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. Interactive ontology debugging incorporates a user who answers queries about entailments of their intended ontology. In order to minimize the amount of user interaction in a debugging session, a user must choose an appropriate query selection strategy. However, the choice of an unsuitable strategy may result in tremendous overhead in terms of time and cost. We present a learning method for query selection which unites the advantages of existing approaches while overcoming their flaws. Our tests show the utility of our approach when applied to a large set of real-world ontologies, its scalability and adequate reaction time allowing for continuous interactivity.

Motivation and Concept:
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. Ontology matching aims at generating a set of semantic links, called alignment, between elements of two standalone ontologies describing related domains. The two ontologies together with the produced alignment, called the aligned ontology, may exhibit a very complex fault structure as a consequence of (1) adding many links between the single ontologies at once and since (2) the actual fault may be located in the produced alignment and/or in one or both of the single ontologies, e.g. if a correct link between two concepts "activates" a source of inconsistency in one of the single ontologies. This makes evident that adequate tool assistance in debugging of such ontologies is indispensable.

Ontology debugging deals with the following problem: Given an ontology $\mathcal{O}$ which does not meet postulated requirements R,[1] 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. Generally, there are many alternative diagnoses for one and the same faulty ontology $\mathcal{O}$. The problem is then to figure out the single diagnosis, called target diagnosis $\mathcal{D}_t$, that enables to formulate the target ontology $\mathcal{O}_t$ featuring the user-intended semantics in terms of entailments and non-entailments. The target ontology can be understood as $\mathcal{O}$ minus the axioms of $\mathcal{D}_t$ plus additional axioms $EX_{\mathcal{D}_t}$ which can be added in order to regain desired entailments which might have been eliminated together with axioms in $\mathcal{D}_t$.

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 target ontology $\mathcal{O}_t$. Roughly speaking, each query is a set of axioms and the user is queried whether the conjunction of these axioms is entailed by $\mathcal{O}_t$. Every positively (negatively) answered query constitutes a positive/entailed (negative/non-entailed) test case fulfilled by $\mathcal{O}_t$. Test cases can

[1] Throughout this work $R = \{\text{consistency, coherency}\}$.

be seen as constraints $\mathcal{O}_t$ must satisfy and are therefore used to gradually reduce the search space for valid diagnoses. Simply put, the overall procedure consists of (1) computing a predefined fixed number of diagnoses (the set of leading diagnoses $\mathbf{D}$, usually $|\mathbf{D}| \approx 10$) as an approximation of all diagnoses, (2) gathering additional information by querying the user, i.e. adding a positive or negative test case, (3) incorporating this information to cut irrelevant areas off the search space, i.e. eliminating diagnoses not complying with the newly specified test case. This loop is continued until the search space is reduced to a single (target) diagnosis $\mathcal{D}_t$. The goal is to achieve this with a minimal number of queries to the user.

The best currently known interactive debugging systems pursue active learning strategies for query generation exploiting meta information in terms of fault probabilities of the user who formulates the ontology. Such a system is described in [1] where fault probabilities are used to calculate for each diagnosis the probability of being the target diagnosis. At each step, the query is selected 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. This entropy-based strategy (ENT) can speed up the debugging procedure if probabilities are specified appropriately, but can also have substantial negative impact on the performance in case of unreasonable probabilities. The problem is that assessment of probabilities is only possible a-posteriori. Consequently, as long as the actual fault is unknown, there is always some risk of suboptimal query selection.

As an alternative, one might prefer to rely on an approach with constant performance which pursues a no-risk strategy without taking into account any meta information. One such strategy is split-in-half (SPL) [1], which selects the query which eliminates half of the leading diagnoses, independent of the answer to the query. In this case, however, possibly well-chosen fault probabilities cannot be exploited, resulting again in inefficient debugging actions. To sum up, the user may choose between a strategy with high potential and high risk and a strategy with no risk and no potential.

Therefore, we introduce a method with high potential and low risk, which can be seen as a hybrid risk optimization method (RIO) exploiting positive aspects of both ENT and SPL. On the one hand, our method takes advantage of the given probabilities as long as good performance is achieved. On the other hand, it gradually gets more independent of meta information if suboptimal behavior is measured. This is accomplished by constantly adapting a reinforcement learning parameter $c \in [0, 0.5]$, which can be seen as the minimal postulated "cautiousness" of the next selected query. The cautiousness of a query is equivalent to its worst case elimination rate w.r.t. the set of leading diagnoses $\mathbf{D}$. E.g., if $|\mathbf{D}| = 10$ and a query Q_1 eliminates 1 (9) leading diagnoses for positive (negative) answer, the cautiousness of Q_1 is $\frac{1}{10}$, whereas a query Q_2 with 5 (5) has an elimination rate of $\frac{5}{10}$. W.r.t., e.g. $c = 0.3$, Q_1 would be a high-risk-query since it eliminates less than $0.3 * 100\%$ of leading diagnoses in the worst case. Thus, it would be dismissed by RIO as a candidate for the next query. By contrast, Q_2 is a non-high-risk-query as it eliminates more diagnoses than claimed by c anyway. Actually, Q_2 is even a no-risk-query because it eliminates 50% of diagnoses in $\mathbf{D}$ in any case. Given a query Q_3 with 7 (3), we call Q_3 more cautious than Q_1 and less cautious than Q_2.

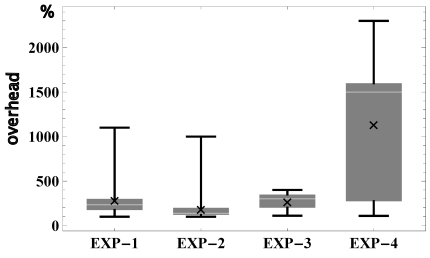
More concretely, RIO works as follows: Select the same query Q_{ENT} as ENT would select, if the cautiousness of Q_{ENT} is greater or equal c. Otherwise, select the query with best entropy-measure among all (if more than one) least cautious non-high-risk-queries. In the (rare) situation that no such query exists, select Q_{ENT}. E.g., let current $c = 0.3$

and $Q_{\text{ENT}} = Q_1$, then RIO would select Q_3 since it has cautiousness $0.3 = c$ and is thus the only least cautious non-high-risk-query. After each answered query, the new information is taken into account by updating the diagnosis probabilities according to the Bayesian rule [1]. Additionally, the cautiousness parameter $c \leftarrow c + a$ is adjusted by a value a which is proportional to 0.5 minus the actual achieved elimination rate of the current query. So, for a an elimination of more than half of the leading diagnoses, RIO gets a bonus ($a < 0$) allowing it to take more risk in the next iteration. Otherwise, a penalty is imposed implying more cautious successive behavior.

The following evaluation will demonstrate that, independently of the quality of specified meta information, RIO exhibits superior average performance compared to ENT and SPL w.r.t. the amount of user interaction required. Furthermore, experiments will show that RIO scales well and that the reaction time measured is well suited for an interactive debugging approach.

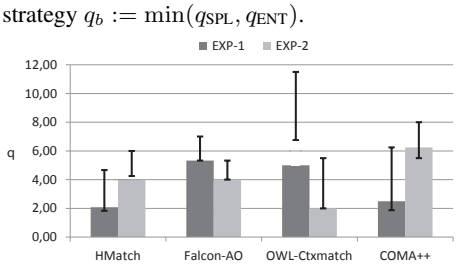
	EXP-1	EXP-2	EXP-3	EXP-4
$q_{\text{SPL}} < q_{\text{ENT}}$	11%	37%	0%	29%
$q_{\text{ENT}} < q_{\text{SPL}}$	81%	56%	100%	71%
$q_{\text{SPL}} = q_{\text{ENT}}$	7%	7%	0%	0%
$q_{\text{RIO}} < \min$	4%	26%	29%	71%
$q_{\text{RIO}} \leq \min$	74%	74%	100%	100%

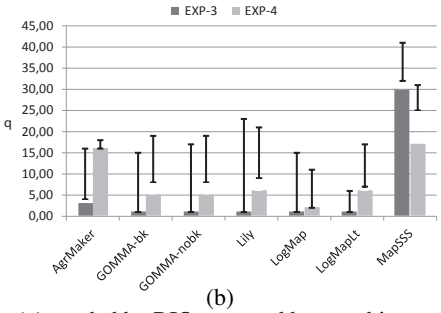
(a)



(b)

Fig. 1. (a) Percentage rates indicating which strategy performed better w.r.t. number of queries. 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 $q_b := \min(q_{\text{SPL}}, q_{\text{ENT}})$.





(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 between q and the queries needed by the per-session better (worse) strategy q_b (q_w). Notation as in Figure 1.

Evaluation:

We performed four experiments EXP-$i(i = 1, \ldots, 4)$ where we applied RIO to a set of incoherent ontologies produced by automatic ontology matchers.[2] As data source for EXP-1 and EXP-2 we used a superset of the dataset[3] used in [2] where it was shown

[2] For details and further results, see http://code.google.com/p/rmbd/wiki/OntologyAlignmentAnatomy

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

that existing debugging approaches suffer from serious problems w.r.t. both scalability and correctness of results when tested on this dataset. As an interactive approach able to query and incorporate additional information into its computations, RIO can cope with cases unsolved in [2]. For the scalability tests in EXP-3 and EXP-4, we used the set of ontologies from the ANATOMY track in the Ontology Alignment Evaluation Initiative (OAEI) 2011.5, which comprises two input ontologies $\mathcal{O}_1$ (Human, 11545 axioms) and $\mathcal{O}_2$ (Mouse, 4838 axioms). The faulty aligned ontologies had up to 17844 axioms.

Available reference alignments enabled to fix a target diagnosis $\mathcal{D}_t$ for each incoherent ontology. Throughout all experiments, unlike state-of-the-art alignment debuggers, we considered the most general problem where the search for the target diagnosis is not restricted to the alignment. In each test run we measured the number of required queries until $\mathcal{D}_t$ was identified. All tests were executed on a Core-i7 (3930K) 3.2Ghz, 32GB RAM and with Ubuntu Server 11.04 and Java 6 installed. In EXP-1/EXP-3 fault probabilities were chosen reasonably (good case), whereas in EXP-2/EXP-4, they were specified in a way $\mathcal{D}_t$ got very improbable (bad case). Queries were answered by an automatic oracle by means of the target ontology obtained through $\mathcal{D}_t$.

Results of $\langle$EXP-1,EXP-2$\rangle$ and $\langle$EXP-3,EXP-4$\rangle$, are summarized in Figure 2(a) and Figure 2(b), respectively. The results illustrate clearly that avg. 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 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 (improvement of $> 80\%$ and 53%, respectively). In $\langle$EXP-3,EXP-4$\rangle$, the savings achieved by RIO were even more substantial (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. Concerning average number of queries per strategy, RIO is the best choice in all experiments. Consequently, RIO is suitable for both good meta information (EXP-1/EXP-3) and poor meta information (EXP-2/EXP-4). Moreover, assuming a user being capable of reading and answering a query in, e.g., half a minute on average, RIO shows best perfromance w.r.t. overall debugging time with savings of up to 50% compared to ENT/SPL. Reaction time of RIO, i.e. avg. time between two successive queries, was always < 12.9s. 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 (Figure 1(b)). Moreover, results show that the different quality of probabilities in {EXP-1,EXP-3} versus {EXP-2,EXP-4} clearly affected performance of ENT and SPL strategies (Figure 1(a)). This perfectly motivates the application of RIO.

References

1. 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)
2. 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)