

Relational Kernel Machines for Learning from Graph-structured RDF Data ^{*}

Veli Bicer¹, Thanh Tran², and Anna Gossen³

¹ FZI Forschungszentrum Informatik, Haid-und-Neu-Str. 10-14,
76131 Karlsruhe, Germany, bicer@fzi.de

² Institute AIFB, Geb. 11.40 KIT-Campus Sd, 76128 Karlsruhe, Germany
duc.tran@kit.edu

³ fluid Operations AG, Altrottstr. 31, 69190 Walldorf, Germany
anna.gossen@fluidops.com

Abstract. Despite the increased awareness that exploiting the large amount of semantic data requires statistics-based inference capabilities, only little work can be found on this direction in the Semantic Web research. On semantic data, supervised approaches, particularly kernel-based Support Vector Machines (SVM), are promising. However, obtaining the right features to be used in kernels is an open problem because the amount of features that can be extracted from the complex structure of semantic data might be very large. Further, combining several kernels can help to deal with efficiency and data sparsity but creates the additional challenge of identifying and joining different subsets of features or kernels, respectively. In this work, we solve these two problems by employing the strategy of *dynamic feature construction* to compute a hypothesis, representing the relevant features for a set of examples. Then, a composite kernel is obtained from a set of *clause kernels* derived from components of the hypothesis. The learning of the hypothesis and kernel(s) is performed in an interleaving fashion. Based on experiments on real-world datasets, we show that the resulting *relational kernel machine* improves the SVM baseline.

1 Introduction

The amount of semantic data captured in ontologies or made publicly available as RDF Linked Data is large and ever increasing. While deductive reasoning has been the research focus so far, and is desirable for some cases, it has become common belief that the scale of semantic data also demands for more efficient statistics-based techniques.

Despite the increased awareness that dealing with the large amount of semantic data benefits from statistics-based inference capabilities, only the few seminal approaches mentioned above can be found. While they show how existing techniques can be applied semantic data, many challenges specific to this scenario remain unresolved.

^{*} This research was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference 01MQ07012. The authors take the responsibility for the contents.

that dealing with the large amount of semantic data benefits from statistics-based inference capabilities, only little work can be found in Semantic Web research. Basically, semantic data can be conceived as a graph where nodes stand for resources and edges capture attribute values or relations between resources. The main machine learning (ML) approaches in the field of *relational learning* that directly operate on relational and graph-structured data of this kind are *Inductive Logic Programming* (ILP) and *Statistical Relational Learning* (SRL). In [7] the authors propose the use of existing SRL approaches and their integration into SPARQL such that both deductive and inductive reasoning can be employed for query answering. Also, it has been shown how ontologies can be exploited as prior knowledge to improve SRL [11]. Lastly, most relevant for this paper is the work on using kernels and *Support Vector Machines* (SVM) for learning on semantic data [4, 3, 1].

While unsupervised SRL approaches provide a probability distribution over the entire set of input data (e.g. the entire RDF graph) for a possibly large amount of random variables, supervised approaches such as ILP and SVM are conducted to make a prediction for one single random variable. While the former is thus more flexible and can be used to infer different kinds of knowledge, the latter focuses on one single prediction problem and as a result, exhibits better scalability and mostly, also better predictive performance. In fact, kernel-based SVM is an important topic in machine learning (ML) due to the robustness of SVM and the fact that kernels flexibly allows for data of arbitrary structure (e.g. as complex as graphs) to be exploited for learning. In particular, it is promising for learning on semantic data [4, 3, 1]. Following this line of research, we address two major problems:

Problem 1. Basically, a kernel measures the similarity of two data points based on their features. *Obtaining the right features* is a classic ML problem, which is exacerbated in the semantic data setting. Here, a data point (e.g. a RDF resource or a triple) might have a large set of features (e.g. all attributes of a resource, all related resources and all those related to them). In the extreme case, the features for every RDF resource correspond to the entire RDF graph. Basically, a preprocessing technique can materialize all the possible features derived from relations between the resources and store them in a single table. However, incorporating all possible features like this is expensive and that might be not affordable when the data is too large. While the previous work on using SVM on semantic data also incorporates complex kernels that can take relations between resources into account, it does not answer the question which relations are to be used for implementing such kernels [4, 3, 1]. In fact, not only SVM but also the mentioned SRL approach [7] require features to be chosen manually by an expert. More precisely, the latter does not train the model on the entire dataset but focuses on the ‘features’ defined in the SPARQL query.

Problem 2. Related to this is the problem of *combining different sets of features*. The authors in [1] already made clear that several kernels are needed for considering different kinds of similarities, and proposed a simple weighted additive combination. Generally, instead of one single kernel, multiple ones might be used for reasons of scalability and data sparsity. The goal is to identify and apply kernels only on some non-sparse subsets of the data. However, not only the question

which kernels are to be used but also the tuning of their weights are left open in [1].

Contributions. We address these two problems of applying kernel-based SVM on semantic and relational data. The contributions can be summarized as follows:

- We present a novel approach that combines relational learning and kernel-based SVM learning to obtain what we call *Relational Kernel Machines* for graph-structured data. For this, we consider learning as a classification problem in which the aim is to train a kernel machine from example data points whose feature sets have to be extracted from the data graph.
- In particular, we employ the strategy of *dynamic feature constructions* used in ILP to compute relevant features by iteratively searching for a hypothesis that covers a given set of examples. The hypothesis is a set of clauses, each can be conceived as an intensional description of a subset of relevant features of the examples.
- We propose a R-convolution kernel called the *clause kernel*, which is created for every clause of the hypothesis and combined to a form of a *composite kernel*, the basis of our kernel machine.
- The technical problem behind this is that the search for a valid hypothesis (i.e., finding relevant features) and the training of the kernel machine (i.e., finding the combination of features/kernels and incorporating it into the SVM) are two dependent processes that need to be solved concurrently. We propose a *coevolution-based genetic algorithm* to solve this multi-objective optimization.

Structure. We introduce the basic concepts in Section 2. Then, our approach is presented in Section 3, where we start with the clause kernels, then discuss how they can be combined and optimized using coevolution, and finally show how the resulting kernel machines can be applied. The experiments are presented in Section 4, followed by related work in Section 5 and conclusion in Section 6.

2 Preliminaries

Learning with Kernel Machines. In this work we consider the task of classification as in the case of a Support Vector Machine (SVM). Assume a dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ with data point samples $x_i \in \mathcal{X}$ and $y_i \in \{+1, -1\}$. Assume that data points x_i are part of a graph-structured data model $G = (V, E, l)$, where $V = \{v_1, \dots, v_n\}$ is a finite set of vertices, $E \subseteq V \times \mathcal{L}_E \times V$ is a set of edges, and $l : V \rightarrow \mathcal{L}_V$ is a vertex labeling function, given a set of vertex labels \mathcal{L}_V and a set of edge labels \mathcal{L}_E . This model might capture semantic data in the form of RDF or relational data. In this paper, we focus on graph-structured RDF data.

In this context, a data point might be a RDF resource, or a RDF triple and the task here is to predict whether a given resource belongs to a class or a pair of resources instantiate a triple pattern $p(\cdot, \cdot)$ (henceforth called target predicate). In particular, we consider a data point x_i as a node pair $x_i = \langle v_1^i, v_2^i \rangle$ and decompose the dataset into positive and negative examples, such that for a target predicate $p(\cdot, \cdot)$, we have $\mathcal{D}^+ = \{\langle v_1^i, v_2^i \rangle \mid p(v_1^i, v_2^i) \in E\}$ and $\mathcal{D}^- = \mathcal{D} \setminus \mathcal{D}^+$. Thus, every data point x_i is represented as a pair of vertices, and its features are captured by edges

to other vertices in the graph. Clearly, such a data point models an RDF triple, and also this notion carries over to the case of RDF resources, e.g. every resource v can be conceived as a triple $Thing(v, type)$ and modeled as $\langle v, Thing \rangle$.

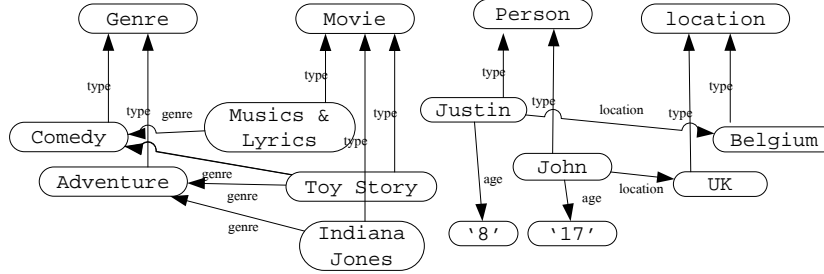


Fig. 1: An example data graph.

An example data graph is illustrated in Fig. 1. A dataset over this graph for the target predicate $likes(\cdot, \cdot)$ is $\mathcal{D} = \{(\langle Justin, ToyStory \rangle, +1), (\langle John, IndianaJones \rangle, +1), (\langle John, Music\&Lyrics \rangle, -1)\}$.

SVM learning from such a dataset can be simply conceived as operating on simple vectors of real numbers representing features of the data points $x^i \in \mathcal{D}$ in a vector space, and in the case of classification, the goal is to find a hyperplane in this space that separates points in \mathcal{D}^- from points in \mathcal{D}^+ . More formally, given the dataset, the standard SVM algorithm learns a discriminant function $f(x)$:

$$f(x) = w^T \phi(x) + b$$

by solving the following optimization problem:

$$\begin{aligned} \min \quad & \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\ \text{w.r.t. } & w \in \mathbb{R}^D, \xi \in \mathbb{R}^N, b \in \mathbb{R} \\ \text{subject to } & y_i(w^T \phi(x) + b) \geq 1 - \xi_i, \xi_i \geq 0 \end{aligned}$$

This optimization is applicable for classification of examples which are not linearly separable. This is where kernels are incorporated into SVM. They can be designed in such a way that implicitly map the input data \mathcal{D} into a higher dimensional space \mathcal{F} where the data points become linearly separable. This mapping is also refer to as the “kernel trick” because it is implicit in the sense that we neither need to access the mapped data, nor the actual mapping function, but it is sufficient to access the results of the inner product of pairs of mapped data points in \mathcal{F} . In particular, this explicit mapping is unnecessary given we have a kernel function $k_i(x_1, x_2) = \phi x_1^T \phi x_2$ that directly returns the dot product of the mapped data points ϕx_1 and ϕx_2 in some feature space.

The use of a combination of kernels for SVM learning has gained popularity and is particularly promising in the case of semantic data. This kind of data

is often sparse in the sense that the feature sets of two given data points might overlap only on some dimensions, e.g. x_1 has an address but x_2 does not. A generic way to focus on some relevant dimensions is to construct an R-convolution kernel introduced by Haussler [6]. Assume that for each data point $x \in \mathcal{X}$ there exists a D -dimensional decomposition $x = (x_1, x_2, \dots, x_D)$ with each $x_i \in \mathcal{X}_i$, and there are the kernels $k_i : \mathcal{X}_i \times \mathcal{X}_i \rightarrow \mathbb{R}$. Then, Haussler’s convolution kernel $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is defined as follows:

$$K(x_1, x_2) = \sum_{x'_1 \in R^{-1}(x_1), x'_2 \in R^{-1}(x_2)} \mu(x'_1, x'_2) \prod_{i=1}^D k_i(x'_1, x'_2)$$

where μ denote a set of non-negative coefficients on $\mathcal{X}_i \times \mathcal{X}_i$, and $R : \mathcal{X}_1 \times \dots \times \mathcal{X}_D \times \mathcal{X}$ is the decomposition relation. In [6], it is shown that $K(x_1, x_2)$ is positive semi-definite and admissible as a kernel. The intuitive idea of R-convolution is to engineer more sophisticated kernels by tailoring simple and primitive kernels. In this work, we decompose the feature set and learn an R-convolution kernel for every subset. Further, a composite kernel is learned from a non-linear combination of these R-convolution kernels to consider the entire feature set. Note that each of the R-convolution kernels is a composite kernel itself, consisting of sub-kernels that focus on some dimensions.

Learning with Logical Representations. ILP learns logical clauses from data represented using expressive knowledge representation formalisms. A well-known system is FOIL, which we extend in this work for computing relevant features from RDF data. Formally, a *hypothesis* $H = \{c_1, \dots, c_n\}$ which is a set of clauses where each clause is $h(\vec{v}) \leftarrow b_1(\vec{v}_1), \dots, b_m(\vec{v}_m)$, h and b_i are binary predicates representing the clause’s head and body, and \vec{v}_i are either variables or constants. For example, a clause indicating all users from the *United Kingdom* who likes *comedy* movies can be stated as follows:

$c_1 : \text{likes}(\text{?user}, \text{?movie}) \leftarrow \text{location}(\text{?user}, \text{UK}), \text{genre}(\text{?movie}, \text{comedy})$

A relation “ \preceq ” is called quasi-order on a set \mathcal{H} if it is reflexive (i.e. $a \preceq a$ for all $a \in \mathcal{H}$) and transitive (i.e. $a \preceq b$ and $b \preceq c$ implies $a \preceq c$). Given a search space (\mathcal{H}, \preceq) of a possible clauses and hypotheses, respectively, a *refinement operator* ρ implies a mapping of a clause c to a set of clauses $\rho(c)$, such that $\rho(c) = \{c' \mid c' \in \mathcal{H}, c' \preceq c\}$. Computing the hypothesis is based on searching and refining clauses in this space. The goal is to find a hypothesis that “covers” all positive examples and does not cover negative examples. While different cover relational have been used, learning from entailment is most widely applied, i.e., the coverage relational $\text{covers}(H, x)$ returns true iff $H \models x$ such that the result of the learning is the set $\{H \mid \forall x_1 \in \mathcal{D}^+ : H \models x_1 \wedge \forall x_2 \in \mathcal{D}^- : H \not\models x_2\}$.

3 Learning from RDF Graphs

In this section, we propose a Relational Kernel Machine for learning from graph-structured data, that can be applied to RDF data. First, we show how the ILP approach for finding hypothesis can be employed to obtain a R-convolution kernel

that is a combination of clause kernels. We also present the notion of clause kernel and discuss how the computation of the hypothesis and kernel can be formulated as a multi-objective optimization problem, and finally explain the use of a genetic algorithm to solve this.

3.1 Clause Kernel

Observe that ILP aims to find a hypothesis that captures the structure and semantics of a given set of examples. It can be seen as an intensional description of the example feature set. Further, the constituent clauses actually define how this feature set can be decomposed into subsets. This ability of the hypotheses to define features as graph substructures is quite useful for constructing a composite kernel from clause kernels, i.e., kernels constructed for every clause. In turn, as every clause contains a set of predicates, which stand for dimensions of a feature subset, we propose to construct a R-convolution kernel for every clause to employ simple kernels on dimensions.

We define a clause kernel $K_c(x_1, x_2)$ over each pair of data points $x_1, x_2 \in X$. Following the R-convolution kernels [6], we define a decomposition relation $R_c^{-1}(x)$ over the data point x for a clause c as follows:

- Given $x = \langle x_1, x_2 \rangle$, a substitution $\theta = \{V_1/x_1, V_2/x_2\}$ instantiates the variables V_1, V_2 of c with x as $c\theta$, and
- given that instantiated clause without the head predicate denoted $c_{body}\theta$, another substitution $\theta' = \{V_3/b_3, \dots, V_m/b_m\}$ instantiates the remaining variables V_i in $c_{body}\theta$ with bindings b_i as $c_{body}\theta\theta'$,
- then θ' is a R-decomposition relation for c denoted $R_c^{-1}(x)$ iff $G \models c_i\theta\theta'$ where G denotes the data graph.

In the RDF context, entailment is evaluated simply via graph pattern matching, i.e., $G \models c_i\theta\theta'$ if $c_i\theta\theta'$ is a binding to the query pattern $c_{body}\theta$ such that it is a subgraph of G .

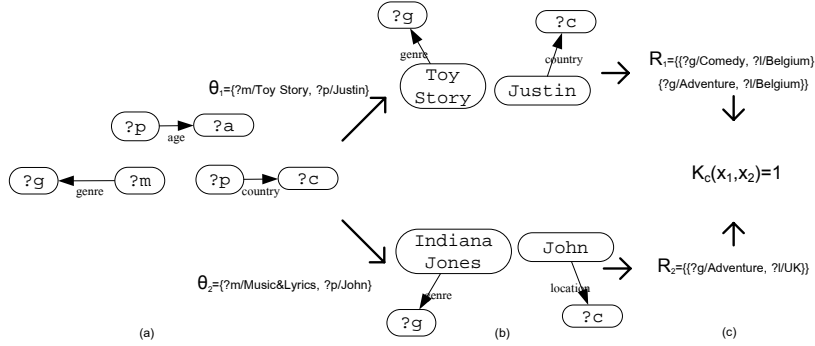


Fig. 2: An example calculation of clause kernel: (a) graph representation of a clause without head, (b) instantiated with data, (c) the results of the instantiated clauses evaluated as graph patterns and the resulting kernel value.

Intuitively speaking, this definition refers to the grounding of some variables in a clause with the data x . For example, for a clause as:

$c_1 : \text{likes}(\text{?user}, \text{?movie}) \leftarrow \text{location}(\text{?user}, \text{?c}), \text{age}(\text{?user}, \text{?a}), \text{genre}(\text{?movie}, \text{?g})$

a substitution $\theta = \{\text{?user}/\text{John}, \text{?movie}/\text{Inception}\}$ for $x = \langle \text{John}, \text{Inception} \rangle$ results in the following instantiated clause:

$p_1 : \text{likes}(\text{John}, \text{Inception}) \leftarrow \text{location}(\text{John}, \text{?c}), \text{age}(\text{John}, \text{?a}), \text{genre}(\text{Inception}, \text{?g})$

A kernel over two data points (x_1, x_2) then can be calculated by instantiating c to obtain $c\theta_1$ and $c\theta_2$, and showing how similar x_1 and x_2 are based on these instantiated clauses. A trivial evaluation of this similarity is to consider the instantiated clauses without their heads $c'_i\theta$ as graph patterns to be evaluated on the data graph. For this, given G , we define a result set $S_{c'\theta} = \{\langle r_1, \dots, r_m \rangle \mid \theta' = \{V_3/r_3, \dots, V_m/r_m\} \wedge G \models c'\theta\theta'\}$. Based on two result sets, a kernel function can be defined as:

$$k(c'_1\theta, c'_2\theta) = |\{ \langle r_i, r_j \rangle \mid r_i \in S_{c'_1\theta} \wedge r_j \in S_{c'_2\theta} \wedge r_i = r_j \}|$$

Intuitively speaking, given $c'\theta$ as a set of feature dimensions, the similarities of two data points are measured by retrieving values of those dimensions represented by variables in $c'\theta$, and check if these points agree on these values. The resulting clause kernel is a R-convolution kernel in the sense that $R_c^{-1}(x)$ decomposes the feature set captured by c into dimensions, and subkernels k_i are used to operate on these dimensions, i.e., $k_i(x'_1, x'_2)$ where $x'_1 \in R_c^{-1}(x_1)$ and $x'_2 \in R_c^{-1}(x_2)$. An example illustrating this is given in Fig. 2. In this case, the two data points agree only on the value *Adventure* for the dimension *genre*.

3.2 Kernel Learning

The kernel function we aim to learn here is constructed as a non-linear combination of many small clause kernels, which in turn, can be further decomposed into dimensions as discussed. As discussed previously, the basic clause kernels can be indexed by a set H that corresponds to a hypothesis in our case. Therefore, we adopt a formulation of kernel learning based on the search of clauses over the search space (\mathcal{H}, \preceq) where \mathcal{H} indicates the set of all possible clauses and hypotheses, respectively. Then, we propose the following optimization scheme:

$$\begin{aligned} \inf_{H \subset \mathcal{H}} \min_{w, \xi, b} \quad & \frac{1}{2} \sum_{c \in H} \frac{1}{d_c} \|w_c\|^2 + C \sum_i \xi_i \\ \text{subject to} \quad & y_i \left(\sum_{c \in H} w_c^T \phi_c(x) + b \right) \geq 1 - \xi_i, \quad \sum_{c \in H} d_c = 1, \quad \xi_i \geq 0, \quad d_c \geq 0 \end{aligned} \quad (1)$$

In the formulation above, on the one hand, the inner minimization problem represents a multiple kernel learning setting, where each clause c in a given hypothesis H contributes to the solution controlled by the value d_c . Thus, each d_c controls the importance given to the squared norm of w_c in the objective function. In [10], this inner optimization is solved as a 2-step alternating optimization algorithm, where the first step consists of solving w , b , and ξ for a fixed d , and the second

step consists of solving the vector d for fixed w , b , and ξ . On the other hand, the outer infimum problem represents a hypothesis search where a set of clauses H are determined over a hypothesis search space \mathcal{H} . Introducing Lagrange multipliers α , λ , we derive the following dual problem for optimization with kernels K_c :

$$\begin{aligned} & \sup_{H \subset \mathcal{H}} \max_{\alpha} \sum_i \alpha_i - \lambda \\ & \text{subject to } \sum_i \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C \quad i = 1, \dots, n, \\ & F(H, d) \leq \lambda \quad \forall H \in 2^{\mathcal{H}} \end{aligned} \quad (2)$$

where $F(H, d)$ is defined as:

$$F(H, d) = \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_{c \in H} d_c K_c(x_i, x_j) \quad (3)$$

The solution to the problem above can actually be decomposed into two parts: First, by directly solving the inner maximization, we obtain the optimal points (α^*, λ^*) for dual variables. Then, plugging the optimal point α^* into $F(H, d)$, the outer optimization problem yields to the following equivalent optimization:

$$\min_{H \in 2^{\mathcal{H}}, d} F(H, d) = \frac{1}{2} \sum_{i,j} \alpha_i^* \alpha_j^* y_i y_j \sum_{c \in H} d_c K_c(x_i, x_j) \quad (4)$$

Such an optimization, however, requires the hypothesis to be pre-given before solving the kernel learning problem. For solving this optimization problem, we also apply a refinement operator that changes the hypothesis iteratively. It is initiated with a top-level clause, indicating the start of the hypothesis and refines it in each step before solving the two-part optimization. At each step, a *refine* function is called that executes a refinement operator $\rho(c')$ over a selected clause c' from the current hypothesis H^t . Refinement operator performs the refinement in two ways:

1. It picks a predicate $b_i(\vec{v}_i)$ from the selected clause c' and finds another predicate b_j that is connected to b_i in the RDF graph by a shared node. For example, for the RDF graph shown in Figure 1, a predicate $location(?user, ?l)$ can be extended by $age(?user, ?a)$.
2. It replaces a free variable in the clause with a ground term (i.e. URI) of an entity. In order to select the best grounded term, it selects the most-used grounded term in the substitutions θ of clause kernels. For instance, the term *Adventure* is used to ground the variable $?g$ in the predicate $genre(?m, ?g)$ according to the example presented in subsection 3.1.

A refinement operator computes a set of refined clauses $\{c_1, \dots, c_k\}$ in each refinement. In order to select the best clause to include into the hypothesis, we apply Kernel Target Alignment (KTA) [2] to the new clauses to determine how each clause kernel K_{c_i} is aligned with the ideal kernel Y . Note that ideal kernel Y is calculated over the labels in the dataset with an outer product $Y = yy^T$, where $y = (y_1, \dots, y_n)^T$. A simple measure of the alignment is provided between the clause kernel K_{c_i} and Y by a Frobenius inner product $\langle \cdot, \cdot \rangle_F$ as:

$$\langle Y, K_{c_i} \rangle_F = \sum_{y_i=y_j} K_{c_i}(x_i, x_j) - \sum_{y_i \neq y_j} K_{c_i}(x_i, x_j) \quad (5)$$

This type of refinement is similar to the ILP techniques such as FOIL in terms of dynamically inducing new clauses of the hypothesis [9]. However, it differs in the way we utilize RDF graph to extend the clauses and test the alignment over the kernel functions before accepting a clause into the hypothesis.

3.3 Algorithm

In this section we present the overall algorithm that solves the hypothesis refinement and kernel learning together. Algorithm 1 shows the overall process which starts with a given initial target clause. To construct the first hypothesis, the *refine* function is called once as depicted in Algorithm 2. Mainly, this function selects a clause c' from a given hypothesis, applies the refinement operator $\rho(c')$ to obtain a set of refined clauses, and selects a subset of these refined clauses by using the KTA to be replaced in the hypothesis with the selected clause c' . The parameter θ indicates the number of clauses to be replaced which we set to 2 in our experiments.

Algorithm 1, then, continues with two nested loops: In the inner loop, kernel learning takes place as a two-step process. A SVM solver is used to obtain optimal (α^*, λ^*) and they are used to update the weight vector \mathbf{d} . For this, we take a simple differentiation of the dual function F w.r.t. every weight d_c as $\frac{\partial F}{\partial d_c} = \frac{1}{2} \sum_{i,j=1}^n \alpha_i^* \alpha_j^* y_i y_j K_c(x_i, x_j)$ and apply an updating scheme such as $d_c \leftarrow d_c + \gamma \frac{\partial F}{\partial d_c}$, where γ is the step size chosen according to Armijo rule as suggested in [10]. In the outer loop, the *refine* function is called again to change the hypothesis and the optimization is done again in the next iteration.

Algorithm 1: Kernel Learning Algorithm

Input: C^0 :initial clause
 $t \leftarrow 0$
 $(H^0, \mathbf{d}^0) \leftarrow \text{refine}(\{C^0\}, \{1\})$
repeat
 repeat
 $K_H \leftarrow \sum_{c \in H^t} d_c^t K_c$
 $(\alpha^*, \lambda^*) \leftarrow \text{Solve the SVM problem in Eq. 2 for } K_H$
 $S \leftarrow F(H^t, \mathbf{d}^t)$
 $d_c^t \leftarrow d_c^t + \gamma \frac{\partial F}{\partial d_c}$, where $\forall c \in H^t$ and for α^*
 until $F(H^t, \mathbf{d}^t) \geq S$
 $(H^{t+1}, \mathbf{d}^{t+1}) \leftarrow \text{refine}(H^t, \mathbf{d}^t)$
 $t \leftarrow t + 1$
until converge

Algorithm 1 iteratively refines the hypothesis and trains a SVM in each iteration. Clearly, this is an expensive procedure because optimal parameters have to be computed for all possible candidate hypothesis, even though many of them

Algorithm 2: Refine

Input: H^t : Hypothesis to be refined, \mathbf{d}^t : Weight vector
Initialize $H^{t+1} \leftarrow H^t$, $\mathbf{d}^{t+1} \leftarrow \mathbf{d}^t$
 $c' \leftarrow$ Select a clause from H^t
Apply refinement operator $\{c_1, \dots, c_k\} \leftarrow \rho(c')$
 $\text{mina} \leftarrow -\infty$, $\text{minI} \leftarrow -1$
 $\text{newClauses} = \emptyset$
for $i = 1$ to k **do**
 $a_i \leftarrow$ Calculate alignment of c_i in Eq. 5
 if $a_i > \text{mina}$ **then**
 $\text{newClauses} \leftarrow \text{newClauses} \cup c_i$
 if $|\text{newClauses}| > \theta$ **then**
 Remove c_{minI} from newClauses
 end if
 $\text{mina} \leftarrow a_i$, $\text{minI} \leftarrow i$
 end if
end for
for all $c \in \text{newClauses}$ **do**
 $H^{t+1} \leftarrow H^t \cup c$
 $d_c^{t+1} \leftarrow \frac{d_{c'}^t}{|\text{newClauses}|}$
end for
Remove c' from H^{t+1} and $d_{c'}$ from \mathbf{d}^{t+1}
return H^{t+1} , \mathbf{d}^{t+1}

may turn out to be non-optimal, i.e., yield high prediction error. We approximate this process by means of co-evolutionary genetic algorithm. Basically, GA iteratively applies crossover and mutation on the fittest solutions to breed new refined solutions. Two different but dependent species of individuals constituting two subpopulations are to be trained. The first subpopulation consists of individuals representing hypotheses and the second one is meant to train the SVM coefficients. It should be noted that the two individuals combined from both subpopulations form one candidate solution to our nested two-loop algorithm. Dual objective is used as a fitness function. Crossover and mutation is applied to two subpopulations separately as the individuals representing a hypothesis is refined in each generation with the *refine* function, and individuals representing SVM coefficients are randomly changed with a uniform crossover and two-point mutation.

3.4 Relational Kernel Machines

The purpose of the learner is to find the coefficients and a multiple kernel for a prediction function which should have the following form:

$$f(x) = \sum_i \alpha_i y_i \sum_{c \in H} d_c K_c(x_i, x)$$

This function can be constructed by selecting two individuals from each one of the subpopulations resulting from our coevolutionary optimization. After the optimization is finished, we can use the prediction error for each pairs of individuals

selected to obtain the best solution to the classification problem. Different variations of loss functions can be applied here. We use the hinge loss function:

$$l(y, f(x)) = \max(0, 1 - y * f(x))$$

4 Experiments

To demonstrate the general feasibility of our learning approach, we conducted experiments on two different evaluation scenarios. In this section, we present each setting, the extracted data-sets and discussion on the results. Our implementation, RDFLearner, is publicly available at <http://code.google.com/p/rdflearner/>.

4.1 The SWRC Ontology

In the first experiment we compare our approach with related work on kernel-based SVM learning [1].

Baseline. In particular, this approach proposes the use of a number of predefined kernels in Support Vector Machines, such as SVMlight⁴. The kernels used in the previous experiment [1] representing the baseline of this work are: *CCOP1*: A combination of common class similarity kernel and two object property kernels on *workedOnBy* and *worksAtProject*; *CCOP2*: The same as CCOP1 with the property *publication* in addition; and *CCOPDP*: CCOP2 plus a data property kernel on *title* of publications. The result was obtained by summing up the combination of corresponding weighted kernels, where the weights were manually defined and set to the same values as used before [1].

Data. For comparison purpose, we use the classification scenario and dataset that was employed previously [1]. As data, we have the SWRC Ontology⁵, which contain data from the Semantic portal of the Institute AIFB. This ontology provides a vocabulary for modeling entities and their relations in the research environment. Top-level concepts include *Person*, *Publication*, *Event*, *Organization*, *Topic* and *Project*. The associated data contain 178 instances of type *Person* that have an affiliation to one of the 4 institute’s research groups. 78 of them are employed as research staff. There are 1232 *Publication* instances, 146 instances of the type *ResearchTopic* and 146 *Project* instances.

Classification Scenario. The task of the SVMs is to predict the affiliation of a person. We take the people in the data, which are associated to the each research group, as positive examples and train a binary classifier for each group. This means we have 4 classifiers which predict whether a person belongs to a particular group or not. The final result is the average of the 4 prediction results. For the experiments the Leave-One-Out cross-validation strategy with 1 as the soft margin parameter for the SVM was applied. It means that for every classifier one example is left out and the classifier is trained over the other examples and tested on the selected one. This is repeated for every example.

⁴ <http://svmlight.joachims.org/>

⁵ <http://ontoware.org/swrc/>

Kernel	Error	Precision	Recall	F
<i>CCOP1</i>	6.88	85.32	46.07	59.83
<i>CCOP2</i>	6.04	90.70	48.78	63.44
<i>CCOPDP</i>	4.49	95.87	57.27	71.71
<i>RDFLearner1</i>	0.85	99.26	97.12	98.18
<i>RDFLearner2</i>	28.25	95.83	31.94	47.91

Table 1: Person2Affiliation experiment results

We use two configurations of our approach called *RDFLearner1* and *RDFLearner2*. For both, the optimization was performed with refinement based on 100 generations and 30 individuals in each subpopulation. We use the function measuring recall as the scoring function.

RDFLearner1 was learned from the full initial dataset. Its results in comparison to the baseline kernels are shown in Table 1. Due to dynamic induction of features, it found for instance *distinguishing features such as* that those *Person* instances that were selected for the positive examples were also instances of type *Employee*. This feature was found by the learner almost in all cases. With the help of dynamic weighting, *RDFLearner1* was able to give more importance to that feature relative to others, resulting in higher precision and recall as shown in Table 1.

For *RDFLearner2*, we deliberately excluded two features from the dataset, namely the *Employee* class membership and *affiliation* relation. These are crucial features which have strong impact in the SVM learned by *RDFLearner1*. Omitting these, the results of *RDFLearner2* were considerably poorer, recall in particular. However, by adopting a different combination of features and weights for the modified dataset, *RDFLearner2* still provide relatively high precision, as shown in Table 1.

Thus, the prediction quality offered by *RDFLearner* is good, and *RDFLearner* outperformed the baseline when the complete dataset was used. Still, we like to emphasize the major difference of the two approaches is that the baseline classifier [1] uses predefined features and weights. This method is simple and can show good results but is applicable only when sufficient domain and expert knowledge is available. In fact, these weights and kernels have to be defined and optimized manually for every classification scenario. Both features and weights are learned automatically in our approach.

4.2 Context-aware Movie Recommendation

As a second experiment, we provide a more comprehensive evaluation of *RDFLearner* on a scenario of context-aware movie recommendations and our experiments are based on the anonymized Filmtipset⁶ dataset. It contains information about the user ratings of movies, the time of the ratings, the people starring in movies, directors, user profile information like home country, age, gender, and movie specific information like genre, comments etc. We randomly select 150 users (out of 1000) from this dataset and the movie ratings (in our case *likes* or *dislikes*, which indicate the positive and negative examples), the movie information, location, and user profile information. This subset is divided into a training set

⁶ <http://www.filmtipset.se/>

with around 80 percent and a test set with around 20 percent of the examples. We convert these data to obtain a RDF representation.

We are interested in predicting the predicate $like(?user, ?movie)$.

Accuracy. First, we focus on the prediction performance under different parameter settings. One way to measure this is using the accuracy defined as $ACC = \frac{(TP+TN)}{(P+N)}$ where TP are true positives, TN - true negatives, P - all positives and N - all negatives. Thus, prediction result has high accuracy when it contains few false positives and false negatives. Figure 3-a plots the accuracy values in percent against different settings we used for the co-evolution, i.e., for different numbers of generations and population sizes. We can observe that accuracy increased both with the number of generations and population size. However, at a certain point, incrementing these numbers does not yield much improvement. We consider this point to be domain and dataset specific and should be found out with experimental observations.

Accuracy Compared to Standard SVM. In addition, at each iteration of the optimization algorithm, we replace the genetic refinement and mutation with a standard SVM classifier for calculating the SVM coefficients. Against this baseline, we compared prediction quality as well as running time. The configuration we used for the following experiments is: 100 generations and 50 population size. The results shown in Figure 3-b indicate that accuracy of both approaches increased with the size of the dataset. We noticed that they generated different coefficients and different numbers of support vectors. Standard SVM aims to find a global optimum of the data points, resulting in a smaller number of support vectors compared to our approach. Despite this, accuracy results of both approaches were quite similar, indicating that our approach was good at finding optimized coefficients.

Running Time Compared to Standard SVM. However, differences in these approaches take influence on the running time. We distinguished the running time of training from running time of prediction. The training time of RDFLearner and the SVM baseline can be observed in Figure 3-c. These results clearly show that applying the heuristics in the co-evolution has significant improvements over standard SVM training. This effect is pronounced when the dataset is large. However, standard SVM is slightly faster at prediction, as shown in Figure 3-d. This is due to the smaller number of support vectors used by the SVM, which means less complex computation is required for prediction.

5 Related Work

The use of learning methods on semantic data receives attention in recent years. Approaches presented in [4, 3, 1] employ kernels to learn from semantic data by using state-of-the-art techniques. Mainly, these approaches discuss relevant features in semantic data, and based on them, specify the kernels to be used by the classifier. In comparison to these approaches, we provide the most generic notion of kernel based on clauses that might capture arbitrary structures (and features). Features and their weights to be used in the kernel are learned and optimize automatically.

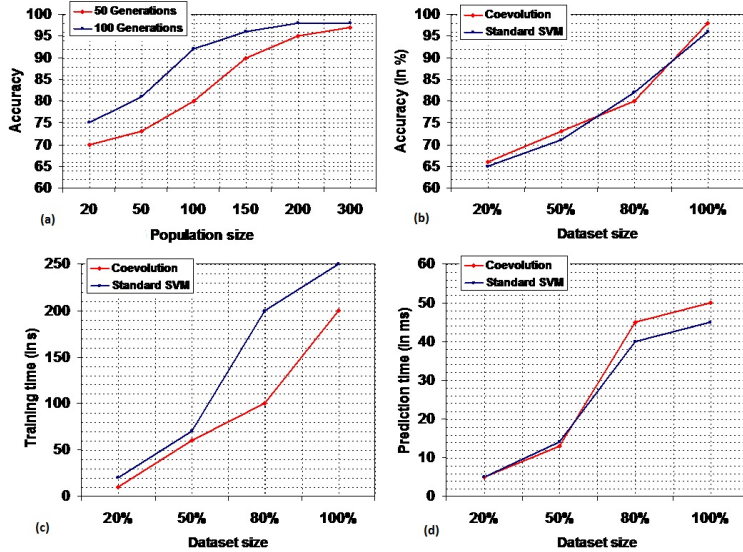


Fig. 3: Experiment Results regarding a) Accuracy against generation and population size, b) accuracy with co-evolution and SVM training of coefficients, c) training time , and d) prediction time

Another related field of research involves the use of SRL techniques [7, 11]. In [7] the authors propose to extend SPARQL with mining support based on the use of existing SRL approaches such that both deductive and inductive reasoning can be employed for query answering. Also, in [11], it has been shown how ontological information can be used to train a probabilistic model with prior knowledge. The drawback of SRL approaches in general is the need to construct a graphical model (e.g. Bayesian or Markov network). This is costly and might not be affordable when dealing with large networks like semantic data on the Web.

In this regard, using kernels in the ILP setting has been shown to perform well, especially when the dimensions of the feature space are unknown. A comprehensive survey discussing the relationship between the kernel-based techniques and ILP is presented in [5]. One prominent example that combines kernel methods and ILP is kFOIL, which utilizes kernel-based learning with hypothesis induction [8]. Though it is conceptually similar to our approach in the sense of using a combination of logical learning and kernel methods, it has been applied to the problem of ILP whereas we use this combination for SVM learning. Further, the applicability of this approach is limited in the semantic data setting, both in terms of the employed kernel and efficiency of optimization. In particular, kFOIL employs a kernel based on logical entailment whereas we compute similarities by considering the surrounding of nodes in the data graph (as captured by a clause). In addition, kFOIL employs an iterative algorithm that revises the hypothesis and trains an SVM in each iteration – this is recognized to be highly inefficient [8]. In our approach, we solve this issue by mapping the optimization problem to kernel learning and utilize a coevolutionary algorithm that searches hypotheses and SVM coefficients simultaneously.

6 Conclusion

We presented the concept of Relational Kernel Machines which is able to automatically identify and optimize features and their weights for learning kernels and a SVM from graph-structured semantic data. It employs ILP-based dynamic propositionalization to compute relevant features by searching for a hypothesis which serves as a description of features. This description is further decomposed into clauses, for which we learn R-convolution kernels, each can be seen as a composite kernel with sub-kernels focusing on some feature dimensions captured by the clauses. In turn, these clause kernels are combined and used as the basis for our kernel machine. The resulting problem is basically multi-objective optimization, as hypothesis and the kernel learning are dependent processes that have to be considered simultaneously. We propose the use of a coevolution algorithm to deal with this. Besides the fact that our approach does not require expert knowledge for finding and tuning features and weights, the experiments showed that our approach outperformed the baseline SVM approach which relies on manually defined kernels.

References

1. S. Bloehdorn and Y. Sure. Kernel methods for mining instance data in ontologies. In *ISWC/ASWC*, pages 58–71, 2007.
2. N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor. On kernel target alignment. *Innovations in Machine Learning*, pages 205–256, 2006.
3. C. d’Amato, N. Fanizzi, and F. Esposito. Classification and retrieval through semantic kernels. In *KES (3)*, pages 252–259, 2008.
4. N. Fanizzi, C. d’Amato, and F. Esposito. Learning with kernels in description logics. In *ILP*, pages 210–225, 2008.
5. P. Frasconi and A. Passerini. Learning with kernels and logical representations. *Probabilistic inductive logic programming*, pages 56–91, 2008.
6. D. Haussler. Convolution kernels on discrete structures (Technical Report UCSC-CRL-99-10). *University of California, Santa Cruz*, 1999.
7. C. Kiefer, A. Bernstein, and A. Locher. Adding data mining support to sparql via statistical relational learning methods. In *ESWC*, pages 478–492, 2008.
8. N. Landwehr, A. Passerini, L. De Raedt, and P. Frasconi. kFOIL: Learning simple relational kernels. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 389. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.
9. J. Quinlan. Learning logical definitions from relations. *Machine learning*, 5(3):239–266, 1990.
10. A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Proceedings of the 24th international conference on Machine learning*, pages 775–782. ACM, 2007.
11. A. Rettinger, M. Nickles, and V. Tresp. Statistical relational learning with formal ontologies. In *ECML/PKDD (2)*, pages 286–301, 2009.