Predicting OWL Reasoners: Locally or Globally?

Viachaslau Sazonau, Uli Sattler, and Gavin Brown

The University of Manchester
Oxford Road, Manchester, M13 9PL, UK
{sazonauv,sattler,gbrown}@cs.manchester.ac.uk

Abstract. We propose a novel approach for performance prediction of OWL reasoners. It selects suitable, small ontology subsets, and then extrapolates reasoner's performance on them to the whole ontology. We investigate intercorrelation of ontology features using PCA. Finally, we discuss various error measures for performance prediction and compare our approach against an existing one using these measures.

1 Introduction

Although there exist many reasoners, there is no single reasoner that performs best on all given inputs. The same ontology may take just several seconds to reason over for some reasoners and require hours to process for others [5]. Recent results of the competition between 14 reasoners taking place at the OWL Reasoner Evaluation Workshop, ORE 2013,[1] show that there is no single winner in all ontology categories. The huge difference in performance of reasoners on a given input often surprises ontology designers. During ontology design and maintenance, reasoning performance may vary significantly and in surprising ways: axiom additions or removals may cause a significant increase or decrease of classification time, often in contrast to intuitive expectations [5]. Recent studies of performance variability suggest that there are ontologies which are *performance homogeneous* and others are *performance heterogeneous* for a reasoner [5]. An ontology $\mathcal{O}$ is performance heterogeneous if classification time of $\mathcal{O}' \subset \mathcal{O}$ is not linearly correlated with its relative size. The latter suggests that interaction effects come into play when certain axioms are part of the ontology and, consequently, loose their impact on reasoner's performance when those axioms are removed.

These observations make performance prediction of a reasoner on a given input challenging and often impossible even for experienced and skilled ontology engineers. However, the ability to predict performance is attractive. Firstly, reasoner developers equipped with a predictor would be able to find out which ontologies are harder for a reasoner. This can speed up tests of reasoners and facilitate new optimizations. Secondly, an estimate of classification time might allow an ontology engineer to decide whether the reasoning task is worth waiting for or not. As a practical example, the progress bar of an ontology editor, such as Protégé 4,[2] could be made more reliable

[1] http://ore2013.cs.manchester.ac.uk/competition/
[2] http://protege.stanford.edu/

with a good performance predictor. It could also supply auto-picking the fastest reasoner for a given ontology from the list of available reasoners.

The contributions of this work are three-fold. Firstly, we suggest an approach to analyse intercorrelation of ontology features via Principal Component Analysis, PCA, given a corpus of ontologies, and observe that ontology features are highly intercorrelated, so that 57 features can be "faithfully" replaced by one or two features. Secondly, we introduce a new method for performance prediction which is competitive with existing performance predictors. Thirdly, we discuss different prediction accuracy measures.

2 Preliminaries

We introduce the machine learning concepts that we will refer to [1]. Assume there is a finite collection of *objects* (here ontologies). Each object from the collection has its own *label* which is not necessarily unique. Hence, the objects can be separated into different categories, or classes, based on their labels. Then, a new object of the same type appears and the task is to predict its label, or to classify it into an appropriate category. This is called the *machine learning classification* problem. In order to make a prediction for an object, some information has to be extracted to describe its properties. It is commonly represented as a multidimensional vector. The vector variables are called *features* and the vector is called a *feature vector*. Feature vectors with known and presumably unknown labels are called *training* and *testing examples*, respectively. A set of N feature vectors of dimensionality d is commonly written as the $N \times d$ matrix: $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id}),\ i = 1 \ldots N$, is the i-th feature vector in the set and T stands for the matrix transposition. Let the label be $y_i,\ i = 1 \ldots N$ and the vector of all labels $\mathbf{y}$. Thus, the data set is $[\mathbf{X} \mid \mathbf{y}^T]$.

Training examples with labels are used to construct a machine learning model. This process is called *learning*, or training through "supervision". The set of related methods is called *supervised machine learning*. In essence, a model is a data structure, for instance, a graph or a probability distribution, which is built from the data set. Once the model is trained, it can be used to predict a label of an unknown object based on its features. More specifically, the model implicitly compares the unknown object with known objects, identifies similar known objects, and uses their labels to make a prediction.

There exist recent attempts to apply supervised machine learning to performance prediction of a reasoner on a given ontology. In [10], the authors define 27 ontology features, or *metrics*, of 4 types: ontology level (*ONT*), class hierarchy level (*CH*), anonymous class expressions (*ANE*) and properties (*PRO*). A feature vector $\mathbf{x}_i$ is extracted from each ontology $\mathcal{O}_i,\ i = 1, ..., N$ in the corpus. It is assigned a label $y_i = b(\mathcal{O}_i, \mathcal{R}),\ i = 1 \ldots N$, which is the *performance bin* of $\mathcal{O}_i$ given a reasoner $\mathcal{R}$. Kang et al. attempt to predict the performance bin $\hat{y} = \hat{b}(\mathcal{O}, \mathcal{R})$ for an ontology $\mathcal{O}$ based on its feature vector $\mathbf{x}$, or "profile".[3] To do so, they train a model using feature vectors $\mathbf{X}$ of *all training ontologies* in the corpus with their *labels* $\mathbf{y} = (y_1, y_2, ..., y_N)$ per reasoner, i.e. the "global" corpus view $[\mathbf{X} \mid \mathbf{y}^T]$. We call such a technique a *global approach*. For basics of predicting algorithm runtime, we refer the interested reader to a survey of machine learning methods for three NP-complete problems [8].

[3] Not to be confused with OWL profiles such as OWL 2 EL

3 Feature Analysis and Dimensionality Reduction

As we will see, it is considerably hard or even impossible to define the "ideal" set of ontology features for performance prediction of reasoners. Nonetheless, we need to take features of different scales into account, e.g. entropy of the graph, EOG, and size of vocabulary, SOV,[4] used by Kang et al. Competing approaches exist in machine learning to address this problem [18].

We skip some features from Kang et al. and add our own features [15]. Features based on the inferred class hierarchy are not included because this requires classification and, thus, makes little sense for performance prediction. Overall, we consider $d = 57$ ontology features including 33 new features. Intuitively, more features should allow us to capture more information about the object. However, it turns out that a smaller set of features can produce a more accurate model in practice [6]. It can be achieved either by eliminating weakly useful features from the original set, hence, *selecting* features, or by transforming the features into a lower dimensional space with minimal loss of information, known as *constructing* features [6].

One of the simple feature selection approaches is based on measuring Pearson's correlation coefficient [6] between labels and features in the data set. However, in order to understand correlations between subsets of features, one would have to consider the correlation coefficients for all subsets, which is clearly infeasible. A common feature construction method is *Principal Component Analysis*, PCA [9]. PCA projects feature vectors of the data set into a new coordinate system, maximizing the variances in the projection space. In summary, PCA helps to identify the presence of *intercorrelation* of features and provides a new, smaller set of uncorrelated features.

4 Experimental Setup

In this work, we investigate three commonly used reasoners: Hermit 1.3.8 [14], Pellet 2.3.0 [16], and JFact 1.0.0, a Java version of FaCT++ [17]. We have chosen the NCBO BioPortal[5] as an ontology corpus for experiments since it is a natural set of ontologies primarily maintained by biomedical domain experts. In general, a corpus of ontologies should be selected carefully because a significant bias towards easy ontologies may cause misleading generalizations. Although dealing with such bias is usually difficult due to the lack of hard ontologies, we need to take this into account.

We exclude empty ontologies, duplicates, multiple versions of the same ontology, and ontologies with loading errors. As a result, we obtain the BioPortal corpus of 357 ontologies of various sizes. In order to confirm the results of PCA experiments (and only for this purpose), we examine an additional corpus, the Tones Ontology Repository.[6] We filter it out via exactly the same procedure. As a result, we obtain the Tones corpus of 196 ontologies. We measure the classification time of each reasoner on each ontology in BioPortal. We run a reasoner 20 times on each ontology and record the average execution time for this ontology. The ontology loading time is not included. A timeout

[4] http://www.csse.monash.edu/~yli/metrics_perf/

[5] http://bioportal.bioontology.org/

[6] http://owl.cs.manchester.ac.uk/repository/

of 1,000 seconds is used for all reasoners. All experiments are executed on the same machine. The operating system is Windows 7, 64-bit. CPU is Intel Core i3-2330M, 2 cores, 2.2 GHz. RAM is DDR2, 4 GB. All algorithms are implemented in Java, JDK 1.7, using OWL API 3.4.3.

In addition to the performance bins specified by Kang et al., we add one more bin E in order to distinguish the hardest ontologies with classification times longer than 1,000 seconds. We also include all ontologies with the classification time less than one second in bin $A \in (0s, 1s)$. Thus, we consider the following 5 bins: $A \in (0s, 1s)$, $B \in [1s, 10s)$, $C \in [10s, 100s)$, $D \in [100s, 1,000s)$, $E \in [1,000s, +\infty)$. The performance statistics on BioPortal is shown in Table 1, where the "Exc" bin means that a reasoner raises an exception on those ontologies and does not finish its job properly.

$\mathcal{R}$	**Bins**					**Exc**
	A	B	C	D	E	
JFact	250	50	16	11	23	7
Hermit	222	53	21	13	16	32
Pellet	247	48	21	11	28	2

Table 1: Performance statistics on BioPortal

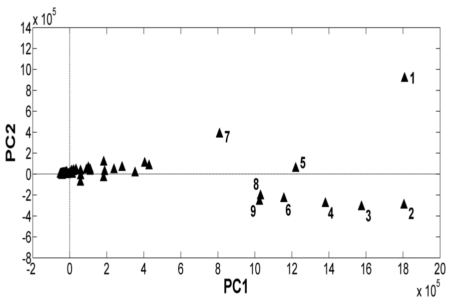
5 Intercorrelation of Ontology Features and Ontology Projections

Given a data set $[\mathbf{X} \mid \mathbf{y}^T]$, PCA deals only with a set of feature vectors $\mathbf{X}$ and *ignores* labels $\mathbf{y}$. It produces a set of PCs, PC_j, with respective variances, λ_j. We apply PCA to BioPortal and obtain a set of PCs, PC_j^B. To check our findings, we also apply PCA to Tones, which induces another set of PCs, PC_j^T, and compare the results. We use PC_j^X to denote both PC_j^B and PC_j^T. As a result, we found out that the first 3 out of 57 components explain ≈99% of all variance for both data sets. Hence, we can hypothesize that the original 57 features are *highly intercorrelated*. Another important thing to notice is that PC_1^B explains $\approx 87\%$ and PC_1^T explains $\approx 96\%$ of total variance for BioPortal and Tones, respectively.

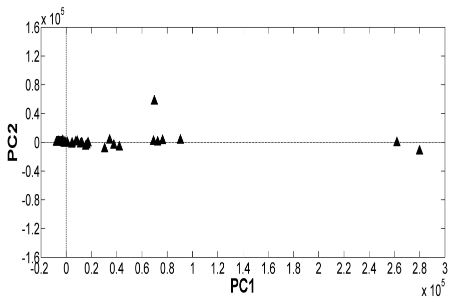
An additional advantage of PCA is that it allows us to visualise a data set in a lower dimensional space. Since just few components have significant variances, we can reasonably visualise the corpora in two dimensional space (PC_1^X, PC_2^X) and explore their spread in the projection space. This way, we preserve $\approx 94\%$ of total variance for BioPortal and $\approx 98\%$ of total variance for Tones. As Figures 1a, 1b show, ontologies of both corpora are mainly spread along the first coordinate with smaller deviations along the second coordinate and even smaller along others.

BioPortal has more outliers, i.e. ontologies which differ from the main cluster of similar ontologies. There are 9 evident outliers out of 357 ontologies. They include 9 out of 10 biggest ontologies in the corpus. It is interesting to examine classification times of these outliers for given reasoners, see Table 2. We use unique identifiers of ontologies for Figure 1a and Table 2. We can see from Tables 1 and 2 that the outliers are clearly among the computationally hardest ontologies for all three reasoners. It is important to note that these outliers have been identified *without any performance measurements*. None of these outliers are the heterogeneous ontologies from [5].

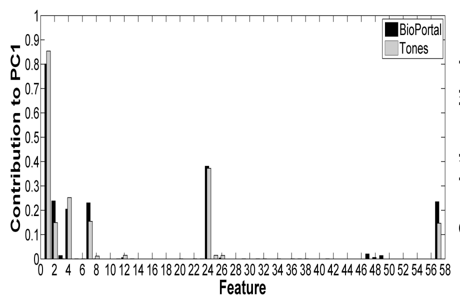
In addition, observing such an efficient representation of the total variance by PC_1^X in Figures 1a, 1b, a reasonable question is how the new features PC_1^B, PC_1^T are re-



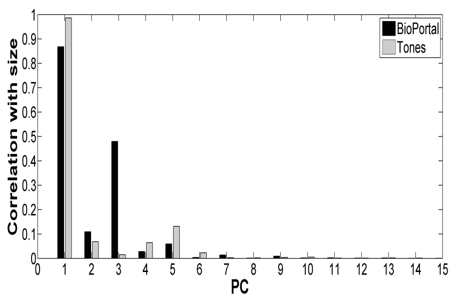
(a) BioPortal: Ontology projections using PC_1^B and PC_2^B



(b) Tones: Ontology projections using PC_1^T and PC_2^T



(c) Contribution coefficients of original features to PC_1^X



(d) Correlation between each PC_i^X and ontology size (correlation ≈ 0 for PC_{16}^X-PC_{57}^X)

Fig. 1: Comparing BioPortal (B) and Tones (T) using PCA, $X \in \{B, T\}$

lated to the original features. We examine the (linear) coefficients with which each of the original features contributes to PC_1^X, see Figure 1c. These coefficients are respective components of the eigenvector with the largest eigenvalue. Figure 1c shows that there are only 6 original features, see Table 3, that contribute significantly to PC_1^X for both corpora (see [15] for descriptions of the features). Two most contributing features are *#ClassNameOccurrences* and *#SubClassOfAxioms*. The first one counts the number of all occurrences of named classes without consideration whether there exist equally named occurrences or not. Table 3 shows that *#ClassNameOccurrences* contributes to the first component much more than *#ClassNames* which is the number of (unique) class names. The feature *#SubClassOfAxioms* counts the number of axioms that explicitly encode subsumption relationships between classes in the ontology. These features are anticipated to be relevant for ontology classification performance. What is not expected, however, is that these 6 features are the only contributing features with similar values for both corpora. This implies that PC_1^B is *approximately equal* to PC_1^T.

Exploring feature vectors, we observe that these 6 identified features significantly correlate with ontology size.[7] Hence, one can be interested how PC_1^X is affected by it. To measure this, we calculate the correlation coefficient between the first component and ontology size. It turns out that the first component correlates surprisingly well with

[7] In the following, ontology size is the number of axioms

ID	**Acr.**	**Size**, #axioms	**DL**	$\mathcal{CT}$, sec		
				JFa	**Her**	**Pel**
1	NCBI	847,755	AL	t	t	t
2	GAZ	652,361	ALE+	206	t	318
3	CCO	624,708	SRI	t	99	t
4	GEXO	549,132	SRI	t	t	t
5	Biom.	660,188	SRIF	t	t	t
6	REXO	463,799	SRI	t	t	t
7	RH-M.	403,210	AL	t	53	26
8	RETO	415,551	SRI	t	t	t
9	CPO	379,734	SR	t	t	t

Table 2: BioPortal outliers revealed by PCA and their classification times (timeout t=1,000 s; DL by OWL API)

ID	**Feature**	**Contrib.**	
		B	T
1	#ClassNameOccurrences	0.80	0.85
24	#SubClassOfAxioms	0.38	0.37
4	#ClassNames	0.21	0.25
2	#ObjectPropertyName-Occurrences	0.24	0.15
57	parsingDepth	0.24	0.15
7	#ObjectSomeValuesFrom	0.23	0.15

Table 3: Six features with the highest contributions to $PC_1^X, X \in \{B, T\}$

ontology size and no other components do, see Figure 1d. As a consequence, one can ask: is ontology size a good performance indicator? Can we predict performance better with 57 features or with size alone? We will answer these questions below.

6 A Local Approach to Performance Prediction

Our "local" approach predicts performance for the whole ontology, given the data gathered from its *samples*. Each sample is an ontology subset, ideally small, and comes with a classification time and a size. We call such a technique a *local approach*. A reasonable way to construct samples is to represent an ontology as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each vertex $v \in \mathcal{V}$ is labelled with some set of axioms, called a *clump*. Edges $\mathcal{E}$ represent some relationships between clumps. From such a representation we can assemble sets of clumps in many possible ways and use these sets as samples for performance prediction.

We introduce *growing-based sampling*. Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the process of sampling begins from a single atom. A next sample is constructed by extending the current sample with a new atom, see Algorithm 1. A sensible way of choosing a next atom can cause significant increases of the classification time of a sample. Obviously, the quality of following performance prediction depends on our notion of a clump, relations between clumps, and the assembling mechanism used. For example, a trivial way is to take small arbitrary ontology subsets as samples without any relations between them. Yet, they are not suitable performance indicators.

A more informed way to define clumps is Atomic Decomposition, AD [3]. In this case, clumps are atoms $a \in AD_{\mathcal{O}}$ which are pairwise disjoint sets of axioms and $\mathcal{O} = \dot{\bigcup}_{a \in AD_{\mathcal{O}}} a$. Then, an ontology is represented as a directed *acyclic* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_a | a \in AD_{\mathcal{O}}\}$, $(v_a, v_b) \in \mathcal{E}$ if $a \prec b$ with $\prec$ the dependency relation from [3]. For $v \in \mathcal{V}$ we use $at(v) \subseteq \mathcal{O}$ to denote axioms in the atom associated with v, i.e. $at(v_a) = a$.

Algorithm 1 produces a set of samples with their performance measurements for an ontology. The relative size limits $\langle c_j \rangle_{j=1}^{n}$ determine the number of samples. New

Algorithm 1 Growing-based sampling

```
1: Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ a graph-based representation of an ontology $\mathcal{O}$, $\mathcal{R}$ a reasoner, $\langle c_j \rangle_{j=1}^{n}$
   relative size limits in increasing order, $m$ an optional parameter
2: Output: $\Omega$ a list of $n$ samples with time measurements by $\mathcal{R}$
3: $\mathcal{S} \leftarrow \emptyset; V \leftarrow \emptyset; H \leftarrow \emptyset; v \leftarrow random(\mathcal{V})$
4: for each $j$ from 1 to $n$ do
5:     while $|\mathcal{S}|/|\mathcal{O}| < c_j$ do
6:         $V \leftarrow V \cup \{v\}; \mathcal{S} \leftarrow \mathcal{S} \cup at(v)$
7:         append $\mathcal{E}(v) \setminus (H \cup V)$ to $H$; $v \leftarrow nextClump(V, H?, m?)$
8:     end while
9:     $\mathcal{CT}(\mathcal{S}, \mathcal{R}) \leftarrow measure(\mathcal{S}, \mathcal{R})$
10:    append $(\mathcal{S}, \mathcal{CT}(\mathcal{S}, \mathcal{R}))$ to $\Omega$
11: end for
12: return $\Omega$
```

atoms are added to a sample $\mathcal{S}$ until a size limit c_j is exceeded. At that moment, the classification time $\mathcal{CT}(\mathcal{S}, \mathcal{R})$ is measured. After that, growing of the sample continues without performance measurements until the next size limit c_{j+1} is exceeded.

The subroutine *nextClump*$(V, H?, m?)$ specifies the behaviour of Algorithm 1. It has one mandatory, V, and two optional, H and m, input arguments. A version of the subroutine *nextClump*() is determined by these input parameters. We use three different strategies of growing-based sampling [15]:

1. *nextClump*(V): select randomly a new atom from the remaining set of atoms;
2. *nextClump*(V, H): select randomly an atom that is directly related to an atom in V;
3. *nextClump*(V, H, m): select randomly an atom that is directly related to an atom in V amongst m most recently added atoms.

We repeat Algorithm 1 r times and gather all $q = rn$ samples in a single suite $\Omega = \langle \mathcal{S}_j, y_j \rangle_{j=1}^{q}$ for an ontology $\mathcal{O}$, where $y_j = \mathcal{CT}(\mathcal{S}_j, \mathcal{R}), j = 1 \dots q$. This local data is then used to make predictions for the whole ontology via regression analysis. Based on the the results of our feature analysis, we assume that the regression function f can be a simple polynomial of $x = |\mathcal{S}|$:

$$f(x, \beta) = \beta_0 + \sum_{l=1}^{L} \beta_l x^l. \tag{1}$$

Popular methods for estimating the unknown parameters β are *least squares* and *non-linear least squares* [12]. Given a performance prediction task, we can assign $\beta_0 = 0$ because $f(0, \beta) = 0$, i.e. the classification time of an empty ontology is zero. After that, once the regression function f is fitted to the local data by estimating remaining parameters β, it can be used to predict the performance via extrapolation $\hat{y} = f(|\mathcal{O}|, \beta)$, given the size $|\mathcal{O}|$ of an ontology $\mathcal{O}$.

Moreover, we suggest to estimate the polynomial degree L for each ontology separately by evaluating how well the model fits the samples. We choose a polynomial of degree L_{min} as a model if it has a minimal residual sum of squares [7], RSS, once the parameters β ($\beta_0 = 0$) are fitted to the samples:

$$L_{min} = arg\ \min_L\ RSS(L), \text{where } RSS(L) = \sum_{j=1}^{q} \left(y_j - \textstyle\sum_{l=1}^{L} \beta_l x_j^l \right)^2 . \quad (2)$$

Thus, where a global approach gathers data from many ontologies and interpolates for unseen ontologies, a local approach collects data only from part of a given ontology and extrapolates to its full size. In contrast to a global approach, it does not rely on the global view of a corpus to exploit similarities/dissimilarities between ontologies and it uses no features except sample sizes.

7 Prediction Accuracy Measures

We aim to test a global and local performance predictor and compare them. Consequently, we need a procedure to evaluate a performance predictor. The first possibility is based on performance bins $b(\mathcal{O}_i, \mathcal{R}),\ i = 1 \dots M$. The model predicts a bin $\hat{b}(\mathcal{O}_i, \mathcal{R})$ for each ontology from the testing set. Whenever it predicts a wrong bin for an ontology, i.e. $\hat{b}(\mathcal{O}_i, \mathcal{R}) \neq b(\mathcal{O}_i, \mathcal{R})$, this is counted as an error. Once the model is tested on all ontologies, the *average classification error*, ACE, is calculated:

$$ACE(\mathcal{R}) = 1 - \frac{1}{M} \sum_{i=1}^{M} \delta(\hat{b}(\mathcal{O}_i, \mathcal{R}), b(\mathcal{O}_i, \mathcal{R})). \quad (3)$$

where $\delta(\hat{b}, b)$ is the Kronecker delta. The classification accuracy is used by Kang et al. to evaluate a model. In fact, it equals $1 - ACE(\mathcal{R})$. However, ACE and, consequently, the classification accuracy have their disadvantages. Firstly, they depend on binning which may cause incorrect conclusions while assessing a model. Secondly, both measures count all wrong predictions equally. For example, predicting bin 5 and bin 2 for actual bin 1 are equally wrong, regardless of the huge difference of the errors.

Consequently, when predicting performance bins, one can consider a second, more precise than ACE way to evaluate the model - the *confusion matrix* [4]. The confusion matrix is especially useful for evaluating the model on the unbalanced data set, i.e. if the distribution of examples against labels is uneven. As Table 1 shows, our data set is, in fact, significantly biased to label A. In contrast to ACE, the confusion matrix allows us to investigate errors the model makes while predicting certain labels (performance bins here). It essentially counts the number of ontologies of actual bin b which are classified to bin $\hat{b}$ for each possible pair $(b, \hat{b})$ under $\{A, B, C, D, E\}$.

A third way to assess a model is to measure the raw differences between the actual and predicted classification time for each testing ontology. The *mean absolute error*, MAE, is a common way to measure the differences between actual and predicted values:

$$MAE(\mathcal{R}) = \frac{1}{M} \sum_{i=1}^{M} |\hat{y}(\mathcal{O}_i, \mathcal{R}) - \mathcal{CT}(\mathcal{O}_i, \mathcal{R})|, \quad (4)$$

where $\hat{y}(\mathcal{O}_i, \mathcal{R})$ is the predicted classification time of a reasoner $\mathcal{R}$ on an ontology $\mathcal{O}_i$, and $\mathcal{CT}(\mathcal{O}_i, \mathcal{R})$ is the actual classification time. This measure avoids ontology binning whose usefulness for evaluating performance predictors is an open question.

However, MAE has its own disadvantage because the hardest ontologies, which are rare, can contribute more to MAE than all remaining ontologies together. In this case, MAE measures the model's fitness to hard ontologies but not to the data set. This is even worse for another common measure, the *mean squared error* (MSE), which amplifies the differences. Thus, all aforementioned measures have their disadvantages, no single measure is enough for predictor evaluation, and we should rather consider them all.

8 Performance Prediction Experiments and Results

We use the *k*-NN classifier [2] as a global approach model. The choice of a machine learning model is not critical here because we test overall utility of global approaches for reasoning performance prediction and plan to compare these results with the local approach. The *k*-NN classifier is a reasonable choice here because, firstly, it allows to obtain sufficiently good predictions for many data sets. Secondly, it has only one input parameter *k* which makes its tuning easy and analysis of its predictions transparent. Thirdly, it can be used to predict both performance bins and classification times.

A common approach for evaluating a supervised machine learning model using the same set of examples is *cross-validation* [11]. In order to obtain the best prediction results, the parameter k needs to be tuned for each reasoner. Observing that our data set consists of at most $N = 357$ examples, we apply *leave-one-out cross-validation* [11], LOOCV. An optimal k is estimated via LOOCV on the first 80% of the data set, and the best model with tuned k is evaluated on the remaining 20%. This procedure is repeated 1,000 times per reasoner, each time shuffling the data set randomly.

For the sake of simplicity, we denote the cross-validation *generalization error*, which is either average ACE or average MAE over all best models, by ACE or MAE, respectively. We conduct two performance prediction experiments for the global approach. In the first one, we extract all 57 features from each ontology in BioPortal and train a model, *Glob.57f*, using the resulting data set. In the second one, we use an ontology size as the only feature to train a model, *Glob.1f*. The latter is based on our earlier observations and PCA results. While constructing the confusion matrix, the parameter k is tuned for each reasoner via LOOCV on the whole data set.

For the local approach, we construct samples for an ontology via Algorithm 1 with the subroutine *nextClump*(V, H, m). This way produces the most performance indicative samples via building a specific path through the ontology graph [15]. Algorithm 1 is executed $r = 10$ times per ontology with the following parameters: $m = 1$, $\langle c_j \rangle_{j=1}^{n} = \langle 0.01, 0.02, ..., 0.10 \rangle$ ($n = 10$). Hence, the algorithm constructs a suite of $q = 100$ samples of relative size $\leq 10\%$ for each ontology. Here, we consider the following trade-off: higher size limits should give more accurate predictions but take longer to be classified. We assume that $f(x)$ is a polynomial of degree $L \leq 2$, considering the observations in [5]. Prediction results of the global approach (Glob.57f, Glob.1f) with respective error deviations and local approach (Local) are gathered in Table 4. We do not report error deviations for a local approach because its predictions are deterministic. Ontologies that cause an exception for a reasoner, see Table 1, are excluded from the data set for that reasoner.

$\mathcal{R}$	ACE		
	Glob.57f	**Glob.1f**	**Local**
JFa	0.230 (0.048)	0.211 (0.045)	**0.169**
Her	0.231 (0.053)	0.237 (0.047)	**0.224**
Pel	0.301 (0.048)	0.301 (0.048)	**0.250**
	MAE, sec		
JFa	61 (24)	69 (18)	**43**
Her	70 (25)	89 (23)	**42**
Pel	106 (31)	127 (27)	**69**

Table 4: Comparison of global and local approaches on BioPortal (with error deviations)

Real bin	**Predicted bin**				
	A	B	C	D	E
A	**245/242/232**	4/7/16	0/0/1	0/0/0	1/1/1
B	31/23/15	**19/27/28**	0/0/7	0/0/0	0/0/0
C	2/0/0	9/9/0	**3/2/11**	0/0/3	2/5/2
D	1/1/0	2/2/0	3/4/1	**0/0/6**	5/4/4
E	2/2/1	5/4/1	4/5/3	0/3/4	**12/9/14**

Table 5: Confusion matrix for JFact on BioPortal (Glob.57f/Glob.1f/Local)

As Table 4 shows, the local approach delivers better prediction accuracy than the global one in all cases, for both ACE and MAE. In addition, one can notice that Glob.1f outperforms Glob.57f for JFact, shows the same error for Pellet, and slightly loses for Hermit, if we compare via ACE. However, if we compare via MAE, Glob.57f slightly outperforms Glob.1f for all reasoners. Despite that, Glob.1f has a more stable model with respect to error deviations. Hence, extracting 57 ontology features makes little sense, especially for bin predictions. One can notice that Pellet is harder to predict than others on BioPortal for all approaches via either ACE or MAE. It is also worth mentioning that the local approach chooses the polynomial of degree $L = 1$ as the best prediction model in 72%, 78%, 76% for JFact, Hermit, and Pellet, respectively. Hence, all reasoners are mostly *linearly* predictable. Table 5 shows the confusion matrix of JFact predictions (the confusion matrices for Hermit and Pellet are similar). Glob.57f and Glob.1f are slightly more accurate than the local approach on the easiest bin A, the latter is more correct on all harder bins B, C, D, E; especially on bins C, D.

$\mathcal{R}$	ACE		
	Glob.57f	**Glob.1f**	**Local**
JFa	0.533 (0.298)	0.375 (0.112)	**0.260**
Her	0.476 (0.245)	0.407 (0.096)	**0.379**
Pel	0.506 (0.107)	0.565 (0.108)	**0.481**
	MAE, sec		
JFa	201 (70)	237 (56)	**141**
Her	198 (68)	260 (53)	**120**
Pel	337 (74)	323 (73)	**227**

Table 6: Comparison of global and local approaches on BioPortal without easy ontologies (with error deviations)

Real bin	**Predicted bin**			
	B	C	D	E
B	**46/50/43**	4/0/7	0/0/0	0/0/0
C	7/10/0	**8/6/11**	0/0/3	1/0/2
D	4/3/0	2/2/1	**0/0/6**	5/6/4
E	6/6/2	2/6/3	2/0/4	**13/11/14**

Table 7: Confusion matrix for JFact on BioPortal without easy ontologies (Glob.57f/Glob.1f/Local)

Considering the strong bias of the corpus, the question arises how performance predictors behave if easy ontologies are excluded. Therefore, for each reasoner, we remove ontologies of bin $A \in (0s, 1s)$ and repeat the same chain of experiments (including training) on the remaining 100, 103, 108 “hard” ontologies, see Table 6. Hard ontologies cause extremely higher errors of performance predictions for all approaches. Glob.1f shows surprisingly more accurate and stable predictions than Glob.57f via ACE for JFact and Hermit but fails to do so for Pellet. In contrast, Glob.57f outperforms Glob.1f via MAE for JFact and Hermit (though Glob.1f is more stable) but fails to do so for Pellet. At last, the local approach outperforms, sometimes significantly, the global one in all cases, for both ACE and MAE. The confusion matrix in Table 7 shows that Glob.57f and Glob.1f are again more accurate on the easiest bin B, while the local approach is more accurate on harder bins C, D, E. We also observe that, if we exclude the heterogeneous ontologies found in [5] from the hard ontologies and repeat the performance prediction experiments (including training), then Table 6 changes just slightly. Hence, heterogeneous ontologies do not cause unusually high errors of predictions.

9 Discussion of Results and Future Work

To conclude, we have observed that performance prediction of reasoners is a hard and interesting problem. In this work, we propose a novel approach, called local, to solve it. It is based on carefully selecting ontology samples, ideally small, and then using them to predict reasoning performance on the full ontology. The AD, as an ontology graph, is useful to construct such samples: samples of sizes $\leq 10\%$ allow us to obtain good predictions via simple extrapolations. In comparison to a global approach, a local approach gives more accurate performance predictions, does not rely on an ontology corpus and, therefore, cannot be biased by it. Moreover, a local approach reveals interesting information about reasoners' behaviour: linear/nonlinear predictability on the corpus. However, it requires reasoning over samples to make a prediction.

Although selecting an unbiased corpus remains a difficult problem [13], we definitely need to acknowledge the bias and interpret observations correctly. Moreover, the error measures for evaluating predictors should be chosen carefully because there is no single best measure. A reasonable way is to use several measures evaluating different aspects of prediction accuracy.

One of our main findings is that, for both BioPortal and Tones, multiple ontology features are efficiently represented by ontology size. Predictions using ontology size alone are comparable to predictions using all 57 ontology features. Therefore, ontology features for performance prediction should be defined and used cautiously. Firstly, we should not mix features of different scales in the feature vector because low-scale features can be "hidden" by others. Nonetheless, if these features are informative, suitable techniques, able to handle different scales, must be used. Secondly, we have to investigate intercorrelation of features because the features may turn out to carry little useful information and even degrade the accuracy.

As future work, we consider local approach improvements. The maximal sample size can be estimated for each ontology separately because 10% samples are not indicative enough to make good predictions for some ontologies. In addition, there may exist more suitable regression models for extrapolation. One can also study appropriate methods to cope with ontology features of different scales with respect to performance prediction, e.g. ensemble learning. Finally, we can work on assembling "good" corpora using dimensionality reduction techniques such as PCA.

References

1. Alpaydin, E.: Introduction to Machine Learning. MIT, 2nd edn. (2010)
2. Cover, T., Hart, P.: Nearest neighbor pattern classification. Information Theory, IEEE Transactions on 13(1), 21–27 (1967)
3. Del Vescovo, C.: The modular structure of an ontology: Atomic Decomposition and its applications. Ph.D. thesis, School of Computer Science, University of Manchester (2013)
4. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27(8), 861 – 874 (2006)
5. Gonçalves, R.S., Parsia, B., Sattler, U.: Performance heterogeneity and approximate reasoning in Description Logic ontologies. In: Proc. of ISWC-12. LNCS, vol. 7649, pp. 82–98 (2012)
6. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. of Mach. Learning Res. 3, 1157–1182 (Mar 2003)
7. Hastie, T., Tibshirani, R., Friedman, J.J.H.: The elements of statistical learning, vol. 1. Springer New York (2001)
8. Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K.: Algorithm runtime prediction: methods & evaluation. Artificial Intelligence 206, 79–111 (2014)
9. Jolliffe, I.T.: Principal component analysis. Springer, second edn. (Oct 2002)
10. Kang, Y.B., Li, Y.F., Krishnaswamy, S.: Predicting reasoning performance using ontology metrics. In: Proc. of ISWC-12. LNCS, vol. 7649, pp. 198–214 (2012)
11. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. pp. 1137–1143. Morgan Kaufmann (1995)
12. Lawson, C.L., Hanson, R.J.: Solving least squares problems. 3 edn. (1995)
13. Matentzoglu, N., Bail, S., Parsia, B.: A corpus of OWL DL ontologies. In: Eiter, T., Glimm, B., Kazakov, Y., Krötzsch, M. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 1014, pp. 829–841. CEUR-WS.org (2013)
14. Motik, B., Shearer, R., Horrocks, I.: A hypertableau calculus for $\mathcal{SHIQ}$. In: Proc. of DL-07. pp. 419–426. Bozen/Bolzano University Press (2007)
15. Sazonau, V.: Performance prediction of OWL reasoners. Master's thesis, School of Computer Science, University of Manchester (2013)
16. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. J. of Web Semantics 5(2), 51–53 (2007)
17. Tsarkov, D., Horrocks, I.: FaCT++ Description Logic reasoner: system description. In: Proc. of IJCAR-06. LNCS, vol. 4130, pp. 292–297. Springer-Verlag (2006)
18. Tuv, E., Borisov, A., Torkkola, K.: Best subset feature selection for massive mixed-type problems. In: Proc. of IDEAL-06. pp. 1048–1056. Springer-Verlag (2006)