Multi-Label Based Learning for Better Multi-Criteria Ranking of Ontology Reasoners

Nourhène Alaya[1,2], Myriam Lamolle[1], and Sadok Ben Yahia[2]

LIASD, University of Paris 8, IUT of Montreuil, Saint-Denis, France.[1]
LIPAH, University of Tunis El-Manar, Faculty of Sciences, Tunis, Tunisia.[2]
{n.alaya,m.lamolle}@iut.univ-paris8.fr, sadok.benyahia@fst.rnu.tn

Abstract. A growing number of highly optimized reasoning algorithms have been developed to allow inference tasks on expressive ontology languages such as OWL(DL). Nevertheless, there is broad agreement that a reasoner could be optimized for some, but not all the ontologies. This particular fact makes it hard to select the best performing reasoner to handle a given ontology, especially for novice users. In this paper, we present a novel method to support the selection ontology reasoners. Our method generates a recommendation in the form of reasoner ranking. The efficiency as well as the correctness are our main ranking criteria. Our solution combines and adjusts multi-label classification and multi-target regression techniques. A large collection of ontologies and 10 well-known reasoners are studied. The experimental results show that the proposed method performs significantly better than several state-of-the-art ranking solutions. Furthermore, it proves that our introduced ranking method could effectively be evolved to a competitive meta-reasoner.

Keywords: Ontology, Reasoner, Multi-label classification, Multi-target regression, Multi-criteria, Ranking, Advising, Meta-reasoning

1 Introduction

A growing number of highly optimized ontology reasoners [10] have been developed to allow inference tasks on expressive ontology languages such as OWL(DL) [6]. Nevertheless, it is well accepted that a reasoner could be optimized for some but not all the ontologies. Indeed, the respective authors of [18, 5] have outlined that, often in practice, reasoners tend to exhibit unpredictable behaviours when dealing with real world ontologies. They noticed that the reasoner performances can considerably vary across the ontologies, even when the size or/and the expressivity of these ones are fixed. Furthermore, Gardiner et al. [4] and more recently Lee et al. [9] pinpointed out that reasoners may disagree over inferences or query answers, computed from the same input ontology. All of the aforementioned authors offered different explanations of these phenomena: bottlenecks in the ontology design [5]; interactions between reasoning optimisation techniques [4]; or even reasoner implementation bugs [9]. Given all of these findings, it is obvious that for a typical OWL user, deciding the most performing reasoner to handle a given ontology is not a trivial task.

Recently, we conducted a preliminary study [2] on designing a system to support users in reasoner selection task for the classification of OWL ontologies. The proposed system, called *RakSOR*, automatically ranks a set of candidate reasoners based on their predicted robustness. The ranked list gathers relevant reasoners, those capable to achieve the reasoning task within a fixed time limit and to deliver *correct* results. It also includes the irrelevant ones. Hence, a user could be informed about what reasoners to select and what to avoid. To put this specification into practice, we defined a set of preference rules based on bucket order principal [3], a special case of partial order. Our method showed good ranking prediction quality comparing to our baseline method[1]. Nevertheless, we admit that the prediction accuracy of the RakSOR algorithm depends heavily on the accuracy of the robustness predictive model of each reasoner. This dependency makes it difficult to further improve the effectiveness of its results. Furthermore, the learning process of the various predictive models is time consuming and highly complex. Besides, we only studied reasoners specifically tuned for the OWL 2 DL profile.

Motivated enough, in this paper, we present a novel accuracy boosted solution for reasoner automatic selection, based on multi-label learning paradigm [16, 20]. We also demonstrate that this method could be used as the core component of a very competitive meta-reasoner. The novel solution, called **Multi-RakSOR**, uses reasoner efficiency and result correctness as main ranking criteria. In addition to our previous reasoner preference rules and order principal [2], Multi-RakSOR considers the OWL profile of the input ontology. Indeed, it supports both OWL 2 DL and EL profiles. Subsequently, it has two sets of alternative reasoners, i.e. EL specialised and DL fully fledged ones. Multi-RakSOR maps the feature values describing an input ontology into a complete ranking over a set of alternative reasoners. It also indicates the expected relevance of each of the ranked reasoners. To achieve this end, the introduced ranking solution combines, in a *consistent* way, multi-label classification [20] and multi-target regression [14] techniques. Moreover, Multi-RakSOR is capable to swiftly compute the ontology features and the ranking predictions. This has particularly incited us to design of a novel meta-reasoner[8] based on our ranking solution.

The main contributions of this paper can be summarized as follows:

1. The design and the implementation of a novel multi-label based solution for the multi-criteria ranking of ontology reasoners.
2. The planning of a large scale experimental evaluations that covers 10 well known reasoners and 1954 unique ontologies, collected from the corpus of the latest Ontology Reasoner Evaluation Workshop (ORE'2015) [13].
3. The comparison of several multi-label learning algorithms. Results shows that Multi-RakSOR performs significantly better in terms of ranking and relevance prediction quality, comparing to the state-of-the-art multi-label solutions.

[1] It is a trivial solution which outputs the same reasoner ranking regardless of the ontology under study.

4. The preliminary study of a meta-reasoner based on multi-label ranking. Evaluation results proves that it can outperform all of the studied reasoners in terms of result correctness. It also highlights that, at average, the meta-reasoner can significantly boost the reasoning efficiency on DL ontologies, but it is less capable on EL ontologies.

2 Background and related works

We recall some multi-label learning key notions. Then, we review known multi-label based works for algorithm selection. Finally, we describe the ontology features employed in our learning process.

2.1 Key notions of multi-label learning paradigm

In the multi-label learning context [16, 20, 14], each input instance is characterized by a d-dimensional feature vector $\mathbf{X}^{(i)} = (x_1^i, x_2^i, \dots, x_d^i)$, associated with a set of m output labels $\mathbf{Y}^{(i)} = (y_1^i, y_2^i, \dots, y_m^i)$. Let $\mathcal{X}$ be the space domain of the input features and let $\mathcal{Y}$ be the domain of the output labels, also called the target variables space. The task of multi-label learning is to train a model, i.e. a function $h : \mathcal{X} \to \mathcal{Y}$. The model is capable to predict the proper output label vector $\widehat{Y} = (\widehat{y_1}, \widehat{y_2}, \dots, \widehat{y_m})$, given the feature vector $X \in \mathcal{X}$ of an unseen input instance. The model is learned from a dataset $\mathcal{D} = \{(X^{(1)}, Y^{(1)}), \dots, (X^{(n)}, Y^{(n)})\}$, which assembles n training examples.

2.2 Multi-label learning techniques for algorithm selection

In our context, an input instance stands for the vector of feature values which describes a user ontology. On the other hand, each target variable stands for a reasoner. More precisely, a target label depicts a reasoner rank or any other score value. Indeed, the target labels could be real-valued, binary, ordinal, categorical or even of mixed types. Each form of target labels has specific multi-label learning task. In this paper, we are interested in three of these tasks. They are mainly:

- **Multi-Label Classification** task (MLC) [20]. It is concerned with learning a model that outputs a bipartition of the output labels into relevant label set P_x and irrelevant label set N_x, where $P_x \cap N_x = \varnothing$ and $P_x \cup N_x = \mathcal{Y}$. Literally, the outputs are binary labels, i.e. $\mathcal{Y} = \{0, 1\}^m$, with 0 means irrelevant target. This approach was used by Olmo et al. [12] to introduce a recommendation system of relevant machine learning algorithms.
- **Label Ranking** task (LR) [20]. It is concerned with learning a model that outputs an ordering of the labels according to their relevance to the input instance. Hence, the h function maps every instance $X \in \mathcal{X}$ to a total strict order, $\prec$, over the set of the output labels. A ranking over $\mathcal{Y}$ can conveniently be represented by a permutation σ of the set of indices $\{1, \dots, m\}$, where $\sigma(i)$ stands for the rank value of the target variable y_i. LR techniques are the building blocks of various algorithm selection systems: meta-learning solutions [15], SAT solver portfolios [11] and the ontology meta-reasoner R_2O_2 [8].

- **Multi-target Regression** task (MTR) [14]. It is also known as the multivariate or multi-output regression task. It is the most general form of multi-label learning task. Indeed, MTR techniques are designed to predict multiple real-valued target variables, whether they are binary ones (MLC case), permutations (LR case) or float values. Formally, the MTR output space has the following form: $\mathcal{Y} \equiv \mathbb{R}^m$. To best of our knowledge, no previous work have employed MTR techniques to rank algorithms.

Based on this review, it seems that label ranking (LR) techniques are promising solutions for the automatic selection of ontology reasoners. Indeed, they simplify the ranking process, i.e. just one predictive model to train and they are known to be highly accurate. Nevertheless, we are convinced that they cannot satisfy all of our requirements. Actually, all of the reviewed works employ single ranking criterion. Besides, they output the rank values of the alternatives without any indication of their relevance. In fact, the computed rankings follow strict total order, with the assumption that all the alternatives are relevant ones. This is a quite different specification from what we need to fulfil. As previously explained, we are interested in ranking relevant and irrelevant reasoners by applying partial order rules. The real challenge is to find a way to incorporate our multi-criteria preference rules in a multi-label learning method without any precision or information lost. Details of the proposed solution to overcome this challenge are outlined in the forthcoming sections.

2.3 Ontology features

In our previous work [1], we proposed a rich set of ontology features, qualitative and quantitative ones, covering a broad range of structural and syntactic attributes of OWL ontologies. These features were put forward to thoroughly describe the ontology design complexity. Our collection gathers a lot of well known state-of-art metrics, some of them are already used reasoner prediction solutions and in ontology quality evaluation systems. Our features are arranged into four categories: (1) *size description* features, which characterize the amount of knowledge explicitly asserted in the ontology; (2) *expressivity description* features, which mainly includes the OWL profile and the description logic family name; (3) *structural features*, which outline the design of named class and property respective inheritance hierarchies; (3) *syntactic features*, which delineate the main characteristic of the OWL grammar. To compute feature values, any full translation of the OWL ontology to particular kind of graph representation is avoided. In this paper experimentations, we discarded the metrics with high-computing cost, like the tree depth of named class hierarchy. This left us with 123 ontology feature values to be measured.

3 Novel Multi-label Learning Method for Multi-criteria Ranking of Ontology Reasoners

In this section, we introduce the **Multi-RakSOR** method. We give a short specification of its reasoner ranking rules, before outlining its learning mechanism

3.1 Reasoner ranking criteria and preference rules

A ranking represents a *preference function* over a set of alternatives. In our context, the alternatives are some set of ontology reasoners considered as promising candidates. This set is denoted by $\mathcal{R}$. In [1, 2], we stressed on the importance of considering not only the runtime of reasoners but also the correctness of their derived results, as comparison criteria. We also highlighted the need to specify particular reasoner robustness judgement constraints, like the range of ontologies, the reasoning task and the success/failure respective states.

In this paper, our study concerns the classification task of OWL DL and OWL EL ontologies within a tight time schedule. We apply the Gardiner et al. [4] reasoner correctness checking method. Accordingly, results delivered by a reasoner are either correct or unexpected. Subsequently, we can specify a first ordinal criterion which split up the set of reasoners into four groups according to their termination state: (1) **S**uccess, when the reasoner terminates the task within a fixed time-limit and delivers correct results; (2) **U**nexpected, in case of an achieved task within the time limit, but has unexpected results, i.e. incorrect; (3) **T**imeout, when the fixed time lapse is exceeded; and (4) **H**alt), when the reasoner crashes and do not terminate the task. Given this specification, we can formally describe the preference rules over ontology reasoners using bucket order principals [3]. In short, a bucket is a set of equally ranked alternatives. Initially, four buckets are defined each of them corresponds to a specific termination state (S,U,T,H). A strict total order over the buckets is also decided: $\mathcal{B}_S \prec \mathcal{B}_U \prec \mathcal{B}_T \prec \mathcal{B}_H$. Clearly, reasoners belonging to $\mathcal{B}_S$ bucket are the most preferred ones. The $\mathcal{B}_U$ reasoners can terminate the reasoning task but the correctness of their results are not approved by our correctness checking method. In our opinion, they are much preferred than reasoners falling in the $\mathcal{B}_T$ bucket, which can not release any results within the fixed time lapse. Of course, the worse reasoners are in $\mathcal{B}_H$ bucket. Seeking more precision, a second ranking criterion standing for the efficiency of the reasoner over the *correctly* classified ontologies is applied. Accordingly, reasoners within the *success* bucket $\mathcal{B}_S$ are linearly sorted in an increasing order w.r.t their reasoning runtime. The final ordered bucket partition over the set of the alternative reasoners, i.e. $\mathcal{R}$, has the following form: $\mathbf{B} = B_1 \prec \ldots \prec B_k \prec \mathcal{B}_U \prec \mathcal{B}_T \prec \mathcal{B}_H$, where $k = |\mathcal{B}_S|$. Reasoner ranks are computed by following these rules. Subsequently, ties may appear in the list of ranks. Indeed, tied reasoners imply that they did not succeed to classify the ontology for the same failure cause.

Multi-RakSOR considers a further important criterion. This is the OWL profile of the input ontology. It is widely known that some reasoners are specifically tuned to particular OWL profiles. For instance, ELK is a highly performing OWL EL specialised reasoner. On the other hand, there is no proof of the correctness of its results when applied to OWL DL ontologies. In short, it is absurd to advise an EL reasoner to handle a DL ontology. Given this fact, Multi-RakSOR splits the set of alternative reasoners into DL and EL specialised subsets, i.e. $\mathcal{R}_{DL}$ and $\mathcal{R}_{EL}$. Once the profile of an input ontology is identified, the above usual ranking rules are applied over the corresponding set of reasoners.

3.2 Specification of the novel multi-label ranking method

In the Multi-RakSOR system, the ranks of reasoners are computed prior to any learning step, by applying to afore-described rules on actual results of reasoner evaluations. The produced ranks together with the metrics describing the studied ontologies are then, provided to the learning component. Afterwards, a multi-label predictive model is trained, to be able to predict these ranks for future unseen ontologies. We assert that providing only the reasoner ranks might be misleading for the users. This is because the ranked list might include some, or even only, reasoners expected to fail the classification of the input ontology. Hence, it is important not only to predict the ranks but also to outline the successful/unsuccessful reasoners.

To satisfy all of these requirements, we introduce a novel multi-label ranking method applied to ontology reasoners. The learning process involves two equally important subsequent goals. The first is to produce the bipartition of set of the output labels $\mathcal{Y}$ into relevant label set P_x and irrelevant label set N_x, with $P_x \cup N_x = \mathcal{Y}$ and $P_x \cap N_x = \emptyset$. The second is a ranking over $\mathcal{Y}$ which respects the previously introduced preference rules. Specially, the ranking should be consistent, this means to satisfy the following couple of conditions:

- There should be no irrelevant labels ranked higher than relevant ones and vice versa. Formally, whenever $\forall y_i \in P_x$ and $\forall y_j \in N_x$, then $y_i \prec y_j$. In other words, y_i is preferred to y_j and it is ranked lower $\sigma(i) < \sigma(j)$.
- The relevant labels must form a strict total ordered set. Formally, $\forall y_i, \forall y_j \in P_x$ with $i \neq j$, then either y_i is preferred to y_j or y_j is preferred to y_i. The irrelevant labels are allowed to have equal ranks.

In the remainder of this paper, by *relevant* we refer to a reasoner expected to terminate the classification task within the fixed time limit and to deliver correct results, otherwise it is said to be irrelevant.

3.3 Multi-RakSOR learning and prediction steps

To put the above specification into practice, a transformation of the multi-label ranking process is proposed. Indeed, the key idea of our solution is to learn a separate multi-label model for each of the following sub-problems: *(i)* a model to predict the ranking with ties of the alternative reasoners, denoted by $h_r()$ and *(ii)* a model to predict the relevance of each reasoner, denoted by $h_b()$. Afterwards, at the prediction time, the computed relevance bipartition and ranking of reasoners for an input ontology, are synchronized by checking their consistency and probably correcting their values. A further issue is about predicting a ranking with ties using multi-label learning techniques. We previously highlighted the lack of multi-label based solutions to predict a ranking which involves a partial order. To overcome this absence, multi-target regression (MTR) techniques [14] are employed. The latter ones can handle different kinds of learning problems provided that the domain of the output variables is within $\mathbb{R}^m$. In other words, no matter whether the rank values are strict or tied, the MTR model will try to predict the closest possible values to the real ones. Now, we can list the required steps to train the Multi-RakSOR predictive model.

Training time. the Multi-RakSOR model h_{mlti} is comprised of 2 sub-models: a multi-label classification (**MLC**) model, $h_b : \mathcal{F}^d \rightarrow \{0,1\}^m$, which predicts the relevance of the output labels, and a multi-target regression (**MTR**) model, $h_r : \mathcal{F}^d \rightarrow \mathbb{R}^m$, which predicts the ranking with ties of these labels. Each of these models is learned independently from dedicated datasets, respectively $\mathcal{D}_b$ and $\mathcal{D}_r$. The latter ones share the same input vectors, which describes the features of the ontologies. It is important to note that the training datasets are profile specific ones. In other terms, for each supported OWL profile, a dataset is assembled and a dedicated Multi-RakSOR predictive model is trained.

Prediction time. During this online stage, to compute the ranking for a new introduced ontology, our system operates in five steps. Firstly, the feature values of the introduced ontology are computed and provided to the prediction component. Afterwards, the Multi-RakSOR predictive model which corresponds to the ontology OWL profile is invoked. Then, the MLC sub-model is applied to get the predicted relevance of each reasoner. Similarly, the MTR model is addressed to get the predicted ranks. Finally, the consistency of the computed ranks are checked and probably adjusted. More details about our ranking checking method are provided in the upcoming subsection.

3.4 Ranking consistency checking method

Based on the specification provided in Subsection 3.2, the ranking checking method must ensure that: *1)* the rank values of the relevant labels form a strict total ordered set of natural numbers; and *2)* no irrelevant reasoner is ranked lower than a relevant one. If one of these rules is broken, then the ranking is adjusted. Algorithm 1 shows the steps achieved by *rankingCheckingMethod().* The procedure takes as input the matrix $\widehat{\mathbf{Y}}$, which encodes the different computed predictions. We design by $max_{\sigma}^{P_x}$ the maximal rank value of the relevant output labels. Known that the ranks take values in $\mathbb{N}^*$ and they are expected to be linearly sorted in an increasing strict order, then we can assert that $max_{\sigma}^{P_x} = |P_x|$. Through the first loop of Algorithm 1, the $max_{\sigma}^{P_x}$ value is computed. The rank values corresponding to the relevant labels are stored in the R_x array. The cells of this array corresponding to irrelevant reasoners are set to 0. The resulting R_x array is then handled by *rankOrderTransformation().* The main role of this function is to ensure the application of our 1^st consistency rule. This idea behind this function is quite straightforward. The ranks in the R_x array are seen as numerical scores. By consequence, the function computes the strict total order of their values. Potential ties of the relevant reasoners rank values are arbitrary broken. For instance, let $[1, 3, 3, 3, 5, 4]$ be the predicted ranks of 6 reasoners and $[1, 0, 1, 1, 1, 0]$ their predicted relevance. By substituting the ranks of the irrelevant reasoners by 0, the resulting R_x array is equal to $[1, 0, 3, 3, 5, 0]$. It is clear

Algorithm 1: The ranking consistancy checking method

```
1  Function rankingCheckingMethod($\widehat{\mathbf{Y}} \in M_{m\times 2}(\mathbb{R})$)
2  $\mathbf{R}_x \leftarrow [0,\ldots,0]$ ; // $\mathbf{R}_x$ is of size $m$
3  $max_{\sigma}^{P_x} \leftarrow 0$ ;
4  for $i \leftarrow 1$ to $|\widehat{\mathbf{Y}}|$ do
5      if $\widehat{\mathbf{Y}}[i][1] = 1$ then
6          $\mathbf{R}_x[i] \leftarrow \widehat{\mathbf{Y}}[i][2]$ ;
7          $max_{\sigma}^{P_x} \leftarrow max_{\sigma}^{P_x} + 1$ ;
8      end
9  end
10 $\mathbf{R}_x \leftarrow rankOrderTransformation(\mathbf{R}_x)$ ;
11 for $i \leftarrow 1$ to $|\widehat{\mathbf{Y}}|$ do
12     if $\widehat{\mathbf{Y}}[i][1] <> 0$ then // Relevant label, update the rank
13         $\widehat{\mathbf{Y}}[i][2] \leftarrow \mathbf{R}_x[i]$ ;
14     else if $\widehat{\mathbf{Y}}[i][2] \leq max_{\sigma}^{P_x}$ then // Irrelevant label, Inconsistency case
15         $updateIrrelevantRanks(\widehat{\mathbf{Y}}, max_{\sigma}^{P_x}, i)$ ;
16     end
17 end
18 return $\widehat{\mathbf{Y}}$ ;
```

that these are non consistent rank values[2]. In this case, *rankOrderTransformation()* function ignores the 0 values and considers the remaining values as scores to be ranked. It finally returns $[1, 0, 2, 3, 4, 0]$. Afterwards, the inconsistencies w.r.t. the second rule are caught by simply verifying whether an irrelevant label has a rank lower than $max_{\sigma}^{P_x}$ (see Algorithm 1, Line 14). If this is the case, then the first inconsistent rank is set to $(max_{\sigma}^{P_x}+1)$, and subsequently, all the remaining rank values of irrelevant reasoners are updated. Our solution is intuitive and inexpensive one, capable at least to fix the inconsistencies. As a matter of fact, in our opinion, the exact ranking of the irrelevant reasoners is less important for the user. In this example, the final adjusted ranking is equal to $[1, 5, 2, 3, 4, 6]$.

4 Data Collection

To build up the Multi-RakSOR system, data describing the empirical performances of reasoners are required. Therefore, a large scale evaluations of an important number of reasoners is conducted using the evaluation tools employed in the latest Ontology Reasoner Evaluation Workshop (ORE 2015) [13]. This includes the evaluation Framework[3] and the ontology corpus.

[2] The ranking contains ties and is not linear.

[3] ORE Framework is available at `https://github.com/andreas-steigmiller/ore-2014-competition-framework/`

Ontologies. This includes 1967 ontology collected from the ORE 2015 corpus [13]. 1920 of them are sampled from three different source corpora [4] and 47 are user submitted ontologies[5]. At first, the OWL profiles of the ontologies are checked. The process revealed that 11 user submitted ontologies do not fit to any of the standard OWL profiles. It was also impossible to load 2 other ontologies. This left us with 1954 validated ontologies, 1191 are within the OWL DL profile and the remaining 763 are OWL EL ones. Afterwards, the selected ontologies are arranged into 2 different collections, based on their profiles.

Reasoners and evaluations. To build up our advising system, we selected a representative subset of popular and efficient ontology reasoners. More precisely, we picked up the 10 best ranked reasoners [6] in both the DL and EL classification challenges of ORE 2015. Then, we assigned them to 2 groups: OWL DL and OWL EL specialised reasoners. In the first group, we can find **Konclude**, **HermiT**[7], **MORe**[8], **TrOWL**, **FaCT++**, **JFact**, **Racer** and finally **Pellet**[9]. The second group has the same reasoners as the first one, in addition to, **ELK** and **ELepHant**. Description and references to these systems could be found in [13]. By following the ORE competition processing steps, two classification challenges (DL and EL) are set up. Each challenge puts the selected reasoners into comparison when attempting to classify the ontology collection that corresponds to their group. To conduct these evaluations, we run the ORE Framework in the sequential mode on a machine equipped with an Intel Core I7, CPU running at 3.4GHz and having 32GB RAM, where 10GB were made available for each reasoner. We set the condition of 3 minutes time limit to classify an ontology by a reasoner, where only 150 seconds were allowed for reasoning and 30 seconds could be used for parsing and writing results. In the ORE Framework, the times are measured in wall clock time instead of CPU time. Figure 1 summarizes the results of the carried out challenges [10]. For each OWL profile and for every reasoner, the percentage of ontologies classified with success is illustrated, together with the failure percentage. Figure 1 details also the different cases of failure (Unexpected, Timeout and Halt). Worth to be noted, the reasoners are ordered according to their success rate. Furthermore, Figure 2 depicts the average runtime exhibited by every reasoner over the correctly classified ontologies, i.e. the success cases, and for each ontology collection. We can notice that Konclude is the most robust reasoner over the DL ontologies, while ELK outperforms all the system over the EL Ontologies. Konclude is rated the 3^{rd} on the EL classification challenge. In general, the success rates of the different reasoners are very close when considering EL ontologies, but they are quiet distinct in the case of DL ones. Indeed, there is a much important failure rate in the DL classification

[4] Available at `https://zenodo.org/record/18578#.WReUzlXyjcc`

[5] Available at `https://zenodo.org/record/50737#.WReW01Xyjcc`

[6] Reasoners are available at `https://zenodo.org/record/50738#.WRhPVVXyjcc`

[7] Specifically, it is the HermiT implementation based on OWL API 4 (HerimT-OA4).

[8] This is exactly the MOReHermiT implementation.

[9] We used the Pellet implementation based on OWL API 4 (Pellet-OA4).

[10] Evaluation results produced by ORE for the 10 reasoners are available at `https://github.com/Alaya2016/OntoClassification-Results2017/`

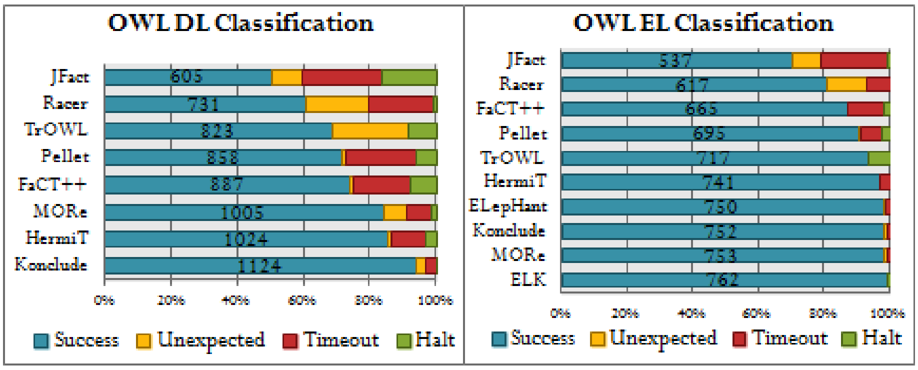

Fig. 1. Summary of reasoner evaluation results of the classification track over DL and EL respective ontology collections.

challenge. This is due to the high expressivity of the OWL 2 DL profile, comparing to the EL one. By looking to Figure 2, we can observe that systems like Hermit, which have the 2^{nd} best success rate, has also the worst reasoning average runtime over the correct cases. In overall, we can remark that reasoner ranks computed based on correctness does not completely meet their ranks based on efficiency. Given these facts, it is obvious that defining general rules to select *"best"* reasoner for any ontology is not a trivial task and may not be effective in practice.

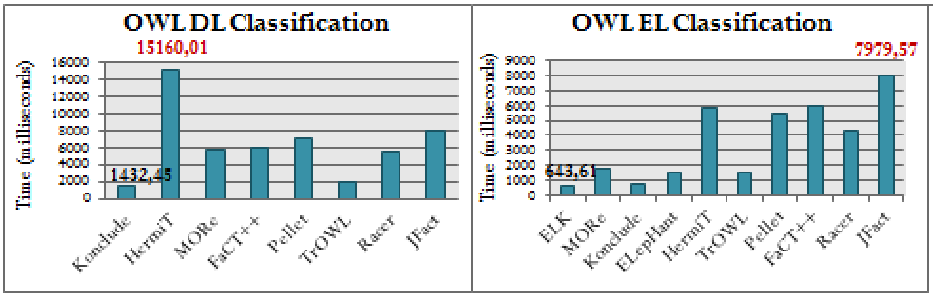

Fig. 2. Comparison of reasoner average runtime for correctly classified ontologies (success cases, time in millisconds) over the DL and the EL respective ontology collections.

Training and testing datasets. For learning evaluation purpose, we split up each ontology collection into training and testing sets. The decomposition respects, in the best possible way, the distribution of ontologies over the size bins. In overall, we selected 654 test ontologies, i.e. 391 DL and 263 EL ones. This selection is hold out to carry out the evaluations of prediction quality. The remaining 800 DL and 500 EL ontologies are gathered in the training set to build up the Multi-RakSOR predictive models. As final step, the features values of the selected ontologies are measured (c.f. Subsection 2.3). Then, the reasoner ranking and relevance vectors are computed for every ontology in the training set. Thus, OWL profile based datasets were created by incorporating the feature vectors and their corresponding relevance/rank vectors.

5 Experimental Evaluation of Multi-RakSOR

Multi-RakSOR[11] is realised with Java. It has a generic design and three main building blocks: the ontology profiler, the multi-learner and the multi-predictor components. In this paper experiments, Multi-RakSOR uses the Binary Relevance (BR) algorithm [7] as the base MLC learner and the Ensemble of Regressor Chains (ERC) algorithm [14] as the base MTR learner. Mulan[12] [17], the Java multi-label learning API, allowed the access to these state-of-art algorithms. We recall that 654 ontologies were held out to evaluate the prediction quality of Multi-RakSOR. Given a test ontology, the predicted reasoner relevance and ranking values are compared to the ideal ones. These are the actual correct relevance values and ordering of the reasoners given the ontology under examination. Afterwards, the agreement between the predicted and the ideal values are assessed using the metrics, we describe in the following. Our results are then compared against existing multi-label learning solutions.

5.1 Evaluation metrics

A two-step evaluation procedure is designed to adjudge the prediction quality of Multi-RakSOR. First, the accuracy of the predicted reasoner relevance bipartition is checked. For this kind of evaluations, the assessment metrics of binary multi-label classification models [20] are employed. In particular, the F1-Measure for each test case is computed and then, averaged over the whole test set. Similarly, the **Hamming loss** (HM-Loss) score is measured and averaged across all the test cases. This metric computes the percentage of misclassified labels. Generally speaking, a good MLC model should maximize its F1-Measure value, while minimizing its HM-Loss value. Later on, the quality of the produced rankings is assessed using 4 metrics falling in 2 categories. We already employed these metrics in [2]. The main purpose of the first metric category is to show to what extent a predicted ranking is correlated to the *ideal* one. It is composed of the average value of the generalized *Kendall Tau correlation coefficient*, denoted by *KendalTauX* and the average value of the *Spearman rank correlation coefficient*, denoted by *SpearmanRho*. The second category is made up from *information retrieval* based metrics. They examine how well we are ranking the reasoners at the top of the list, i.e. at the K^{th} position. To obtain an overall idea of precision considering the whole test set, the *Mean Average Precision* is computed and denoted by MAP@K. We put our focus on two particular values: MAP@1 and MAP@3.

[11] A demo application reproducing the evaluations of Multi-RaKSOR is available at `https://github.com/Alaya2016/Multi-RakSORDemo/`

[12] Mulan is available at `http://mulan.sourceforge.net/download.html`

5.2 Multi-label learning methods

We compare the quality of Multi-RakSOR relevance prediction against 4 well known MLC solutions [20]: (1) the neural network approach for the multi-label classification task (*BP-MLL*); (2) the Random K-Labelsets method (RAKEL); (3) the adaptive boosting algorithm for multi-label learning *(AdaBoost.MH)*; and (4) the multi-label k-Nearest Neighbor (*ML-kNN*) algorithm. In a second stage, we compare the quality of Multi-RakSOR predicted rankings to those produced by 4 *label ranking* (LR) solutions [15]: (1) the K-Nearest Neighbor approach for label ranking (*LR-kNN*); (2) the predictive clustering trees for ranking (*PCTR*); (3) the label ranking trees (*LRT*); and (4) the ranking by pairwise comparison algorithm (*RPC*). It is worth to be noted that these algorithms predict only the ranks of the target labels and do not separate them into relevant/irrelevant ones. However, they are the building blocks of the meta-reasoner[13] R_2O_2 [8]. For each learning task (MLC, MTR) and for every training dataset (DL, EL), we train the predictive models of all of the aforementioned learning solutions. Then, we assess their predictive quality over our testing datasets.

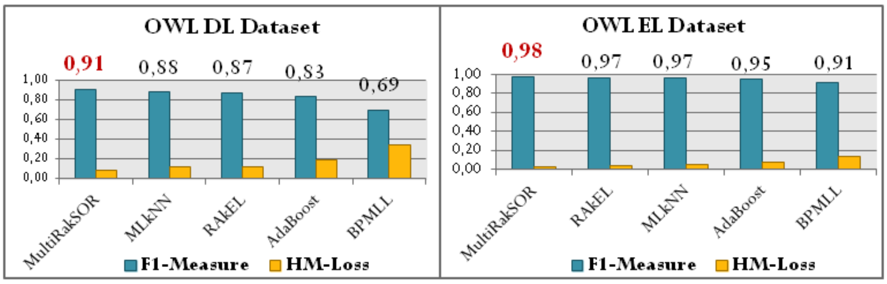

Fig. 3. Comparison summary of relevance prediction quality achieved by the examined works over each of the OWL profile based test datasets.

5.3 Relevance prediction assessment results

Figure 3 depicts the assessment results of the reasoner relevance bipartition predictions achieved by the the studied solutions. For both the DL and the EL datasets, Multi-RakSOR showed very high prediction capabilities, characterized by a score of F1-measure above 0.91 and low Hamming Loss (HM-Loss) value. It outperformed all the other MLC solutions and rated the first over the two datasets. We can also remark that predicting the relevance bipartition over EL test cases seems to be more easier than over DL cases. Actually, Multi-RakSOR has achieved a close to optimum F1-Measure. This could be explained that all of the studied reasoners has showed closer performances when classified EL ontologies. Hence, their performance are almost predictable on this kind of ontologies.

[13] It is to be considered that we didn't get access to R_2O_2 running executable. Hence, we were enable to establish any comparison with this system.

5.4 Ranking prediction assessment results

Figure 4 sketches the assessment results of the ranking quality achieved by the examined MTR methods over the DL and EL testing datasets. First, it can be noted that in both the datasets and according to the different assessment metrics, Multi-RakSOR have outperformed its base MTR leaner, the ERC algorithm. This observation pinpoints the positive impact of our proposed ranking correction step (c.f. Subsection 3.4). Accordingly, we can assert that checking the consistency of the predicted ranks w.r.t. the predicted relevance of reasoners is effective and can contribute to the overall improvement of the ranking quality. Multi-RaKSOR can identify the top most performing reasoner, regarding both the correctness and the efficiency criteria, with a precision of more than 88%, for both DL and EL ontologies. According to Kendall TauX, Multi-RaKSOR is also capable to predict the ties across the irrelevant reasoners and produce rankings that are at 94% positively correlated to the real ones.

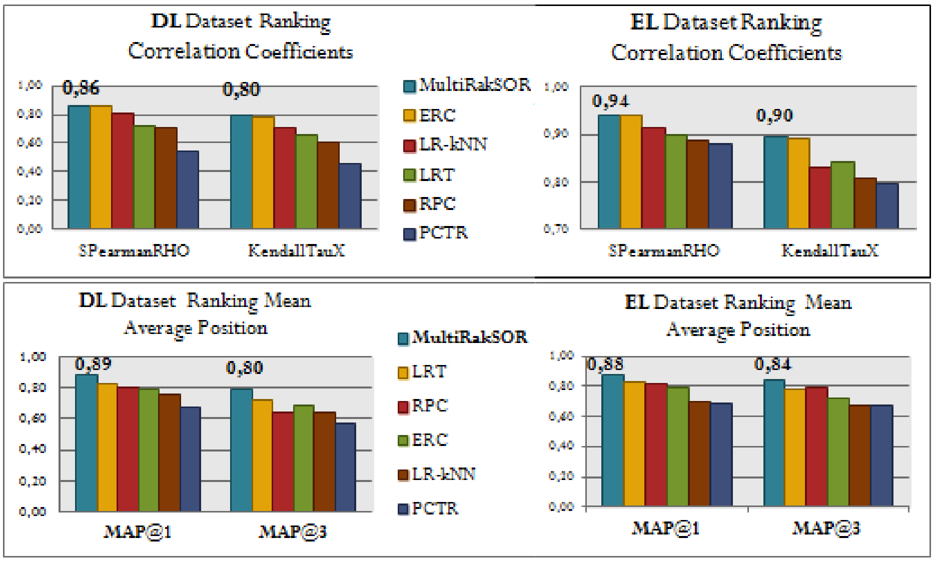

Fig. 4. Comparison summary of the ranking prediction quality achieved by the examined works over each of the OWL profile based test datasets.

More interestingly, Multi-RakSOR method have overpassed all of its counterparts, LR-KNN, RPC, PCTR and LRT, w.r.t. the different assessments measures. This result is important since these are well recognized algorithms in the field of label ranking. Besides, they are part of R_2O_2, the only existing reasoner ranking system. These empirical results could be explained by the fact that none of these algorithms is originally designed to predict a ranking which includes ties. Even that they can handle such ranking, they are not particularly optimized for. Once again, these findings show that our proposed Multi-RakSOR method adds good multi-label ranking abilities to the existing solutions. Despite the challenges, these results proves that it was worthwhile to investigate time and effort in exploring and accommodating multi-label learning techniques for better automatic ranking of ontology reasoners.

6 Experimental Evaluation of Meta-RakSOR

The main purpose of this set of evaluations is to investigate the effectiveness of building a meta-reasoner upon our multi-label ranking solution. A meta-reasoner has various common treats with SAT algorithm portfolio approach [19, 11]. The portfolio aims to take advantage of the complementarity of the algorithms by combining them. Roughly speaking, a portfolio could be built in by gathering a set of performing algorithms, together with an intelligent selector capable to decide the most performing one to any input instance. Multi-RakSOR ranking method could be seen as an intelligent selector by only considering the reasoner on the top of the ranked list. From a theoretical perspective, Multi-RakSOR could easily be evolved into a meta-reasoner. Indeed, the computing algorithms of ontology features have polynomial complexity with respect to the size of the inputs. Furthermore, the predictions are computed in constant time. In the following, we experimentally examine the worthiness of this assumption.

Reasoner	**Success**	**Failure**			**AVG.**
		#U	**#T**	**#H**	**Time**
M-RakSOR	367	12	8	4	1545.69
Konclude	366	12	11	2	1358.43
HermiT	334	3	42	12	14243.19
MORe	326	26	35	4	6428.77
FaCT++	295	4	69	23	4509.77
Pellet	287	4	84	16	2157.61
TrOWL	274	84	0	33	2157.61
Racer	245	67	75	4	4813.06
JFact	196	39	94	62	8258.99

Table 1. Results summary of the **OWL DL** classification challenge (Test Set).

Reasoner	**Success**	**Failure**			**AVG.**
		#U	**#T**	**#H**	**Time**
M-RakSOR	263	0	0	0	2166.13
ELK	262	0	0	1	635.68
Konclude	259	4	0	0	817.01
MORe	259	3	1	0	1807.03
ELepHant	258	1	4	0	1179.67
HermiT	256	0	7	0	6418.43
TrOWL	243	0	0	20	1116.69
Pellet	242	1	16	4	6081.54
FaCT++	232	0	29	2	6092.75
Racer	214	31	18	0	3994.49
JFact	186	24	52	1	5767.34

Table 2. Results summary of the **OWL EL** classification challenge (Test Set).

The upgraded version of our system is called **Meta-RakSOR**. Given an input ontology, Meta-RakSOR computes its feature vector then, predicts the ranks of the available reasoners using the approach described in Section 3. Finally, the reasoner with the lowest predicted rank value is invoked. In our experiments, we repeated this process for each EL and DL test ontology. We stored the computational time of the full prediction and classification steps. Then, we compared the Meta-RakSOR achieved results to those computed by the other reasoners (c.f. Section 4). Table 1 and Table 2 report the evaluation results respectively over DL and EL ontology test sets[14]. They report the number of success and

[14] Meta-RakSOR can handle both DL and EL profiles.

failure cases and the average runtime in milliseconds over the correctly classified ontologies (i.e. the AVG. Time column). It can be observed that Meta-RakSOR has the highest level of correctly processed ontologies for both the DL and EL ontologies. It is rated the first on both challenges. However, Meta-RakSOR has not outperformed the other reasoners in terms of classification runtime. It has the 2^{nd} lowest average runtime over correctly classified DL ontologies, just behind Konclude. However, it is placed the 6^{th} by considering the average runtime over EL ontologies. Nevertheless, for this set of ontologies, it showed better performance than known reasoners, like FaCT++, HermiT and Pellet. In overall, the achieved results are very encouraging ones. It proves the potential benefits of Meta-RakSOR, especially in terms of result correctness. Still, we assert that using a meta-reasoner for light-weighted and inexpressive ontologies is not worthwhile, since the time overhead due to the prediction steps may overpass the actual classification time. Based on this observation, Meta-RakSOR could be improved by fixing a default reasoner to be applied for *easy* cases without needing any prediction effort.

7 Conclusion

In this paper, we introduced a novel automatic ranking mechanism of ontology reasoners. It combines multi-label classification and multi-target regression techniques. It achieves both reasoner ranking and reasoner relevance prediction in a consistent way. The proposed system considers the correctness and the efficiency of reasoners in the ranking process. We studied separately OWL DL and OWL EL specific reasoners. We achieved high ranking prediction quality and outperformed existing solutions. We also examined the feasibility of employing our ranking solution as the key component of a meta-reasoner. The experimental results showed the potential of this proposal. However, it is obvious that more optimisation steps are required to improve its performances. For future work, we are intending to examine more reasoner evaluation criteria, such as the energy and memory consumption. This is a real challenge for reasoning on mobile devises. We are also planning to study different reasoning tasks, like consistency checking and realisation. Actually, our ultimate goal is to conduct different reasoner evaluation campaigns under a variety of machine, time and memory configurations and for different reasoning tasks. Once these data is gathered, a larger scale multi-criteria ranking system of ontology reasoners could be established.

References

1. Alaya, N., Yahia, S.B., Lamolle, M.: What makes ontology reasoning so arduous? Unveiling the key ontological features. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics. pp. 4:1–4:12 (2015)
2. Alaya, N., Yahia, S.B., Lamolle, M.: RakSOR: Ranking of ontology reasoners based on predicted performances. In: Proceedings of the 28th IEEE International Conference on Tools with Artificial Intelligence. pp. 1076–1083 (2016)

3. Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., Vee, E.: Comparing partial rankings. SIAM Journal of Discrete Math. 20, 628–648 (2006)
4. Gardiner, T., Tsarkov, D., Horrocks, I.: Framework for an automated comparison of description logic reasoners. In: Proceedings of the International Semantic Web Conference, USA. pp. 654–667 (2006)
5. Gonçalves, R.S., Parsia, B., Sattler, U.: Performance heterogeneity and approximate reasoning in description logic ontologies. In: Proceedings of the 11th International Conference on The Semantic Web. pp. 82–98 (2012)
6. Group, W.O.W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), available at `http://www.w3.org/TR/owl2-overview/`
7. Ioannou, M., Sakkas, G., Tsoumakas, G., Vlahavas, I.P.: Obtaining bipartitions from score vectors for multi-label classification. In: Proceedings of the 22nd International Conference on Tools with Artificial Intelligence, ICTAI. pp. 409–416. IEEE Computer Society (2010)
8. Kang, Y., Krishnaswamy, S., Li, Y.: R_2O_2: An efficient ranking-based reasoner for OWL ontologies. In: Proceedings of the 14th International Semantic Web Conference, ISWC. pp. 322–338 (2015)
9. Lee, M., Matentzoglu, N., Parsia, B., Sattler, U.: A multi-reasoner, justification-based approach to reasoner correctness. In: Proceedings of the 14th International Semantic Web Conference, ISWC, Bethlehem, USA. pp. 393–408 (2015)
10. Matentzoglu, N., Leo, J., Hudhra, V., Sattler, U., Parsia, B.: A survey of current, stand-alone OWL reasoners. In: Proceedings of the 4th International Workshop on OWL Reasoner Evaluation. pp. 68–79 (2015)
11. Oentaryo, R.J., Handoko, S.D., Lau, H.C.: Algorithm selection via ranking. In: Proceedings of Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 1826–1832 (2015)
12. Olmo, J.L., Romero, C., Gibaja, E., Ventura, S.: Improving meta-learning for algorithm selection by using multi-label classification: A case of study with educational data sets. International Journal of Computational Intelligence Systems 8(6), 1144–1164 (2015)
13. Parsia, B., Matentzoglu, N., Gonçalves, R.S., Glimm, B., Steigmiller, A.: The OWL Reasoner Evaluation (ORE) 2015 Resources, pp. 159–167. Springer, Cham (2016)
14. Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., Vlahavas, I.: Multi-target regression via input space expansion: treating targets as inputs. Machine Learning 104(1), 55–98 (2016)
15. Sun, Q., Pfahringer, B.: Pairwise meta-rules for better meta-learning-based algorithm ranking. Machine Learning 93(1), 141–161 (2013)
16. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, 2nd ed., pp. 667–685 (2010)
17. Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I.: Mulan: A java library for multi-label learning. Journal of Machine Learning Research 12, 2411–2414 (2011)
18. Wang, T.D., Parsia, B.: Ontology performance profiling and model examination: First steps. In: Proceedings of The 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC + ASWC. pp. 595–608. Korea (2007)
19. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for sat. J. Artif. Int. Res. 32, 565–606 (2008)
20. Zhang, M., Zhou, Z.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. 26(8), 1819–1837 (2014)