Adding data mining support to SPARQL via statistical relational learning methods

Kiefer, C ; Bernstein, A ; Locher, A

DOI: https://doi.org/10.1007/978-3-540-68234-9_36

Adding Data Mining Support to SPARQL via Statistical Relational Learning Methods

Christoph Kiefer, Abraham Bernstein, and André Locher

Department of Informatics, University of Zurich, Switzerland
{kiefer,bernstein}@ifi.uzh.ch, andre@outerlimits.ch

Abstract. Exploiting the complex structure of relational data enables to build better models by taking into account the additional information provided by the links between objects. We extend this idea to the Semantic Web by introducing our novel SPARQL-ML approach to perform data mining for Semantic Web data. Our approach is based on traditional SPARQL and statistical relational learning methods, such as Relational Probability Trees and Relational Bayesian Classifiers.
We analyze our approach thoroughly conducting three sets of experiments on synthetic as well as real-world data sets. Our analytical results show that our approach can be used for any Semantic Web data set to perform instance-based learning and classification. A comparison to kernel methods used in Support Vector Machines shows that our approach is superior in terms of classification accuracy.

1 Introduction

The success of statistics-based techniques in almost every area of artificial intelligence and in practical applications on the Web challenges the traditional logic-based approach of the Semantic Web. We believe that we should treat statistical inference techniques as a complement to the existing Semantic Web infrastructure. Consequently, a big challenge for Semantic Web research is not if, but how to extend the existing Semantic Web techniques with statistical learning and inferencing capabilities. In this paper we (1) argue that the large and continuously growing amount of *interlinked* Semantic Web data is a perfect match for *statistical relational learning (SRL) methods* due to their focus on relations between objects in addition to features/attributes of objects of traditional, propositional data mining techniques; and (2) show two concrete Semantic Web research areas, where machine learning support is useful: service classification [6] and semantic data prediction [1]. Moreover, we think that the fact that companies such as Microsoft and Oracle have recently added data mining extensions to their relational database management systems underscores their importance, and calls for a similar solution for RDF stores and SPARQL respectively.

To support the integration of traditional Semantic Web techniques and machine learning-based, statistical inferencing, we developed an approach to create and work with data mining models in SPARQL. Our framework enables to predict/classify unseen data (or features) and relations in a new data set based on

the results of a mining model. In particular, our approach enables the usage of SRL methods, which take the relations between objects into account. This allows us to induce statistical models without prior propositionalization (*i.e.*, translation to a single table) [2]—a cumbersome and error-prone task.

Our Contributions. We propose a novel extension to SPARQL called SPARQL-ML to support data mining tasks for knowledge discovery in the Semantic Web. Our extension introduces new keywords to the SPARQL syntax to facilitate the induction of models as well as the use of the model for prediction/classification. We, therefore, extend the SPARQL grammar with the `CREATE MINING MODEL` and the `PREDICT` statements (among others) as explained in Section 4.

To ensure the extensibility of SPARQL-ML with other learning methods, we created the *SPARQL Mining Ontology (SMO)* that enables the seamless integration of additional machine learning techniques (see Section 4). We show that SRLs—we use Relational Probability Trees (RPTs) and Relational Bayesian Classifiers (RBCs) [10, 11]—are able to exploit the rich and complex heterogeneous relational structure of Semantic Web data (Section 5).

Experimental Results. To validate our approach, we perform three sets of experiments (all in Section 5): first, in the *project success experiment*, we show that, using a synthetic data set, the combination of statistical inference with logical deduction produces superior performance over statistical inference only; second, the *Semantic Web service domain prediction experiment* expands on these findings using a well-known Semantic Web benchmarking data set; last, the *SVM-benchmark experiment* shows the superiority of our approach compared to a state-of-the-art kernel-based Support Vector Machine (SVM) [1] using a real-world data set.

2 Related Work

Little work has been done so far on seamlessly integrating knowledge discovery capabilities into SPARQL. Recently, Kochut and Janik [9] presented SPARQLeR, an extension of SPARQL to perform semantic association discovery in RDF (*i.e.*, finding complex relations between resources). One of the main benefits of our work is that we are able to use a multitude of different, pluggable machine learning techniques to not only perform semantic association discovery, but also classification and clustering.

Similarily, Gilardoni *et al.* [4] argued that machine learning techniques are needed to build a semantic layer on top of the traditional Web. Therefore, the support from tools that are able to work autonomously is needed to add the required semantic annotations. We claim that our SPARQL-ML approach offers this support and, thus, facilitates the process of (semi-) automatic semantic annotation (through classification).

We are aware of two independent studies that focus on data mining techniques for Semantic Web data using Progol—an Inductive Logic Programming (ILP) system. Edwards *et al.* [3] conducted an empirical investigation of the quality of various machine learning methods for RDF data classification, whereas

Hartmann [5] proposed the ARTEMIS system that provides data mining techniques to discover common patterns or properties in a given RDF data set. Our work extends their suggestions in extending the Semantic Web infrastructure in general with machine learning approaches, enabling the exploration of the suitability of a large range of machine learning techniques (as opposed to few ILP methods) to Semantic Web tasks without the tedious rewriting of RDF data sets into logic programming formalisms.

A number of studies relate to our experiments in Section 5. The study of Hess and Kushmerick [6] presented a machine learning approach for semi-automatic classification of web services. Their proposed application is able to determine the category of a WSDL web service and to recommend it to the user for further annotation. They treated the determination of a web service's category as a text classification problem and applied traditional data mining algorithms, such as Naive Bayes and Support Vector Machines. Our second experiment (see Section 5.2) is similar in that it employs OWL-S service descriptions. In contrast to [6], we employ SRL algorithms to perform service classification.

Last, Bloehdorn and Sure [1] explored an approach to classify ontological instances and properties using SVMs (*i.e.*, kernel methods). They presented a framework for designing such kernels that exploit the knowledge represented by the underlying ontologies. Inspired by their results, we conducted the same experiments using our proposed SPARQL-ML approach (see Section 5.3). Initial results show that we can outperform their results by a factor of about 10%.

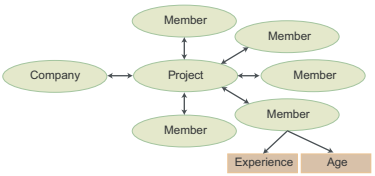
3 Background—A Brief Introduction to SRL

We briefly review the two statistical relational learning (SRL) methods we use in this paper: Relational Bayesian Classifiers (RBCs) and Relational Probability Trees (RPTs). These methods have been shown to be very powerful for SRL, as they model not only the intrinsic attributes of objects, but also the extrinsic relations to other objects [2, 10, 11].

3.1 Relational Bayesian Classifiers (RBCs)

An RBC is a modification of the traditional Simple Bayesian Classifier (SBC) for relational data [11]. SBCs assume that the attributes of an instance are conditionally independent of its class C. Hence, the probability of the class given an example instance can be computed as the product of the probabilities of the example's attributes $A_1, \ldots, A_n$ given the class (*e.g.*, $P(C = YES \mid A_1, \ldots, A_n) = \alpha P(C = YES) \prod_{i=1}^{n} P(A_i \mid C = YES)$, where α is a scaling factor dependent only on the attributes $A_1, \ldots, A_n$). RBCs apply this independence assumption to relational data. Before being able to estimate probabilities, RBCs decompose (flatten) structured examples down to the attribute level.

Figure 1 shows an example instance (subgraph) to predict the success of a business project in a relational data set. This heterogeneous instance is transformed to homogenous sets of attributes shown in the table on the right. The



Success	Member's Age	Member's Exp.
YES	{23,25,37,32,41}	{2,4,3,1,12}
YES	{18,25}	{1,7}
NO	{17,26,45}	{1,3,17}
...	...	...

Fig. 1. Example relational data represented as subgraph [11] on the left, and decomposed by attributes on the right.

attributes for the prediction of the project success include the age and experience distributions of the project members. Each row in the table stands for a subgraph, each column represents one of its attributes, and the cells contain the multisets (or distributions) of values of attributes.

Learning an RBC model then basically consists of estimating probabilities for each attribute. Such estimation methods include, but are not limited to, average-value and independent-value estimations [11]. As an example for average-value estimation, the probability of a business project B to be successful (*i.e.*, $C = YES$) based only on the attribute "Member Age" (*i.e.*, the project members age distribution) could be estimated by $P(C = YES \,|\, B) = \alpha P(Average_{Age} \geq 33 \,|\, C = YES) P(C = YES)$. For an elaborate treatment of this estimation techniques, please refer to the papers of Neville *et al.* [10, 11].

3.2 Relational Probability Trees (RPTs)

RPTs extend standard probability estimation trees (also called *decision trees*) to a relational setting, in which data instances are heterogeneous and interdependent [10]. The algorithm first transforms the relational data to multisets of attributes (Figure 1). It then attempts to construct an RPT by searching over the space of possible binary splits of the data based on the relational features, until further processing no longer changes the class distributions significantly.

Figure 2 shows an example RPT for the task of predicting whether or not a business project is successful. The leaf nodes show the distribution of the training examples (that "reached the leaf") and the resulting class probabilities of the *isSuccessful* target label. Note that the probability estimates are smoothed by a Laplace correction (*i.e.*, $\frac{p+1}{N+C}$, where p is the number of examples of a specific class, N the total number of examples, and C the number of classes). For the topmost left leaf this evaluates to $P(YES) = \frac{0+1}{117+2} = 0.008$ [13].

The features for splitting the data are created by mapping the multisets of values into single value summaries with the help of aggregation functions [10]. For our business project example in Figure 1 the root node splits the data (*i.e.*, the projects) along the summary count(Non-Managers) into projects with less than 4 non-managers and those with four or more non-managers.

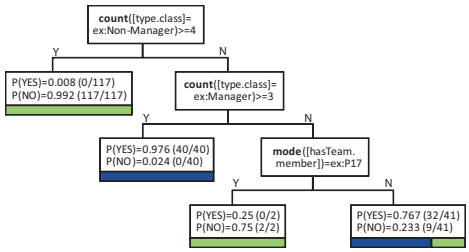


Fig. 2. Example RPT (pruned) to predict the successfulness of business projects.

4 Our Approach: SPARQL-ML

SPARQL-ML (SPARQL Machine Learning) is an extension of SPARQL that extends the Semantic Web query language with knowledge discovery capabilities. Our extensions add new syntax elements and semantics to the official SPARQL grammar described in [15]. In a nutshell, SPARQL-ML facilitates the following two tasks on any Semantic Web data set: (1) train/learn/induce a model based on training data using the new `CREATE MINING MODEL` statement (Section 4.1); and (2), apply a model to make predictions via two new *property functions* (Section 4.2). The model created in the `CREATE MINING MODEL` step follows the definitions in our *SPARQL Mining Ontology (SMO)* presented in Section 4.1.

We implemented SPARQL-ML as an extension to ARQ—the SPARQL query engine for Jena.[1] Our current version of SPARQL-ML[2] supports, but is not limited to *Proximity*[3] and *Weka*[4] as data mining modules. Note that to preserve the readability of this paper we focus solely on supervised induction methods (in particular classification/regression), even though SPARQL-ML supports the whole breadth of machine learning methods provided by its data mining modules.

4.1 Learning a Model

SPARQL-ML enables to induce a classifier (model) on any Semantic Web training data using the new `CREATE MINING MODEL` statement. The chosen syntax was inspired by the Microsoft Data Mining Extension (DMX) that is an extension of SQL to create and work with data mining models in Microsoft SQL Server Analysis Services (SSAS) 2005.[5]

The extended SPARQL grammar is tabulated in Table 1 and Listing 1.1 shows a particular example query (used in Section 5.1). Note that we omit prefixes in all queries for compactness. Our approach adds the *CreateQuery* symbol

[1] `http://jena.sourceforge.net/`
[2] Available at `http://www.ifi.uzh.ch/ddis/sparql-ml.html`
[3] `http://kdl.cs.umass.edu/proximity/index.html`
[4] `http://www.cs.waikato.ac.nz/ml/weka/`
[5] `http://technet.microsoft.com/en-us/library/ms132058.aspx`

[1]	*Query*	::=	Prologue(SelectQuery | ConstructQuery | DescribeQuery | AskQuery | CreateQuery)
[2]	*CreateQuery*	::=	CREATE MINING MODEL' SourceSelector '{' Var 'RESOURCE' 'TARGET' (Var ('RESOURCE' | 'DISCRETE' | 'CONTINUOUS') 'PREDICT'?)+ '}' DatasetClause* WhereClause SolutionModifier UsingClause
[1.2]	*UsingClause*	::=	'USING' SourceSelector BrackettedExpression

Table 1. Extended SPARQL grammar for the `CREATE MINING MODEL` statement.

```
1  CREATE MINING MODEL <http://www.example.org/projectSuccess>
2    { ?project RESOURCE TARGET
3      ?success DISCRETE PREDICT {'YES','NO'}
4      ?member RESOURCE
5      ?class RESOURCE
6    }
7  WHERE
8    { # SPARQL Basic Graph Pattern (BGP) matching part (lines 9-11)
9      ?project ex:isSuccess ?success .
10     ?project ex:hasTeam   ?member .
11     ?member  rdf:type     ?class .
12   } USING <http://kdl.cs.umass.edu/proximity/rpt>
```

Listing 1.1. SPARQL-ML `CREATE MINING MODEL` query.

to the official SPARQL grammar rule of *Query* [15]. The structure of *CreateQuery* resembles the one of *SelectQuery*, but has complete different semantics: the *CreateQuery* expands to Rule 1.1 adding the new keywords `CREATE MINING MODEL` to the grammar followed by a *SourceSelector* to define the name of the trained model. In the body of *CreateQuery*, the variables (attributes) to train the model are listed. Each variable is specified with its content type, which is currently one of the following: `RESOURCE`—variable holds an RDF resource (IRI or blank node), `DISCRETE`—variable holds a discrete/nominal literal value, `CONTINUOUS`—variable holds a continuous literal value, and `PREDICT`—tells the learning algorithm that this feature should be predicted. The first attribute is additionally specified with the `TARGET` keyword to denote the resource for which a feature should be predicted (cf. Neville *et al.* [10]).

After the usual *DatasetClause*, *WhereClause*, and *SolutionModifier*, we introduced a new *UsingClause*. The *UsingClause* expands to Rule 1.2 that adds the new keyword `USING` followed by a *SourceSelector* to define the name and parameters of the learning algorithm.

Semantics of `CREATE MINING MODEL` Queries. According to Pérez *et al.* [12], a SPARQL query consists of three parts: the *pattern matching part*, the *solution modifiers*, and the *output*. In that sense, the semantics of the `CREATE MINING MODEL` queries is the construction of new triples describing the metadata of the trained model (*i.e.*, a new output type). An example of such metadata for a particular model is shown in Listing 1.2, which follows the definitions of our SPARQL Mining Ontology (SMO) in Figure 3. The ontology enables to permanently save the parameters of a learned model, which is needed by the predict queries (see next section). The ontology includes the model name, the used learning algorithm, all variables/features being used to train the classifier, as well as

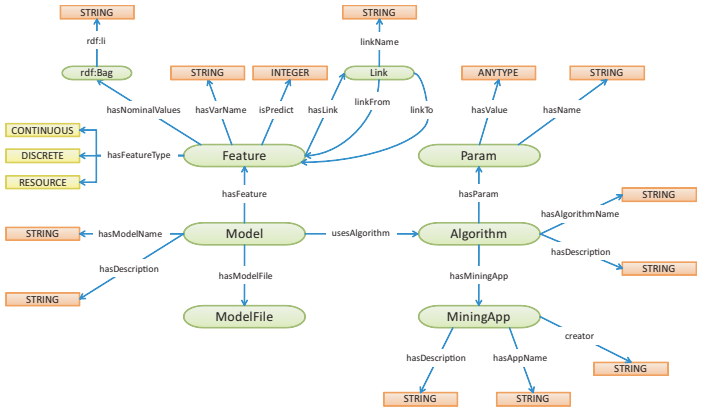


Fig. 3. SPARQL-ML Mining Ontology (SMO).

additional information, such as where to find the generated model file. In Listing 1.2, lines 1–7 show the constructed triples of a model with name *projectSuccess*, while lines 9–15 show the metadata for a particular feature of the model.

4.2 Making Predictions

After inducing a model with `CREATE MINING MODEL`, SPARQL-ML allows the user to make predictions with the model via two new property functions (*i.e.*, `sml:predict` and `sml:mappedPredict`).[6] The concept behind property functions is simple: whenever the predicate of a triple pattern is prefixed with a special name (*i.e.*, `sml`), a call to an external function is made and arguments are passed to the function (in our case by the object of the triple pattern).

The example query in Listing 1.3 explains the usage of `sml:mappedPredict` (line 7). As arguments, the function takes the identifier of the previously learned model (Section 4.1) and the instance as defined by the parameters used whilst training the model (in our case specified by the variables `?project`, `?success`, `?member`, and `?class`). In Listing 1.1, we induced a classifier to predict the value for the variable `?success` on the *training data*. This classifier is then used on line 7 in Listing 1.3 to predict the value for `?award` on the *test data*. The result of the prediction, either 'YES' or 'NO', and its probability are finally bound on line 6 to the variables `?prediction` and `?probability` respectively.

One benefit of the chosen approach is that we are able to use a number of different models in the same query. In some sense, the property functions introduce *virtual triples* [8] (*i.e.*, for the predictions and probabilities) into the query result set that are computed at run-time rather than present in the knowledge base.

[6] `http://jena.sourceforge.net/ARQ/extension.html#propertyFunctions`

```
<http://www.example.org/projectSuccess>
   smo:hasModelFile <http://www.example.org/projectSuccess/model_RPT.xml> ;
   smo:hasFeature <http://www.example.org/projectSuccess#project> ;
   smo:hasFeature <http://www.example.org/projectSuccess#success> ;
   smo:usesAlgorithm <http://kdl.cs.umass.edu/proximity/rpt> ;
   smo:hasModelName "projectSuccess" ;
   a       smo:Model .

<http://www.example.org/projectSuccess#success>
   smo:hasVarName "success" ;
   smo:isPredict "1" ;
   smo:hasFeatureType "DISCRETE" ;
   smo:hasLink <http://www.example.org/projectSuccess/link/isSuccessful> ;
   smo:hasNominalValues _:b1 ;
   a       smo:Feature .

<http://www.example.org/projectSuccess/link/isSuccessful>
   smo:linkName "isSuccess" ;
   smo:linkFrom <http://www.example.org/projectSuccess#project> ;
   a       smo:Link .

_:b1  rdf:li "NO" ;
   rdf:li "YES" ;
   a       rdf:Bag .
```

Listing 1.2. Example metadata for a learned classifier.

```
1  SELECT DISTINCT ?person ?award ?prediction ?probability
2  WHERE
3    { ?person ex:hasAward  ?award .
4      ?person ex:hasFriend ?friend .
5      ?friend rdf:type     ?class .
6      ( ?prediction ?probability )
7                 sml:mappedPredict ( <http://www.example.org/projectSuccess>
8                                     '?project = ?person' '?success = ?award'
9                                     '?member = ?friend' '?class = ?class' )
10   }
```

Listing 1.3. SPARQL-ML example predict query 1: apply the model on a data set with a different ontology structure.

Semantics of Predict Queries. The semantics of our predict queries is basically that of a *prediction join*:[7] (1) `mappedPredict` maps the variables in the Basic Graph Pattern (BGP) to the features in the specified model, which allows us to apply a model on a data set with a different ontology structure; (2) `mappedPredict` creates instances out of the mappings according to the induced model; (3) the model is used to classify an instance as defined in the `CREATE MINING MODEL` query (Listing 1.1); and (4), the values of the prediction and its probability are bound to variables in the predict query (line 6 in Listing 1.3). Note that we also defined a shorter version for `mappedPredict` in case the model is used on the same data set (*i.e.*, `sml:predict`; Listing 1.4).

[7] `http://msdn2.microsoft.com/en-us/library/ms132031.aspx`

```
SELECT DISTINCT ?project ?success ?prediction ?probability
WHERE
  { ?project ex:isSuccess ?success .
    ?project ex:hasTeam   ?member .
    ?member  rdf:type     ?class .
    ( ?prediction ?probability )
                    sml:predict ( <http://www.example.org/projectSuccess>
                                  ?project ?success ?member ?class )
  }
```

Listing 1.4. SPARQL-ML example predict query 2: apply the model on a data set with the same ontology structure.

5 Experimental Analysis

The goal of our experiments was to show the usefulness and the simplicity of the integration of machine learning methods with the existing Semantic Web infrastructure. Furthermore, we wanted to show the advantage that the combination of logic deduction and statistical induction holds over induction only. Last, we wanted to compare the performance of the SRL algorithms that we integrated into SPARQL-ML with another state-of-the-art approach. To that end, we conducted three experiments. The *project success experiment* and the *Semantic Web service domain prediction experiment* both show the ease of use as well as the advantage of the combination of induction and deduction. The *SVM-benchmark experiment* compares the prediction performance of our approach to another state-of-the-art method. In the following, we present each of the experiments in detail describing our experimental setup and discussing the empirical results.

5.1 The Project Success Experiment

Evaluation Procedure and Data Set. In order to show the ease of use and predictive capability of our system, we put together a proof of concept setting with a small, artificially created data set. We chose an artificial data set to better understand the results and to reduce any experimental noise. The constructed *business project data set* consists of different business projects and the employees of an imaginary company. The company has 40 employees each of which having one out of 8 different occupations. Figure 4 shows part of the created ontology in more detail. 13 employees belong to the superclass `Manager`, whereas 27 employees belong to the superclass `Non-Manager`.

We then created business projects and randomly assigned up to 6 employees to each project. The resulting teams consist of 4 to 6 members. Finally, we randomly defined each project to be successful or not, with a bias for projects being more successful, if more than three team members are of type `Manager`. The resulting data set contains 400 projects with different teams. The prior probability of a project being successful is 35%. We did a 50:50 split of the data and followed a single holdout procedure, swapping the roles of the testing and training set and averaged the results.

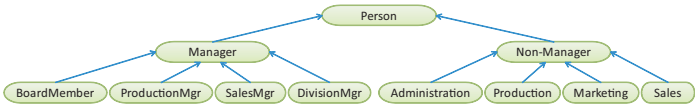


Fig. 4. Example business ontology.

Experimental Results. Listing 1.1 shows the `CREATE MINING MODEL` query that we used in the model learning process. We tested different learning algorithms with and without the support of inferencing. With the reasoner disabled, the last triple pattern in the `WHERE` clause (line 11) matches only the direct type of the received employee instance (*i.e.*, if an employee is a 'direct' instance of class `Manager`). This is the typical situation in relational databases without the support of inheritance. With inferencing enabled, the last triple pattern also matches all inferred types, indicating if an employee is a `Manager` or not.

Given the bias in the artificial data set, it is to be expected that the ability to infer if a team member is a `Manager` or not is central to the success of the induction procedure. Consequently, we would expect that models induced on the inferred model should exhibit a superior performance. The results shown in Figure 5 confirm our expectations. The Figure shows the results in terms of prediction accuracy (ACC; in legend), Receiver Operating Characteristics (ROC; graphed), and the area under the ROC-curve (AUC; also in legend). The ROC-curve graphs the true positive rate (y-axis) against the false positive rate (x-axis), where an ideal curve would go from the origin to the top left (0,1) corner, before proceeding to the top right (1,1) one [14]. It has the advantage to show the prediction quality of a classifier independent of the distribution (and, hence, prior) of the underlying data set. The area under the ROC-curve is, typically, used as a summary number for the curve. Note that a random assignment whether a project is successful or not is also shown as a line form the origin (0,0) to (1,1). The learning algorithms shown are a Relational Probability Tree (RPT), a Relational Bayes Classifier (RBC), both with and without inferencing, and, as a baseline, a k-nearest neighbor learning algorithm (k-NN) with inferencing and $k = 9$ using a maximum common subgraph isomorphism metric [16] to compute the closeness to neighbors.

As the Figure shows, the relational methods clearly dominate the baseline k-NN approach. As expected, both RPT and RBC with inferencing outperform the respective models without inferencing. It is interesting to note, however, that RPTs seem to degrade more with the loss of inferencing than RBCs. Actually, the lift of an RBC with inferencing over an RBC without inferencing is only small. These results support our assumption that the combination of induction and deduction should outperform pure induction. The major limitation of this finding is the artificial nature of the data set. We, therefore, decided to conduct a second experiment with the same goal using a real-world data set, which we present next.

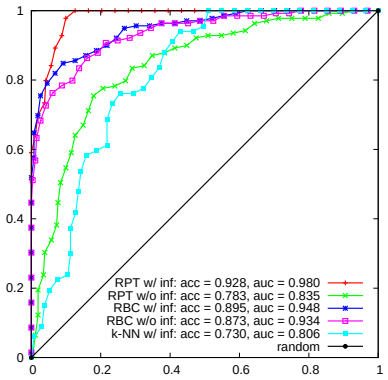


Fig. 5. ROC-Curves of business project success prediction.

5.2 The Semantic Web Service Domain Prediction Experiment

Evaluation Procedure and Data Set. Our first experiment showed that SPARQL-ML can easily induce and apply accurate models that rely on inferred data. The goal of our second set of experiments was to provide further support for these findings using a real-world test-collection in a non-binary classification task. Specifically, we used the OWLS-TC v2.1 service retrieval test collection that contains 578 OWL-S Semantic Web service descriptions.[8] OWLS-TC contains OWL-S services in seven service categories: *communication*, *economy*, *education*, *food*, *medical*, *travel*, and *weapon*. The prior distribution of the services is *communication* = 5.02%, *economy* = 35.63%, *education* = 23.36%, *food* = 4.33%, *medical* = 8.99%, *travel* = 18.34%, and *weapon* = 4.33% (*i.e.*, *economy* is the domain with most services).

For our experiments, we induced an RPT to predict the service category of a service based on its input and output concepts. We limited our investigations to the I/O parameters as we believe that they are most informative for this task. Again, we ran the experiment once on the asserted and once on the (logically) inferred model. Furthermore, we performed a 10-fold cross validation, where 90% of the data was used to learn a classification model and the remaining 10% to test the effectiveness of the learned model. This approach is standard practice in machine learning.

Experimental Results. Listing 1.5 shows the `CREATE MINING MODEL` query that we used in the model learning step. By using `OPTIONAL` patterns, we enable the inclusion of services with no outputs or inputs. The additional `OPTIONAL` pattern for the `rdfs:subClassOf` triple enables us to run the same query on the asserted and the inferred data.

The averaged classification accuracy of the results of the 10 runs is 0.5102 on the asserted and 0.8288 on the inferred model. Hence, the combination of

[8] `http://projects.semwebcentral.org/projects/owls-tc/`

```
CREATE MINING MODEL <http://www.ifi.uzh.ch/ddis/services>
  { ?service RESOURCE TARGET
    ?domain DISCRETE PREDICT
      { 'communication','economy','education','food','medical','travel',
        'weapon'
      }
    ?profile RESOURCE
    ?output RESOURCE
    ?outputType RESOURCE
    ?outputSuper RESOURCE
    ?input RESOURCE
    ?inputType RESOURCE
    ?inputSuper RESOURCE
  }
WHERE
  { ?service service:presents  ?profile .
    ?service service:hasDomain ?domain .
    OPTIONAL
      { ?profile profile:hasOutput     ?output .
        ?output  process:parameterType ?outputType .
        OPTIONAL
          { ?outputType rdfs:subClassOf ?outputSuper . }
      }
    OPTIONAL
      { ?profile profile:hasInput      ?input .
        ?input   process:parameterType ?inputType .
        OPTIONAL
          { ?inputType rdfs:subClassOf ?inputSuper . }
      }
  } USING <http://kdl.cs.umass.edu/proximity/rpt> ('maxDepth' = 6)
```

Listing 1.5. `CREATE MINING MODEL` query for service classification.

Domain	FP Rate		Precision		Recall		F-measure	
	w/o inf	*w/ inf*	*w/o inf*	*w/ inf*	*w/o inf*	*w/ inf*	*w/o inf*	*w/ inf*
communication	0.007	0.004	0.819	0.900	0.600	0.600	0.693	0.720
economy	0.081	0.018	0.810	0.964	0.644	0.889	0.718	0.925
education	0.538	0.090	0.311	0.716	0.904	0.869	0.463	0.786
food	0	0.002	0	0.960	0	0.800	0	0.873
medical	0.006	0.030	0	0.688	0	0.550	0	0.611
travel	0	0.069	1	0.744	0.245	0.873	0.394	0.803
weapon	0.002	0.002	0.917	0.964	0.367	0.900	0.524	0.931
average	**0.091**	**0.031**	**0.551**	**0.848**	**0.394**	**0.783**	**0.399**	**0.807**
t-test (paired, one-tailed)		**p=0.201**		**p=0.0534**		**p=0.00945**		**p=0.0038**

Table 2. Detailed results for the Semantic Web service classification experiments.

logical deduction with induction improves the accuracy by 0.3186 over pure induction. The detailed results in Table 2 further confirm the results for all seven domains by listing the typical data mining measures false positive rate (FP Rate), Precision, Recall, and F-measure for all categories. As the results of the t-test show, the differences for Recall and F-measure are (highly) significant. The results for Precision just barely misses significance at the 95% level.

When investigating the structure of the relational probability trees, the trees induced on the inferred model clearly exploit inheritance relations using the `rdfs:subClassOf` predicate, indicating that the access to the newly inferred triples improves the determination of a service's category. These observations

further support our finding that a combination of deduction and induction is useful for Semantic Web tasks and can be easily achieved with SPARQL-ML.

5.3 The SVM-Benchmark Experiment

Evaluation Procedure and Data Set. With our third set of experiments, we aimed to show possible advantages of SPARQL-ML over another state-of-the-art method. Specifically, we compared the off-the-shelf performance of a simple 19-lines SPARQL-ML statement (see Listing 1.6) with a Support Vector Machine (SVM) based approach proposed by Bloehdorn and Sure [1] following exactly their evaluation procedure.[9] In their work, they introduced a framework for the design and evaluation of kernel methods that are used in Support Vector Machines, such as SVM^{light} [7]. The framework provides various kernels for the comparison of classes as well as datatype and object properties of instances. Moreover, it is possible to build customized, weighted combinations of such kernels. Their evaluations include two tasks: (1) prediction of the affiliation a person belongs to (*person2affiliation*), and (2) prediction of the affiliation a publication is related to (*publication2affiliation*).

As a dataset they used the SWRC ontology—a collection of OWL annotations for persons, publications, and projects, and their relations from the University of Karlsruhe.[10] The data contains 177 instances of type `Person`, 1155 of type `Publication`, as well as some instances of types `Topic`, `Project`, and `ResearchGroup`.

For the person2affiliation task, we used the `worksAtProject` and `worksAtTopic` (essentially `workedOnBy`) object properties as well as properties pointing to the publications of a given person (`publication`). For the publication2affiliation task, we used the `isAbout` and `author` object properties pointing to associated topics and authors respectively. In order to predict the affiliation a publication is related to, we defined the affiliation to be the research group, where the major part of the authors of this publication belong to.

Experimental Results. Table 3 summarizes the macro-averaged results that were estimated via Leave-One-Out Cross-Validation (LOOCV). We applied both, an RBC and an RPT learning algorithm to both tasks. The table also reports the best-performing SVM results from Bloehdorn and Sure's experiments. The RBC clearly outperformed the RPT in both predictions, hence, we report only on the results given by the RBC. For both tasks the performance of the inferred model is not very different from the one induced on the asserted model. When consulting Listing 1.6 (for person2affiliation) it is plausible to conclude that the only inferred properties (types of persons and publications) do not help to classify a person's or a publication's affiliation with an organizational unit.

As Table 3 also shows, our method clearly outperforms the kernel-based approach in terms of prediction error, recall, and F-Measure, while having an only slightly lower precision. The slightly lower precision could be a result of the

[9] We would like to thank them for sharing the exact data set used in their paper.
[10] `http://ontoware.org/projects/swrc/`

```
CREATE MINING MODEL <http://www.ifi.uzh.ch/ddis/affiliations>
  { ?person RESOURCE TARGET
    ?group RESOURCE PREDICT
    ?project RESOURCE
    ?topic RESOURCE
    ?publication RESOURCE
    ?pertype RESOURCE
    ?pubtype RESOURCE
  }
WHERE
  { ?person swrc:affiliation ?group .
    ?person rdf:type         ?personType .
    OPTIONAL { ?person swrc:worksAtProject ?project . }
    OPTIONAL { ?person swrc:worksAtTopic ?topic . }
    OPTIONAL
      { ?person      swrc:publication ?publication .
        ?publication rdf:type         ?publicationType .
      }
  } USING <http://kdl.cs.umass.edu/proximity/rbc>
```

Listing 1.6. `CREATE MINING MODEL` query for the person2affiliation task.

person2affiliation					**publication2affiliation**				
algorithm	*err*	*prec*	*rec*	F_1	*algorithm*	*err*	*prec*	*rec*	F_1
sim-ctpp-pc, c=1	4.49	95.83	58.13	72.37	sim-cta-p, c=10	0.63	99.74	95.22	97.43
RBC w/o inf	3.53	87.09	80.52	83.68	RBC w/o inf	0.09	98.83	99.61	99.22
RBC w/ inf	3.67	85.72	80.18	82.86	RBC w/ inf	0.15	97.90	99.25	98.57

Table 3. LOOCV results for the *person2affiliation* and *publication2affiliation* tasks.

limitation to just a few properties used by an off-the-shelf approach without a single parameter setting, whereas the SVM approach is the result of extensive testing and tuning of the kernel method's properties and parameters.

We conclude from this experiment, that writing a SPARQL-ML query is a simple task for everyone familiar with the data and the SPARQL-ML syntax. Kernels, on the other hand, have the major disadvantage that the user has to choose from various kernels, kernel modifiers, and parameters. This constitutes a major problem for users not familiar with kernels and SVM algorithms.

6 Conclusions, Limitations, and Future Work

We have presented a novel approach we call SPARQL-ML that extends traditional SPARQL with data mining support to perform knowledge discovery in the Semantic Web. We showed how our framework enables to predict/classify unseen data in a new data set based on the results of a mining model. In particular, we demonstrated how models trained by statistical relational learning (SRL) methods outperform models not taking into account additional deduced (or inferred) information about the links between objects.

We fully analyzed SPARQL-ML on synthetic and real-world data sets to show its excellent prediction/classification quality as well as its superiority to other related approaches, such as kernel methods used in Support Vector Machines.

Our approach is extensible in terms of the supported machine learning algorithms and generic as it is applicable for any Semantic Web data set.

We note that the performance loss when mining on inferred knowledge bases (see Table 3) is a limitation of the employed ontologies (or data sets) and not of our approach and the used techniques themselves. We speculate that the loss could be eliminated by using more comprehensive ontologies.

Future work will evaluate further relational learning methods such as the ones proposed by NetKit[11] or Alchemy.[12] Finally, given the usefulness and the ease of use of our novel approach, we believe that SPARQL-ML could serve as a standardized approach for data mining tasks on Semantic Web data.

References

1. S. Bloehdorn and Y. Sure. Kernel Methods for Mining Instance Data in Ontologies. In *ISWC*, pages 58–71, 2007.
2. S. Dzeroski. Multi-Relational Data Mining: An Introduction. *SIGKDD*, 5(1):1–16, 2003.
3. P. Edwards, G. A. Grimnes, and A. Preece. An Empirical Investigation of Learning from the Semantic Web. In *ECML/PKDD Workshop*, pages 71–89, 2002.
4. L. Gilardoni, C. Biasuzzi, M. Ferraro, R. Fonti, and P. Slavazza. Machine Learning for the Semantic Web: Putting the user into the cycle. In *Dagstuhl Seminar*, 2005.
5. J. Hartmann. A Knowledge Discovery Workbench for the Semantic Web. In *SIGKDD Workshop*, 2004.
6. A. Hess and N. Kushmerick. Machine Learning for Annotating Semantic Web Services. In *AAAI*, 2004.
7. T. Joachims. *SVM light—Support Vector Machine*, 2004. Software available at `http://svmlight.joachims.org/`.
8. C. Kiefer, A. Bernstein, and M. Stocker. The Fundamentals of iSPARQL: A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. In *ISWC*, pages 295–309, 2007.
9. K. Kochut and M. Janik. SPARQLeR: Extended SPARQL for Semantic Association Discovery. In *ESWC*, pages 145–159, 2007.
10. J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning Relational Probability Trees. In *KDD*, pages 625–630, 2003.
11. J. Neville, D. Jensen, and B. Gallagher. Simple Estimators for Relational Bayesian Classifiers. In *ICDM*, pages 609–612, 2003.
12. J. Pérez, M. Arenas, and C. Gutierrez. The Semantics and Complexity of SPARQL. In *ISWC*, pages 30–43, 2006.
13. F. Provost and P. Domingos. Well-Trained PETs: Improving Probability Estimation Trees. *CeDER Working Paper #IS-00-04*, 2000.
14. F. J. Provost and T. Fawcett. Robust Classification for Imprecise Environments. *Machine Learning*, 42(3):203–231, 2001.
15. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Technical report, W3C Recommendation 15 January 2008.
16. G. Valiente. *Algorithms on Trees and Graphs*. Springer-Verlag, Berlin, 2002.

[11] `http://www.research.rutgers.edu/~sofmac/NetKit.html`
[12] `http://alchemy.cs.washington.edu/`