X_j \text{ violates } r\} \neq \emptyset$*. The (unique)* partition *of a query* X_j *is denoted by* $\langle \mathbf{D}_j^P, \mathbf{D}_j^N, \mathbf{D}_j^\emptyset \rangle$ *where* $\mathbf{D}_j^\emptyset = \mathbf{D} \setminus (\mathbf{D}_j^P \cup \mathbf{D}_j^N)$*.* $\mathbf{X_D}$ *terms a set of queries and associated partitions w.r.t.* $\mathbf{D}$ *in which one and the same partition of* $\mathbf{D}$ *occurs at most once and only if there is an associated query for this partition.*

[1] So, we will speak of $\mathbf{D}$ instead of $\mathbf{MD}$ throughout this work. Note that the restriction to a subset of $\mathbf{MD}$ does not necessarily have implications on the completeness of the associated ontology debugging algorithm. E.g., the algorithm can be iterative and recompute new diagnoses on demand and nevertheless guarantee completeness (as the algorithm presented in this work).

Algorithm 1: Generation of Queries and Partitions

Input: DPI $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$, set of corresponding diagnoses $\mathbf{D}$
Output: a set of queries and associated partitions $\mathbf{X_D}$

1 **foreach** $\mathbf{D}_k^P \subset \mathbf{D}$ **do**
2 $X_k \leftarrow$ `getEntailments`$(\mathcal{O}, \mathcal{B}, P, \mathbf{D}_k^P)$;
3 **if** $X_k \neq \emptyset$ **then**
4 **foreach** $\mathcal{D}_r \in \mathbf{D} \setminus \mathbf{D}_k^P$ **do**
5 **if** $\mathcal{O}_r^* \models X_k$ **then** $\mathbf{D}_k^P \leftarrow \mathbf{D}_k^P \cup \{\mathcal{D}_r\}$;
6 **else if** `reqViolated`$(\mathcal{O}_r^* \cup X_k)$ **then** $\mathbf{D}_k^N \leftarrow \mathbf{D}_k^N \cup \{\mathcal{D}_r\}$;
7 **else** $\mathbf{D}_k^\emptyset \leftarrow \mathbf{D}_k^\emptyset \cup \{\mathcal{D}_r\}$;
8 **if** $\neg$`includesPartition`$(\mathbf{X_D}, \left\langle \mathbf{D}_k^P, \mathbf{D}_k^N, \mathbf{D}_k^\emptyset \right\rangle)$ **then**
9 $\mathbf{X_D} \leftarrow \mathbf{X_D} \cup$ `minimizeQuery`$(\left\langle X_k, \left\langle \mathbf{D}_k^P, \mathbf{D}_k^N, \mathbf{D}_k^\emptyset \right\rangle \right\rangle)$
10 **return** $\mathbf{X_D}$;

Note that, in general, there can be n_q queries for a particular partition of $\mathbf{D}$ where n_q can be zero or some positive integer. We are interested in (1) only those partitions for each of which $n_q \geq 1$ and (2) only one query for each such partition. The set $\mathbf{X_D}$ includes elements such that (1) and (2) holds. $\mathbf{X_D}$ for a given set of minimal diagnoses $\mathbf{D}$ w.r.t. a DPI can be generated as shown in Algorithm 1. In each iteration, given a set of diagnoses $\mathbf{D}_k^P \subset \mathbf{D}$, common entailments[2] $X_k := \left\{ e \,|\, \forall \mathcal{D}_i \in \mathbf{D}_k^P : \mathcal{O}_i^* \models e \right\}$ are computed (GETENTAILMENTS) and used to classify the remaining diagnoses in $\mathbf{D} \setminus \mathbf{D}_k^P$ to obtain the partition $\langle \mathbf{D}_k^P, \mathbf{D}_k^N, \mathbf{D}_k^\emptyset \rangle$ associated with X_k. Then, if the partition $\langle \mathbf{D}_k^P, \mathbf{D}_k^N, \mathbf{D}_k^\emptyset \rangle$ does not already occur in $\mathbf{X_D}$ (INCLUDESPARTITION), the query X_k is minimized [11] (MINIMIZEQUERY) such that its partition is preserved, yielding a query $X_k' \subseteq X_k$ such that any $X_k'' \subset X_k'$ is not a query or has not the same partition. Finally, X_k' is added to $\mathbf{X_D}$ together with its partition $\langle \mathbf{D}_k^P, \mathbf{D}_k^N, \mathbf{D}_k^\emptyset \rangle$. Function REQVIOLATED($arg$) returns $true$ if arg violates some requirement $r \in R$.

Asking the user a query X_j means asking them ($\mathcal{O}^* \models X_j?$). Let the answering of queries by a user be modeled as function $u : \mathbf{X_D} \rightarrow \{t,f\}$. If $u_j := u(X_j) = t$, then $P \leftarrow P \cup \{X_j\}$ and $\mathbf{D} \leftarrow \mathbf{D} \setminus \mathbf{D}_j^N$. Otherwise, $N \leftarrow N \cup \{X_j\}$ and $\mathbf{D} \leftarrow \mathbf{D} \setminus \mathbf{D}_j^P$. Prospectively, according to Definition 2, only those diagnoses are considered in the set $\mathbf{D}$ that comply with the new DPI obtained by the addition of a test case. This allows us to formalize the problem we address in this work:

Problem Definition (Query Selection) ***Given*** $\mathbf{D}$ *w.r.t. a DPI* $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$*, a stopping criterion* $stop : \mathbf{D} \rightarrow \{t, f\}$ *and a user* u*,* ***find*** *a next query* $X_j \in \mathbf{X_D}$ *such that (1)* $(X_j, \ldots, X_q)$ *is a query sequence of minimal length and (2) there exists a* $\mathcal{D}^* \in \mathbf{D}$ *w.r.t.* $\langle \mathcal{O}, \mathcal{B}, P', N' \rangle_R$ *such that* $stop(\mathcal{D}^*) = t$*, where* $P' := P \cup \{X_i \,|\, X_i \in \{X_j, \ldots, X_q\}, u_i = t\}$ *and* $N' := N \cup \{X_i \,|\, X_i \in \{X_j, \ldots, X_q\}, u_i = f\}$.

Two strategies for selecting the "best" next query have been proposed [11]:
Split-in-half strategy (SPL), selects the query X_j which minimizes the following scoring function: $sc_{split}(X_j) := \left| |\mathbf{D}_j^P| - |\mathbf{D}_j^N| \right| + |\mathbf{D}_j^\emptyset|$. So, SPL prefers queries which eliminate half of the diagnoses independently of the query outcome. **Entropy-based**

[2] Note, when we speak of entailments throughout this work, we address (only) the *finite* set of entailments computed by the classification and realization services of a DL-reasoner.

strategy (ENT), uses information about prior probabilities p_t for the user to make a mistake when using a syntactical construct of type $t \in CT(\mathcal{L})$, where $CT(\mathcal{L})$ is the set of constructors available in the used knowledge representation language $\mathcal{L}$, e.g. $\{\forall, \exists, \sqsubseteq, \neg, \sqcup, \sqcap\} \subset CT(\text{OWL DL})$. These fault probabilities p_t are assumed to be independent and used to calculate fault probabilities of axioms ax_k as $p(ax_k) = 1 - \prod_{t \in CT}(1 - p_t)^{n(t)}$ where $n(t)$ is the number of occurrences of construct type t in ax_k. The probabilities of axioms can in turn be used to determine fault probabilities of diagnoses $\mathcal{D}_i \in \mathbf{D}$ as

$$p(\mathcal{D}_i) = \prod_{ax_r \in \mathcal{D}_i} p(ax_r) \prod_{ax_s \in \mathcal{O} \setminus \mathcal{D}_i} (1 - p(ax_s)). \quad (1)$$

ENT selects the query $X_j \in \mathbf{X_D}$ with highest expected information gain, i.e. that minimizes the following scoring function [11]:

$$sc_{ent}(X_j) = \sum_{a \in \{t,f\}} p(u_j = a) \log_2 p(u_j = a) + p(\mathbf{D}_j^{\emptyset}) + 1$$

where $p(u_j = t) = \sum_{\mathcal{D}_r \in \mathbf{D}_j^P} p(\mathcal{D}_r) + \frac{1}{2} p(\mathbf{D}_j^{\emptyset})$ and $p(\mathbf{D}_j^{\emptyset}) = \sum_{\mathcal{D}_r \in \mathbf{D}_j^{\emptyset}} p(\mathcal{D}_r)$. The answer $u_j = a$ is used to update probabilities $p(\mathcal{D}_k)$ for $\mathcal{D}_k \in \mathbf{D}$ according to the Bayesian formula, yielding $p(\mathcal{D}_k | u_j = a)$.

The result of the evaluation in [11] shows that ENT reveals better performance than SPL in most of the cases. However, SPL proved to be the best strategy in situations when misleading prior information is provided, i.e. the target diagnosis $\mathcal{D}^*$ has low probability. So, one can regard ENT as a high risk strategy with high potential to perform well, depending on the priorly unknown quality of the given fault information. SPL, in contrast, can be seen as a no-risk strategy without any potential to leverage good meta information. Therefore, selection of the proper combination of prior probabilities $\{p_t \mid t \in CT(\mathcal{L})\}$ and query selection strategy is crucial for successful diagnosis discrimination and minimization of user interaction.

Example (continued): To illustrate this, let a user who wants to debug our example ontology $\mathcal{O}$ set $p(ax_i) := 0.001$ for $ax_{i(i=1,\ldots,4)}$ and $p(ax_5) := 0.1, p(ax_6) := 0.15$, e.g. because the user doubts the correctness of ax_5, ax_6 while being quite sure that $ax_{i(i=1,\ldots,4)}$ are correct. Assume that $\mathcal{D}_2$ corresponds to the target diagnosis $\mathcal{D}^*$, i.e. the settings provided by the user are inept. Application of ENT starts with computation of prior fault probabilities of diagnoses $p(\mathcal{D}_1) = p(\mathcal{D}_2) = p(\mathcal{D}_3) = p(\mathcal{D}_4) = 0.003$, $p(\mathcal{D}_5) = 0.393$, $p(\mathcal{D}_6) = 0.591$ (Formula 1). Then $(\mathcal{O}^* \models X_1?)$ with $X_1 := \{DeptEmployee(s), Student(s)\}$, will be identified as the optimal query since it has the minimal score $sc_{ent}(X_1) = 0.02$ (see Table 1 for queries and partitions w.r.t. the example ontology). However, since the unfavorable answer $u_1 = f$ is given, this query eliminates only two of six diagnoses $\mathcal{D}_4$ and $\mathcal{D}_6$. The Bayesian probability update then yields $p(\mathcal{D}_2) = p(\mathcal{D}_3) = p(\mathcal{D}_4) = 0.01$ and $p(\mathcal{D}_5) = 0.97$. As next query X_2 with $sc_{ent}(X_2) = 0.811$ is selected and answered unfavorably ($u_2 = t$) as well which results in the elimination of only one of four diagnoses $\mathcal{D}_5$. By querying X_3 ($sc_{ent}(X_3) = 0.082$, $u_3 = t$) and X_4 ($sc(X_4) = 0$, $u_4 = t$), the further execution of this procedure finally leads to the target diagnosis $\mathcal{D}_2$. So, application of ENT requires four queries to find $\mathcal{D}^*$. If SPL is used instead, only three queries are required. The algorithm can select one of the two queries X_5 or X_9 because each eliminates half of all

diagnoses in any case. Let the strategy select X_5 which is answered positively ($u_5 = t$). As successive queries, X_6 ($u_6 = f$) and X_1 ($u_1 = f$) are selected, which leads to the revelation of $\mathcal{D}^* = \mathcal{D}_2$.

This scenario demonstrates that the no-risk strategy SPL (*three queries*) is more suitable than ENT (*four queries*) for fault probabilities which disfavor the target diagnosis. Let us suppose, on the other hand, that probabilities are assigned more reasonably in our example, e.g. $\mathcal{D}^* = \mathcal{D}_6$. Then it will take ENT only *two queries* (X_1, X_6) to find $\mathcal{D}^*$ while SPL will still require *three queries*, e.g. (X_5, X_1, X_6).

This example indicates that, unless the target diagnosis is known in advance, one can never be sure to select the best strategy from SPL and ENT. In section 3 we present a learning query selection algorithm that combines the benefits of both SPL and ENT. It adapts the way of selecting the next query depending on the elimination rate (like SPL) and on information gain (like ENT). Thereby its performance approaches the performance of the better of both SPL and ENT.

Table 1. A set $\mathbf{X_D}$ of queries and associated partitions w.r.t. the initial DPI $\langle \mathcal{T}, \mathcal{A}, \emptyset, \emptyset \rangle_{\{\text{coherence}\}}$ of the example ontology $\mathcal{O}$

Query	$\mathbf{D}_i^P$	$\mathbf{D}_i^N$	$\mathbf{D}_i^\emptyset$
$X_1 : \{DeptEmployee(s), Student(s)\}$	$\mathcal{D}_4, \mathcal{D}_6$	$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_5$	$\emptyset$
$X_2 : \{PhD(s)\}$	$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_6$	$\mathcal{D}_5$	$\emptyset$
$X_3 : \{Researcher(s)\}$	$\mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_6$	$\mathcal{D}_1, \mathcal{D}_5$	$\emptyset$
$X_4 : \{Student(s)\}$	$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6$	$\mathcal{D}_3$	$\emptyset$
$X_5 : \{Researcher(s), Student(s)\}$	$\mathcal{D}_2, \mathcal{D}_4, \mathcal{D}_6$	$\mathcal{D}_1, \mathcal{D}_3, \mathcal{D}_5$	$\emptyset$
$X_6 : \{DeptMember(s)\}$	$\mathcal{D}_3, \mathcal{D}_4$	$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_5, \mathcal{D}_6$	
$X_7 : \{PhD(s), Student(s)\}$	$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_4, \mathcal{D}_6$	$\mathcal{D}_3, \mathcal{D}_5$	$\emptyset$
$X_8 : \{DeptMember(s), Student(s)\}$	$\mathcal{D}_2$	$\mathcal{D}_1, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_5, \mathcal{D}_6$	$\emptyset$
$X_9 : \{DeptEmployee(s)\}$	$\mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_6$	$\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_5$	$\emptyset$

3 Risk Optimization for Query Selection

The proposed Risk Optimization Algorithm (RIO) extends ENT strategy with a dynamic learning procedure that learns by reinforcement how to select the next query. Its behavior is determined by the achieved performance in terms of diagnosis elimination rate w.r.t. the set of leading diagnoses $\mathbf{D}$. Good performance causes similar behavior to ENT, whereas aggravation of performance leads to a gradual neglect of the given meta information, and thus to a behavior akin to SPL. Like ENT, RIO continually improves the prior fault probabilities based on new knowledge obtained through queries to a user.

RIO learns a "cautiousness" parameter c whose admissible values are captured by the user-defined interval $[\underline{c}, \overline{c}]$. The relationship between c and queries is as follows:

Definition 4 (Cautiousness of a Query). *We define the* cautiousness $c_q(X_i)$ of a query $X_i \in \mathbf{X_D}$ *as follows:*

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

A query X_i is called braver *than query X_j iff $c_q(X_i) < c_q(X_j)$. Otherwise X_i is called* more cautious *than X_j. A query with maximum cautiousness $\overline{c_q}$ is called* no-risk query.

Definition 5 (Elimination Rate). *Given a query X_i and the corresponding answer $u_i \in \{t, f\}$, the* elimination rate $e(X_i, u_i) = \frac{|\mathbf{D}_i^N|}{|\mathbf{D}|}$ *if $u_i = t$ and $e(X_i, u_i) = \frac{|\mathbf{D}_i^P|}{|\mathbf{D}|}$ if $u_i = f$. The answer u_i to a query X_i is called* favorable *iff it maximizes the elimination rate $e(X_i, u_i)$. Otherwise u_i is called* unfavorable. *The minimal or* worst case elimination rate $\min_{u_i \in \{t,f\}}(e(X_i, u_i))$ *of X_i is denoted by $e_{wc}(X_i)$.*

So, the cautiousness $c_q(X_i)$ of a query X_i is exactly the worst case elimination rate, i.e. $c_q(X_i) = e_{wc}(X_i) = e(X_i, u_i)$ given that u_i is the unfavorable query result. Intuitively, parameter c characterizes the minimum proportion of diagnoses in $\mathbf{D}$ which should be eliminated by the successive query.

Definition 6 (High-Risk Query). *Given a query X_i and cautiousness c, X_i is called a* high-risk query *iff $c_q(X_i) < c$, 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 $NHR_c(\mathbf{X_D}) \subseteq \mathbf{X_D}$ we denote the set of non-high-risk queries w.r.t. c. For given cautiousness c, the set of queries $\mathbf{X_D}$ can be partitioned in high-risk queries and non-high-risk queries.*

Example (continued): Let the user specify $c := 0.3$ for the set $\mathbf{D}$ with $|\mathbf{D}| = 6$. Given these settings, $X_1 := \{DeptEmployee(s), Student(s)\}$ is a non-high-risk query since its partition $\langle \mathbf{D}_1^P, \mathbf{D}_1^N, \mathbf{D}_1^\emptyset \rangle = \langle \{\mathcal{D}_4, \mathcal{D}_6\}, \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_5\}, \emptyset \rangle$ and thus its cautiousness $c_q(X_1) = 2/6 \geq 0.3 = c$. The query $X_2 := \{PhD(s)\}$ with the partition $\langle \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_4, \mathcal{D}_6\}, \{\mathcal{D}_5\}, \emptyset \rangle$ is a high-risk query because $c_q(X_2) = 1/6 < 0.3 = c$ and $X_3 := \{Researcher(s), Student(s)\}$ with $\langle \{\mathcal{D}_2, \mathcal{D}_4, \mathcal{D}_6\}, \{\mathcal{D}_1, \mathcal{D}_3, \mathcal{D}_5\}, \emptyset \rangle$ is a no-risk query due to $c_q(X_3) = 3/6 = \overline{c_q}$.

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

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

where $\varepsilon \in (0, \frac{1}{2})$ 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 $\varepsilon = 0$, the elimination rate of a no-risk query $e(X_s, u_s) = \frac{1}{2}$ resulting always in $adj = 0$. The value of ε can be set to an arbitrary real number, e.g. $\varepsilon := \frac{1}{4}$. 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. Note, for the user-defined interval $[\underline{c}, \overline{c}] \subseteq [\underline{c_q}, \overline{c_q}]$ must hold.

$\underline{c} - c_q$ and $\overline{c_q} - \overline{c}$ represent the minimal desired difference in performance to a high-risk (ENT) and no-risk (SPL) query selection, respectively. By expressing trust (disbelief) in the prior fault probabilities through specification of lower (higher) values for $\underline{c}$ and/or $\overline{c}$, the user can take influence on the behavior of RIO.

Example (continued): Assume $p(ax_i) := 0.001$ for $ax_{i(i=1,\dots,4)}$ and $p(ax_5) := 0.1$, $p(ax_6) := 0.15$ and the user rather disbelieves these fault probabilities and thus sets $c = 0.4$, $\underline{c} = 0$ and $\overline{c} = 0.5$. In this case RIO selects a no-risk query X_3 just as SPL would do. Given $u_3 = t$ and $|\mathbf{D}| = 6$, the algorithm computes the elimination rate $e(X_3, t) = 0.5$ and adjusts the cautiousness by $c_{adj} = -0.17$ which yields $c = 0.23$. This allows RIO to select a higher-risk query in the next iteration, whereupon the target diagnosis $\mathcal{D}^* = \mathcal{D}_2$ is found after asking three queries. In the same situation, ENT (starting with high-risk query X_1) would require four queries.

RIO, described in Algorithm 2, starts with the computation of minimal diagnoses. GETDIAGNOSES function implements a combination of HS-Tree and QuickXPlain algorithms [11]. 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 $\mathbf{D}$ using formula (1). Next it adjusts the probabilities as per the Bayesian theorem taking into account all previous query answers which are stored in P and N. Finally, the resulting probabilities $p_{adj}(\mathcal{D}_i)$ are normalized. Based on the set of leading diagnoses $\mathbf{D}$, GENERATEQUERIES generates queries according to Algorithm 1. GETMINSCOREQUERY determines the best query $X_{sc} \in \mathbf{X_D}$ according to sc_{ent}: $X_{sc} = \arg\min_{X_k \in \mathbf{X_D}}(sc_{ent}(X_k))$. If X_{sc} is a non-high-risk query, i.e. $c \leq c_q(X_{sc})$ (determined by GETQUERYCAUTIOUSNESS), X_{sc} is selected. In this case, X_{sc} is the query with best information gain in $\mathbf{X_D}$ and moreover guarantees the required elimination rate specified by c.

Otherwise, GETALTERNATIVEQUERY selects the query $X_{alt} \in \mathbf{X_D}$ $(X_{alt} \neq X_{sc})$ 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 $L_c := \{X_r \in NHR_c(\mathbf{X_D}) \mid \forall X_t \in NHR_c(\mathbf{X_D}) : c_q(X_r) \leq c_q(X_t)\}$. If there is no such query $X_{alt} \in \mathbf{X_D}$, then X_{sc} is selected.

Algorithm 2: Risk Optimization Algorithm (RIO)

Input: diagnosis problem instance $\langle \mathcal{O}, \mathcal{B}, P, N \rangle_R$, 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 $\mathcal{D}$

```
1  D ← ∅;
2  repeat
3      D ← getDiagnoses(D, n, O, B, P, N);
4      DP ← getProbabilities(DP, D, P, N);
5      X_D ← generateQueries(O, B, P, D);
6      X_s ← getMinScoreQuery(DP, X_D);
7      if getQueryCautiousness(X_s, D) < c then X_s ← getAlternativeQuery(c, X_D, DP, D);
8      if getAnswer(X_s) = yes then P ← P ∪ {X_s};
9      else N ← N ∪ {X_s};
10     c ← updateCautiousness(D, P, N, X_s, c, c_, c¯);
11 until (aboveThreshold(DP, σ) ∨ eliminationRate(X_s) = 0);
12 return mostProbableDiag(D, DP);
```

Given the user's answer u_s, the selected query $X_s \in \{X_{sc}, X_{alt}\}$ is added to P or N accordingly (see Section 2). 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. If a stop condition is met, the presently most likely diagnosis is returned (MOSTPROBABLEDIAG).

4 Evaluation

Goals. This evaluation should demonstrate that (1) there is a significant discrepancy between existing strategies SPL and ENT concerning user effort where the winner depends on the quality of meta information, (2) RIO exhibits superior average behavior compared to ENT and SPL w.r.t. the amount of user interaction required, irrespective of the quality of specified fault information, (3) RIO scales well and (4) its reaction time is well suited for an interactive debugging approach.

Provenance of Test Data. As data source for the evaluation we used faulty real-world ontologies produced by automatic ontology matching systems (cf. Example in Section 2). Matching of two ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ is understood as detection of correspondences between elements of these ontologies [12]:

Definition 7 (Ontology matching). *Let $Q(\mathcal{O}) \subseteq \mathbf{S}(\mathcal{O})$ denote the set of matchable elements in an ontology $\mathcal{O}$, where $\mathbf{S}(\mathcal{O})$ denotes the signature of $\mathcal{O}$. An ontology matching operation determines an* alignment *$\mathcal{M}_{ij}$, which is a set of correspondences between matched ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$. Each* correspondence *is a 4-tuple $\langle x_i, x_j, r, v\rangle$, such that $x_i \in Q(\mathcal{O}_i)$, $x_j \in Q(\mathcal{O}_j)$, r is a semantic relation and $v \in [0, 1]$ is a confidence value. We call $\mathcal{O}_{i\mathcal{M}j} := \mathcal{O}_i \cup \phi(\mathcal{M}_{ij}) \cup \mathcal{O}_j$ the* aligned ontology *for $\mathcal{O}_i$ and $\mathcal{O}_j$ where ϕ maps each correspondence to an axiom.*

Let in the following $Q(\mathcal{O})$ be the restriction to atomic concepts and roles in $\mathbf{S}(\mathcal{O})$, $r \in \{\sqsubseteq, \sqsupseteq, \equiv\}$ and ϕ the natural alignment semantics [6] that maps correspondences one-to-one to axioms of the form $x_i\ r\ x_j$. We evaluate RIO using aligned ontologies by the following reasons: (1) Matching results often cause inconsistency/incoherence 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.

Test Datasets. We used two datasets D1 and D2: Each faulty aligned ontology $\mathcal{O}_{i\mathcal{M}j}$ in D1 is the result of applying one of four ontology matching systems to a set of six independently created ontologies in the domain of conference organization. For a given

pair of ontologies $\mathcal{O}_i \neq \mathcal{O}_j$, each system produced an alignment $\mathcal{M}_{ij}$. The average size of $\mathcal{O}_{i\mathcal{M}j}$ per matching system was between 312 and 377 axioms. D1 is a superset of the dataset used in [13] for which all debugging systems under evaluation manifested correctness or scalability problems. D2, used to assess the scalability of RIO, is the set of ontologies from the ANATOMY track in the Ontology Alignment Evaluation Initiative[3] (OAEI) 2011.5 [12], which comprises two input ontologies $\mathcal{O}_1$ (11545 axioms) and $\mathcal{O}_2$ (4838 axioms). The size of the aligned ontologies generated by results of seven different matching systems was between 17530 and 17844 axioms. [4]

Reference Solutions. For the dataset D1, based on a manually produced reference alignment $\mathcal{R}_{ij} \subseteq \mathcal{M}_{ij}$ for ontologies $\mathcal{O}_i, \mathcal{O}_j$ (cf. [7]), we were able to fix a target diagnosis $\mathcal{D}^* := \phi(\mathcal{M}_{ij} \setminus \mathcal{R}_{ij})$ for each incoherent $\mathcal{O}_{i\mathcal{M}j}$. In cases where $\mathcal{D}^*$ represented a non-minimal diagnosis, it was randomly redefined as a minimal diagnosis $\mathcal{D}^* \subset \phi(\mathcal{M}_{ij} \setminus \mathcal{R}_{ij})$. In case of D2, given the ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the output $\mathcal{M}_{12}$ of a matching system, and the correct reference alignment $\mathcal{R}_{12}$, we fixed $\mathcal{D}^*$ as follows: We carried out (prior to the actual experiment) a debugging session with DPI $\langle \phi(\mathcal{M}_{12} \setminus \mathcal{R}_{12}), \mathcal{O}_1 \cup \mathcal{O}_2 \cup \phi(\mathcal{M}_{12} \cap \mathcal{R}_{12}), \emptyset, \emptyset \rangle_{\{\text{coherence}\}}$ and randomly chose one of the identified diagnoses as $\mathcal{D}^*$.

Test Settings. We conducted 4 experiments EXP-i ($i = 1, \ldots, 4$), the first two with dataset D1 and the other two with D2. In experiments 1 and 3 we simulated good fault probabilities by setting $p(ax_k) := 0.001$ for $ax_k \in \mathcal{O}_i \cup \mathcal{O}_j$ and $p(ax_m) := 1 - v_m$ for $ax_m \in \mathcal{M}_{ij}$, where v_m is the confidence of the correspondence underlying ax_m. Unreasonable fault information was used in experiments 2 and 4. In EXP-4 the following probabilities were defined: $p(ax_k) := 0.01$ for $ax_k \in \mathcal{O}_i \cup \mathcal{O}_j$ and $p(ax_m) := 0.001$ for $ax_m \in \mathcal{M}_{ij}$. In EXP-2, in contrast, we used probability settings of EXP-1, but altered the target diagnosis $\mathcal{D}^*$ in that we precomputed (before the actual experiment started) the 30 most probable minimal diagnoses, and from these we selected the diagnosis with the highest number of axioms $ax_k \in \mathcal{O}_{i\mathcal{M}j} \setminus \phi(\mathcal{M}_{ij})$ as $\mathcal{D}^*$.

Throughout all four experiments, we set $|\mathbf{D}| := 9$ (which proved to be a good trade-off between computation effort and representativeness of the leading diagnoses), $\sigma := 85\%$ and as input parameters for RIO we set $c := 0.25$ and $[\underline{c}, \overline{c}] := [c_{\min}, c_{\max}] = [0, \frac{4}{9}]$. To let tests constitute the highest challenge for the evaluated methods, the initial DPI was specified as $\langle \mathcal{O}_{i\mathcal{M}j}, \emptyset, \emptyset, \emptyset \rangle_{\{\text{coherence}\}}$, i.e. the entire search space was explored without adding parts of $\mathcal{O}_{i\mathcal{M}j}$ to $\mathcal{B}$, although $\mathcal{D}^*$ was always a subset of the alignment $\mathcal{M}_{ij}$ only. In practice, given such prior knowledge, the search space could be severely restricted and debugging greatly accelerated. All tests were executed on a Core-i7 (3930K) 3.2Ghz, 32GB RAM with Ubuntu Server 11.04 and Java 6 installed. [5]

Metrics. Each experiment involved a debugging session of ENT, SPL as well as RIO for each ontology in the respective dataset. In each debugging run we measured the number of required queries (q) until $\mathcal{D}^*$ was identified, the overall debugging time (*debug*) assuming that queries are answered instantaneously and the reaction time (*react*),

[3] `http://oaei.ontologymatching.org`

[4] Source ontologies, produced alignments by each matcher, and reference alignments were downloaded from `http://bit.ly/Zffkow` (D1) and `http://bit.ly/KohlNB` as well as `http://bit.ly/MU5Ca9` (D2).

[5] See `http://code.google.com/p/rmbd/wiki` for code and details.

i.e. the average time between two successive queries. The queries generated in the tests were answered by an automatic oracle by means of the target ontology $\mathcal{O}_{i\mathcal{M}j} \setminus \mathcal{D}^*$.

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

Results of both experimental sessions, ⟨EXP-1,EXP-2⟩ and ⟨EXP-3,EXP-4⟩, are summarized in Figures 2(a) and 2(b), respectively. 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 in {SPL,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 26% of the cases in EXP-2, RIO even 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 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. Recalling Figure 1(b), this means that RIO can avoid query overheads of over 2000%. Table 2, which provides average values for q, *react* and *debug* per strategy, demonstrates that RIO is the best choice in all experiments w.r.t. q. Consequently, RIO is suitable for both good and poor meta information.

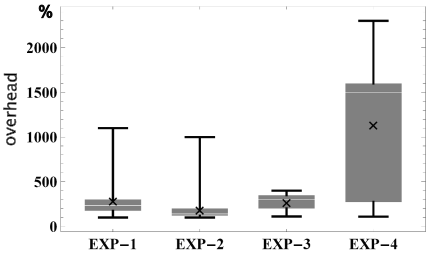
As to time aspects, RIO manifested good performance, too. Since times consumed in ⟨EXP-1,EXP-2⟩ are almost negligible, consider the more meaningful results obtained in ⟨EXP-3,EXP-4⟩. While the best reaction time in both experiments was achieved by SPL, we can clearly see that SPL was significantly inferior to both ENT and RIO concerning q and *debug*. 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., 30 sec 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, RIO

Table 2. 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

	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**

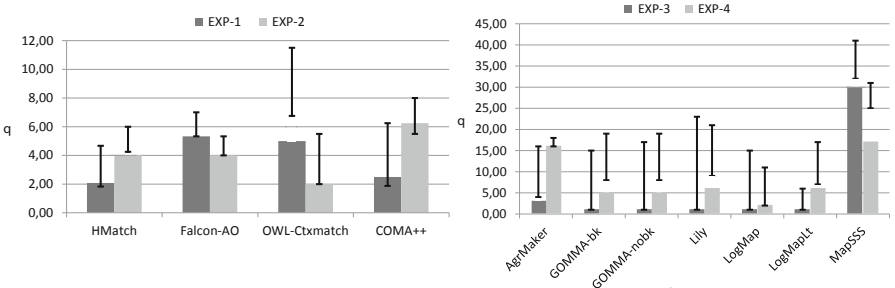
	EXP-1	EXP-2	EXP-3	EXP-4
$q_{\text{SPL}} < q_{\text{ENT}}$	11%	37%	0%	29%
$q_{\text{ENT}} < q_{\text{SPL}}$	81%	56%	100%	71%
$q_{\text{SPL}} = q_{\text{ENT}}$	7%	7%	0%	0%
$q_{\text{RIO}} < \min$	4%	26%	29%	71%
$q_{\text{RIO}} \leq \min$	74%	74%	100%	100%

(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.



(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.

would save 25% (w.r.t. ENT) and 50% (w.r.t. SPL) of debugging time on average. All in all, the measured times confirm that RIO is well suited for *interactive* debugging.

5 Related Work

A similar interactive technique was presented in [8], where a user is successively asked single ontology axioms in order to obtain a partition of a given ontology into a set of desired and a set of undesired consequences. However, given an inconsistent/incoherent ontology, this technique starts from an empty set of desired consequences aiming at adding to this set only axioms which preserve coherence, whereas our approach starts from the complete ontology aiming at finding a minimal set of axioms responsible for the violation of pre-specified requirements.

An approach for alignment debugging was proposed in [5]. This work describes approximate algorithms for computing a "local optimal diagnosis" and complete methods to discover a "global optimal diagnosis". Optimality in this context refers to the maximum sum of confidences in the resulting coherent alignment. In contrast to our framework, diagnoses are determined automatically without support for user interaction.

Instead, techniques for manual revision of the alignment as a procedure *independent* from debugging are demonstrated.

6 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 good scalability and reaction time as well as superior average performance to two common approaches in the field in all tested cases w.r.t. required user interaction. Highest achieved savings amounted to more than 80% and user interaction overheads resulting from the wrong choice of strategy of up to 2300% could be saved. In the hardest test cases, the new strategy was not only on average, but in 100% of the test cases at least as good as the best other strategy.

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. Friedrich, G., Shchekotykhin, K.: A General Diagnosis Method for Ontologies. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 232–246. Springer, Heidelberg (2005)
3. Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
4. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all Justifications of OWL DL Entailments. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
5. Meilicke, C.: Alignment Incoherence in Ontology Matching. Ph.D. thesis, Universität Mannheim (2011)
6. Meilicke, C., Stuckenschmidt, H.: An Efficient Method for Computing Alignment Diagnoses. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 182–196. Springer, Heidelberg (2009)
7. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Reasoning Support for Mapping Revision. Journal of Logic and Computation 19(5), 807–829 (2008)
8. Nikitina, N., Rudolph, S., Glimm, B.: Interactive Ontology Revision. Journal of Web Semantics 49(721) (2011)
9. Reiter, R.: A Theory of Diagnosis from First Principles. Artif. Intell. 32(1), 57–95 (1987)
10. Schlobach, S., Huang, Z., Cornet, R., Harmelen, F.: Debugging Incoherent Terminologies. Journal of Automated Reasoning 39(3), 317–349 (2007)
11. 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)
12. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering X(X), 1–20 (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), Tenerife, Spain, pp. 1–12 (2008)