Kernel Methods for Mining Instance Data in Ontologies*

Stephan Bloehdorn[1] and York Sure[2]

[1] Institute AIFB, University of Karlsruhe,
D-76128 Karlsruhe, Germany
bloehdorn@aifb.uni-karlsruhe.de
[2] SAP AG, Research Center CEC Karlsruhe,
Vincenz-Prießnitz-Str. 1, D-76131 Karlsruhe, Germany
york.sure@sap.com

Abstract. The amount of ontologies and meta data available on the Web is constantly growing. The successful application of machine learning techniques for learning of ontologies from textual data, i.e. mining *for* the Semantic Web, contributes to this trend. However, no principal approaches exist so far for mining *from* the Semantic Web. We investigate how machine learning algorithms can be made amenable for directly taking advantage of the rich knowledge expressed in ontologies and associated instance data. Kernel methods have been successfully employed in various learning tasks and provide a clean framework for interfacing between non-vectorial data and machine learning algorithms. In this spirit, we express the problem of mining instances in ontologies as the problem of defining valid corresponding kernels. We present a principled framework for designing such kernels by means of decomposing the kernel computation into specialized kernels for selected characteristics of an ontology which can be flexibly assembled and tuned. Initial experiments on real world Semantic Web data enjoy promising results and show the usefulness of our approach.

1 Introduction

The standardization of the ontology languages RDF(S) [1] and OWL [2] has led to an ever increasing amount of available semantic annotations. As of August 2007, the statistics of the Semantic Web search engine *Swoogle*[1] count a total of 1,238,295 publicly available "error-free pure Sematic Web Documents". Research has actively addressed the problem of learning knowledge structures – mostly from text data – *for* the Semantic Web, a field commonly referred to as *Ontology Learning* [3]. Only little work has been directed towards the question on learning *from* Semantic Web data and principled approaches are still missing. We strongly believe that mining Semantic Web data will become a crucial issue to deal with the massively growing amount of Semantic Web data. The need for mining Semantic Web data may arise in the context of two scenarios.

* This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

[1] http://swoogle.umbc.edu/

K. Aberer et al. (Eds.): ISWC/ASWC 2007, LNCS 4825, pp. 58–71, 2007.

On the one hand, we expect that it will be tempting to extend the scope of all kinds of adaptive systems we already find on the current Web (such as personal recommendation systems) from the still dominating document (text) data to Semantic Web data. On the other hand, classifiers that act on semantic web data can complement the cripsp reasoning procedures of current Semantic Web type systems possibly leading to hybrid systems where logic-based reasoning procedures interact with machine learning techniques inspired by statistical notions.

In this paper, we investigate the question how machine learning algorithms can be made amenable to work on instances that are described by means of an ontological vocabulary and how we can exploit the rich knowledge encoded in the respective ontologies. We will do so by placing the problem of learning from Semantic Web data in the context of the field of *kernel methods*. Kernel methods (see e.g. [4] for excellent and comprehensive introductions) are one of the most prominent paradigms in modern machine learning research. While Support Vector Machines can safely be regarded as the best known kernelized learning algorithm, many other well-known supervised and unsupervised machine learning algorithms can be kernalized as well. The main idea behind kernel-based learning algorithms is that they express the learned hypothesis by means of linear combinations of a specific type of similarity functions, the so-called kernel functions, where one argument is fixed to some training data item. An intriguing property of kernel-based learning algorithms is that kernel functions need not be restricted on vector-type data as arguments but can be defined directly on data items of arbitrary type as long as some mild restrictions are ensured. This allows to directly work on heterogenous and interconnected data that does not have a natural vector-style representation. Rephrasing the earlier statement we can thus say that this paper investigates how machine learning algorithms can be made amenable for taking advantage of the knowledge expressed in ontologies and associated metadata *by designing kernel functions defined on instances that reference certain ontologies*. We introduce a framework for kernel design on Semantic Web-type data that (i) builds on common notions of similarity, (ii) ensures the validity of the kernel(s) regardless of parameter choices of the user.

This paper is organized as follows: we introduce a number of preliminaries on kernel methods as well as knowledge representation and review related work in Section 2. In Section 3, we introduce our framework for kernel design on ontological instances. We illustrate the instantiation of this framework in Section 4 in the context of two experiments based on the SWRC and GALEN ontologies. We conclude in Section 5.

2 Preliminaries

In this section we shortly review the main preliminary notions necessary for the understanding of the main technical contributions of the paper. We give a brief overview of kernel methods and of knowledge representation with ontologies in sections 2.1 and 2.2 respectively. The interested reader is pointed to [4] for comprehensive introductions to the field of kernel methods and to [5,6] for introductions to ontologies and knowledge representation, in particular description logic-type approaches.

2.1 Machine Learning with Kernel Methods

Kernel methods are powerful machine learning techniques that have widespread adoption in the machine learning community and have been shown to be powerful learning algorithms in various standard and non-standard learning settings. The major paradigm behind kernel methods is the decoupling of the employed learning algorithms from the representations of the data instances under investigation. Different learning algorithms for various tasks can be "kernelized" such as, for example, Support Vector Machines [7] for classification and regression (i.e. supervised learning tasks) or Kernel-kMeans and Kernel-PCA [8] for clustering and dimensionality reduction (i.e. unsupervised learning tasks). In the "unkernalized" variant, these algorithms operate on simple vectors of real numbers. The hypotheses generated by these algorithms are typically tied to a geometric interpretation within the corresponding vector space such as the notion of a separating hyperplane in the case of classification with Support Vector Machines. By virtue of the design of the algorithms of interest, the input vectors need not be accessible directly by them. Instead, it is sufficient that they are able to access the evaluations of the inner product $\langle x, y \rangle$ of two vectors x, y in this space. The resulting hypotheses are then expressed using linear combinations of the input objects. In the context of this paper, we omit too much technical detail on this approach, also referred the *dual* representation of the respective algorithms.

The second component of kernel methods is the so-called *kernel function*. The kernel function computes the similarity of data instances in such a way that it is equivalent to an inner product in some (possibly unknown) vector space.

Definition 1 (Kernel Function). *Any function* $\kappa : X \times X \rightarrow \mathbb{R}$ *that for all* $x, z \in X$ *satisfies* $\kappa(x, z) = \langle \phi(x), \phi(z) \rangle$, *is a valid kernel, whereby X is some input domain under consideration and* ϕ *is a mapping from X to some inner product space F, called the feature space.*

As such, the computations of the plain inner product in the unkernlaized algorithms can be replaced by any valid kernel function. The kernel function, which can be regarded as a function that encodes a particular notion of similarity of data items of the input domain, simultaneously serves three purposes: (i) it provides the interface between the learning algorithm and the data, which is particularly interesting for data items that do not take the traditional form of vectors, (ii) it can leverage the performance of the algorithm by incorporating prior knowledge about the problem domain, and (iii) its evaluation might be computationally advantageous compared to an explicit construction of the feature space in terms of memory and/or computation requirements. The kernel function as such thus becomes an interesting subject of research. However, restrictions apply on the choice of the employed function to make it a valid kernel, namely, that the function needs to be a positive semi-definite function. Constructing appropriate positive semi-definite kernels from arbitrary data is thus not a trivial task. However, several closure properties aid the construction of valid kernels from known valid kernels, which will also be exploited in the construction of kernels in section 3. In particular, kernels are closed under sum, product, multiplication by a positive scalar and combination with well-known kernel modifiers.

Well-known kernel modifiers for a valid kernel $k(x, y)$ on some input set $x, y \in \mathcal{X}$ are, among others: (i) the normalisation kernel which scales the kernel results to $[0, 1]$ (in analogy with the cosine of two vectors):

$$k_{normalised}(x, y) = \frac{k(x, y)}{\sqrt{k(x, x)k(y, y)}};$$

or (ii) the Gaussian kernel:

$$k_{gaussian}(x, y) = \exp\left(-\frac{k(x, x) - 2k(x, y) + k(y, y)}{2\sigma^2}\right), \quad \sigma \in \mathbb{R}^+.$$

As we will often be occupied with dealing with sets of objects we introduce here two well-known kernels defined on sets $A, B \subseteq \mathcal{X}$ of data items $x \in \mathcal{X}$: (i) the intersection kernel

$$k_{\cap}(A, B) = |A \cap B| ;$$

and (ii) the crossproduct kernel:

$$k_{\times}(A, B) = \sum_{x_a \in A} \sum_{x_b \in B} k_{base}(x_a, x_b) ,$$

where $k_{base}(\cdot, \cdot)$ is any valid kernel defined on $\mathcal{X}$. In particular, note that the crossproduct kernel boils down to the intersection kernel when used with the so-called matching kernel:

$$k_{matching}(x_i, x_j) = \delta(x_i, x_j)$$

where $\delta(x_i, x_j) = 1$ if $x_i = x_j$ and $\delta(x_i, x_j) = 0$ otherwise. Proofs for the presented kernel closure properties, kernel modifiers and set kernels are omitted but can for example be found in [4].

2.2 Ontologies and Knowledge Representation

Ontologies provide a shared and common understanding of a domain of discourse and provide a vocabulary to describe data instances. A popular knowledge representation formalism is the family of Description Logics (DLs) which allow to model the relevant properties of a domain by means of classes (unary predicates), which denote sets of individuals, object properties (binary predicates), which denote binary relationships between individuals, and datatype properties (binary predicates) which denote relations between individuals and specific datatypes. Terminological axioms in the ontology can combine and relate classes and roles by the use of various concept and role constructors. Assertional axioms in associated knowledge bases make statements about the instances of the domain. The semantics of a certain description logic is typically given as *model theoretic semantics* by relating the syntax of the logic and the models of a domain.

The practical importance of description logics stems from the fact that they form the basis of the Web Ontology Language OWL[2]. In particular, the sublanguage OWL DL is based on the description logic $\mathcal{SHOIN}(\mathbf{D})$ for which a number of practical reasoning algorithms are known and implemented. As an adopted Semantic Web standard, OWL

is increasingly used to define ontologies and these ontologies – in turn – are increasingly used to describe data instances such as people, publications, services and the like.

The kernels presented in the next section will be based on probing instances for membership in certain classes or properties, during this process, deductions made using terminological axioms may enrich the explicitely asserted facts with deducted facts. During the remainder of this paper it is important to keep in mind that, due to the open world reasoning of DLs, the deduced facts are only the necessary facts. Other facts may or may not hold but they may just not be deduced necessarily.

2.3 Related Work

Research on learning algorithms for data that comes in a logic-based representation is mostly rooted in the field of *Inductive Logic Programming (ILP)* [9]. ILP algorithms directly exploit the underlying logic and search for logical concept descriptions that accuratly describe the concept(s) hidden in the data. As a positive side-effect, the resulting concept descriptions can thus be directly incorporated in the overall knowledge representation framework. As such they do, however, mostly completely circumvent any statistical notions such that softer hidden concepts are hard to identify.

Research on kernels for structured data, i.e. for data that is expressed in a formalism different from the standard vectorial representation, is a young research field that has recently become a major topic of investigation in the machine learning community. A good and fairly recent overview of basic types of kernels for structured data can be found in [10]. In the following, we will shortly describe the main works related to our research. Most of the work on kernels for structured data is rooted in the idea of the *convolution kernel* [11]. The main argument behind this kernel is that the semantics of composite objects can often be based on the semantics of its parts. The kernel thus aims to first evaluate basic kernels on the combinations of parts and sum the respective comparisons in the overall kernel, a direction that we will also investigate in this paper. Recently, [12] proposed a logic-based kernel on individuals represented as (closed) terms in a typed higher-order logic. Neglecting the technicalities of the approach this logic essentially allows the construction of complex types such as sets or lists out of other types, including standard types as e.g. natural numbers. The work presents a kernel defined on terms in the associated logic. However, the kernel is not capable of incorporating intensional background knowledge. In different spirit, [13] have proposed a kernel on Prolog proof trees. In this setting, the individuals are described in the context of global background knowledge in first-order logic. The idea of this kernel is then to measure the similarity of two individuals by means of the similarity of the proof trees of a special logic progam, called the *visitor program*, which probes certain characteristics of the individuals that may be of interest for the domain. In contrast to other approaches, this kernel allows to exploit background knowledge in a principled way. However, as acknowledged by the authors, the design of appropriate visitor programs is not necessarily intuitive and the proof trees may easily contain unnecessary noise that may hurt the overall performance.

3 Designing Kernels for Instance Data

In this section, we look at transfering kernel design principles to Semantic Web data. We introduce a principled framework for defining kernel on instances that are described with respect to some ontology. We can view this framework as a toolbox for formulating adequate kernels on Semantic Web data. The framework we describe is generic and we believe that it is capable of capturing most of the interesting future scenarios. As explained earlier, kernels should capture the similarity of the arguments. Along the common lines of interpretation of similarity, we will considering two instances the more similar, the more common (or similar) characteristics they have.

Before digging into the definitions of the kernel components in the next two sections let us fix some notation. We will implicitly define the kernels with respect to some fixed ontology $\mathcal{O}$. We will denote the set of named individuals in $\mathcal{O}$ as $\mathcal{I}$, the set of atomic classes $C \subseteq \mathcal{I}$ as C, the set of object properties $p \subseteq \mathcal{I} \times \mathcal{I}$ as $\mathcal{P}_O$ and the set of datatype properties $p \subseteq \mathcal{I} \times \text{dom}(d)$ as $\mathcal{P}_\mathcal{D}$.

We now introduce four layers of similarity of instances, each of which is defined using a particular (set- or cross product-) kernel. The model is roughly inspired by the layer model of [14] but the layers follow a different breakdown of the overall instance similarity as summarized in the following. The *identity layer* solely considers the identity of two instances. The *class layer* considers similarities of instances based on the classes the arguments can be shown to instantiate. The *property layers* consider similarities of instances based on the data properties and/or object properties the arguments can be shown to instantiate.

3.1 Identity Layer Kernel

On this layer we employ a particularly simple kernel, the identity kernel, which basically performs a binary check on the identity of the two arguments.

Definition 2 (Identity Kernel). *Given two instances x, z and an ontology $\mathcal{O}$, we define the identity kernel as: $k_{identity}(x, z) := \delta(x, z)$ with $\delta(x, z) = 1$ if $\mathcal{O} \models (x \equiv z)$ and $\delta(x, z) = 0$ otherwise.*

We immediately note the correspondence to the matching kernel such that the kernel will typically evaluate to 0 unless the it is evaluated on an instance with itself or the ontology contains an explicit equivalence axiom for the two argument instances.

3.2 Kernels on the Class Layer

We see the class(es) two individuals instantiate as a basic building block for their comparison. Intuitively, this type of similarity is useful only in those cases where there is some variation in the classes that are instantiated.

Definition 3 (Common Class Kernel). *Given two instances x, z and an ontology $\mathcal{O}$, we define the common class kernel as: $k_{class}(x, z) := |\{C \in C \mid \mathcal{O} \models C(x)\} \cap \{C \in C \mid \mathcal{O} \models C(z)\}|$.*

The class intersection kernel is a valid kernel, as it is a specific instantiation of the set intersection kernel. It can be interpreted as easily by defining the mapping $\phi(\cdot)$ as a mapping into a vector space whose dimensions correspond to the atomic classes defined in the ontology. The kernel value is the higher the more atomic classes are instantiated by the argument instances at the same time, if there is no common class the kernel will be zero. We can extend this kernel by means of a weighting scheme $\mu : C \rightarrow \mathbb{R}^+$ as follows:

Definition 4 (Weighted Common Class Kernel). *Given two instances x, z and an ontology O, we define the weighted common class kernel as:*

$$k_{class'}(x,z) := \sum_{c \in \{C \in C \mid O \models C(x)\} \cap \{C \in C \mid O \models C(z)\}} \mu(c).$$

3.3 Kernels on the Data Property Layer

As a next building block for determining the similarity of two individuals we consider properties. The general structure of this kernel is a sum of kernel evaluations of all compatible pairings of properties. We begin by defining kernels for data properties as follows.

Definition 5 (Data Property Kernel). *Given two instances x, z, a data property $p \in \mathcal{P}_\mathcal{D}$ and an ontology O, we define the data property kernel as:*

$$k^p_{DP}(x,z) := \sum_{\{d \mid O \models p(x,d)\}} \sum_{\{e \mid O \models p(z,e)\}} k_p(d,e)$$

whereby k_p is a valid kernel on the datatype referenced by the respective datatype property.

Again, the kernel property can be easily verified as the kernel is an instantiation of the crossproduct kernel. The kernel makes use of an underlying base kernel that is defined on the respective *datatype* to which the computes the similarity of pairs of data. For basic datatypes, such as strings or numeric values, a variety of useful kernels have been defined. For String datatypes, we may for example consider the well-known bag-of-words type kernels or the String Kernel introduced in [15]. For numerical values, the Gaussian Kernel defined on real numbers might be a useful choice. Again, we refer to [4] for information on such kernels.

3.4 Kernels on the Object Property Layer

In the same spirit as with the datatype property kernel, we define the object property kernels.

Definition 6 (Object Property Kernel). *Given two instances x, z, an object property $p \in \mathcal{P}_O$ and an ontology O, we define the object property kernels as crossproduct kernels:*

$$k^p_{OP}(x,z) := \sum_{\{v \mid O \models p(x,v)\}} \sum_{\{w \mid O \models p(z,w)\}} k_p(v,w)$$

$$k^{p}_{OP'}(x,z) := \sum_{\{v \mid O \models p(v,x)\}} \sum_{\{w \mid O \models p(w,z)\}} k_p(v,w)$$

whereby k_p is a valid kernel on instance data.

This definition requires two further remarks. First, note that for each object properties we have to define two separate kernels, depending on whether the argument instances should be considered as the source or the target of the object property. Second, the definition in terms of the crossproduct kernel again makes reference to an underlying base kernel k_p, this time defined on *instances* just as the overall kernel itself. While being powerful and flexible, this approach can be tricky as we need to avoid cycles during the computation of the kernel. A cycle arises if a kernel defined at a higher level is used again as a kernel it directly or indirectly depend on. In essence we therfore require that the kernel reference graph conforms to a tree structure. As an alternative, the matching kernel can be used as base kernel, such that the object property kernel essentially boils down to an intersection kernel requiring no further dependencies.

3.5 Integrated Kernel Calculation

We have so far presented isolated building blocks for kernel calculations on individuals such as kernels comparing the class structure or the property extensions of the instances. All valid kernel combination operators, in particular products or any form of weighted addition would be possible to combine the results. In our view, an weighted additive combination appears to be the most intuitive approach. Similar to the weights $\mu(C)$ for classes, the weight parameters can be used to tune the contribution of the corresponding kernels. Typically, only a small set of properties will be considered in the combined kernel with all other property kernels having weights equal to 0.

4 Illustrations and Evaluations

In this section we aim to show how the kernel framework introduced in this paper can be applied to real world Semantic Web data. We have implemented basic versions of the kernels introduced in the last section based on the ontology management and reasoning infrastructure *KAON2* [2]. Our implementation, the *KAON2Similarity API*[3] provides a number of instantiations of basic kernels together with supporting infrastructure such as kernel modifiers, kernel aggregators and a caching module that can be flexibly plugged together for the purpose at hand. We have further developed an JNI-based extension of the *SVMlight* Support Vector Machine software [16] that allows to deal with the kernels available via *Kaon2Similarity*[4].

As the task of learning from Semantic Web data has not yet been addressed actively, a comparative evaluation is not possible due to the lack of standardised evaluation data sets and baseline approaches. In fact, we aim at building a repository of standardized data sets for this purpose and see our experiments reported in this section as a first step

[2] http://kaon2.semanticweb.org/

[3] http://kaon2similarity.ontoware.org/

[4] http://www.aifb.uni-karlsruhe.de/WBS/sbl/software/jnikernel

Table 1. Galen Experiments: Statistics and Results

Number of overall positive Instances		
Galen class	$O_{original}$	O_{weak}
Biological_entity	550	63
Physical_entity	587	11
Complex	116	88
Continuant_entity	600	96
Discrete_entity	433	78
Mass_entity	70	16
Occurrent_entity	95	0

Results on Galen Test Set				
Galen class	err	prec	rec	F_1
Biological_entity	0.03	1.00	0.96	0.98
Physical_entity	0.02	0.99	0.96	0.97
Complex	0.01	1.00	0.93	0.96
Continuant_entity	0.02	1.00	0.97	0.98
Discrete_entity	0.01	1.00	0.98	0.99
Mass_entity	0.00	1.00	1.00	1.00
Occurrent_entity	0.02	1.00	0.75	0.85

into this direction. In this section, we thus introduce examples and results of two learning problems based on typical Semantic Web data sets. Both problems enjoy promosing results and, even though they are comparatively simple, they illustrate the approach very well. In future work, we intend to widen such experiments to larger-scale settings.

4.1 Common Class Kernel for the GALEN Ontology

We first illustrate the use of machine learning classification using only the simple common class kernel. For this purpose, we performed some experiments with the OWL DL version of the GALEN Upper Ontology[5]. The ontology contains atomic 175 classes, together with restrictions arranged in 193 SubClassOf axioms, 51 EquivalentClasses axioms, 127 DisjointClasses axioms and no Nominals. The idea of the experiment was to simply immitate the classification behaviour of the ontology given a semantically weakend ontology. We used 7 sublasses of the Self_standing_entity, namely the Biological_entity, Physical_entity, Complex, Continuant_entity, Discrete_entity, Mass_entity and Occurent_entity classes[6]. As only the TBox of the ontology is available, we randomly populated the ontology with 1000 individuals. We then filtered the ontology axoims and retained only SubClassOf and ClassMember axioms to obtain a weaker and semantically different ontology O_{weak}. Table 1 summarizes the entailment statistics for the 7 classes for the original and the changed version of the ontology.

We then split the overall instance set in two groups of 500 instances for training and testing with class labels assigned according to the semantics of $O_{original}$. We trained (and tested) a Support Vector machine using the common class kernel (with a normalisation modifier) only using the information present in O_{weak}. The results of the test runs are also reported in table 1. As the results show, the common class kernel is easily able to immitate the reasoning behaviour of the complex ontology on a weaker – semantically different – ontology. These consistent positive results can probably be explained by the fact that the weaker ontology still contains clear and crisp pattern for certain Class (sub-) structures that can be detected easily by the learning algorithm. Nevertheless they show a potential for learning logical patterns in a statistical manner.

[5] `http://www.cs.man.ac.uk/~rector/ontologies/simple-top-bio/`

[6] The Non_Biological_entity, Organelle and Non_Physical_entity were left out as they are either complements of existing concepts or did not have enough positive examples.

4.2 Mixed Object Property Kernels for the SWRC Ontology

We built a second experiment around the SWRC ontology [17] and the metadata available within the Semantic Portal of the Institute AIFB. The SWRC ontology initially grew out of the activities in the KA2 project and is now applied in a number of projects also outside of AIFB. The ontology has been ported to various knowledge representation languages including OWL[7].

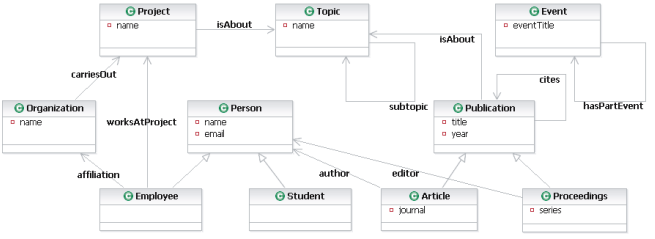

Fig. 1. Main classes and properties of the SWRC Ontology

The SWRC ontology generically models key entities relevant for typical research communities and the relations between them. The current version of the ontology comprises a total of 53 concepts in a taxonomy and 42 object properties, 20 of which are participating in 10 pairs of inverse object properties. All entities are enriched with additional annotation information. SWRC comprises at total of six top level concepts, namely the Person, Publication, Event, Organization, Topic and Project concepts. Figure 1 shows a small portion of the SWRC ontology with its main top-level concepts and relations. Since its initial versions, the SWRC ontology has been used and adapted in a number of different settings, most prominently for providing structured metadata for web portals. These include the web portal of the authors' institute AIFB[8]. The November 2006 version of the AIFB metadata comprises an additional set of 2,547 instances. 1,058 of these can be deduced to belong to the Person class 178 of which have an affiliation to one of the institute's groups (the others are external co-authors) with 78 of these being currently employed research staff. 1232 instances instantiate the Publication class, 146 instances instantiate the ResearchTopic class and 146 instances instantiate the Project class. The instances are connected by a total of 15,883 object property axioms and participate in a total of 8,705 datatype properties.

Given the information in the SWRC ontology, we have designed two classification problems that we have experimentally evaluated, namely the assignment of instances of the Person and Paper classes to one of the four research groups they are affiliated with (or, in the case of papers, where one of the authors is affiliated with.). Note that the information on the affiliations in the AIFB Portal is maintained by the Institute's administration and can thus be considered a very clean learning problem. On the other hand, the data about people's interests and projects or about paper metadata maybe noisy, or

[7] http://ontoware.org/projects/swrc/

[8] http://www.aifb.uni-karlsruhe.de/about.html

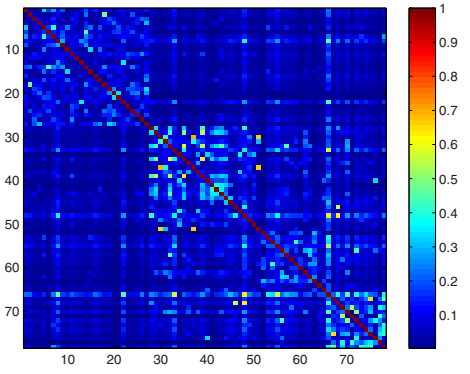

Fig. 2. Visualization of the (normalised) kernel matrix of 78 active AIFB researchers using the k_{cttp}. The researchers are ordered according to their research group affiliations.

Table 2. Leave-one-out classification results for the *persons2affiliation* problem on SWRC for different kernels and kernel modifiers. All numbers are percentages.

persons2affiliation, $c = 1$				
kernel config	err	prec	rec	F_1
sim-ctp-p	6.88	85.32	46.07	59.83
sim-ctp-pc	5.48	93.09	53.64	68.06
sim-ctp-pgc	6.32	94.86	43.28	59.45
sim-ctpp-p	6.04	90.70	48.78	63.44
sim-ctpp-pc	4.49	95.83	58.13	72.37
sim-ctpp-pgc	5.90	96.43	45.14	61.49
sim-ctpp-star-pc	4.49	95.87	57.27	71.71

persons2affiliation, $c = 10$				
kernel config	err	prec	rec	F_1
sim-ctp-p	7.72	78.10	43.10	55.55
sim-ctp-pc	6.46	86.87	49.77	63.29
sim-ctp-pgc	6.18	86.25	51.61	64.58
sim-ctpp-p	7.87	72.59	51.65	60.36
sim-ctpp-pc	5.20	90.90	54.22	67.92
sim-ctpp-pgc	5.34	86.62	57.16	68.87
sim-ctpp-star-pc	4.92	89.83	57.11	69.83

inconsistent because this kind of data is maintained auonomously by the researchers. For each of the two tasks, we have employed a couple of kernels. Obviously, the used kernels did not rely directly on any of the target properties as the learning problem would be trivial otherwise.

For the *person2affiliation* task, we designed a set of three kernels. The *sim-ctp* kernels correspond to a kernel combining the common class similarity kernel (with a small weight, 0.1) and the workedOnBy and worksAtProject object properties, pointing to (or from) associated publications, research topics and projects, each with weight 1. The *sim-ctpp* kernels additionally use the publication object property, also with weight 1. In all these cases, the object property kernels are designed as matching kernels. As a variant to this, the *sim-ctpp-star* kernel used an embedded bow cosine kernel on the publications title filed for the publication object property. The modifiers *p*, *pc* and *pgc* indicate wheter the respective sub-kernels where simply summed up, individually normalised and summed up or summed up and normalised afterwards, respectively. For the *papers2affiliation* task, we also designed a set of three kernels. As a kind of basline, we report results on the *sim-t* kernel, which corresponds to the bow cosine kernel on the title datatype propery. The *sim-cta* kernels combine the common class kernel (again with

Table 3. Leave-one-out classification results for the *papers2affiliation* problem on SWRC for different kernels and kernel modifiers. All numbers are percentages.

papers2affiliation, $c = 1$				
kernel config	err	prec	rec	F_1
sim-t-p	7.49	86.78	47.83	61.67
sim-cta-p	0.69	99.68	95.10	97.34
sim-cta-pc	0.69	99.79	94.52	97.09
sim-cta-pg1	6.84	96.39	56.97	71.62
sim-cta-pg3	1.38	99.59	90.15	94.63
sim-cta-pgc	0.85	99.44	94.28	96.79
sim-ctap-p	0.77	99.82	94.68	97.18
sim-ctap-pc	0.73	99.45	94.52	96.92
sim-ctap-pg1	7.63	95.93	53.14	68.40
sim-ctap-pg3	1.38	99.59	90.28	94.70
sim-ctap-pgc	0.91	99.33	94.39	96.80
sim-ctat-p	0.61	99.71	95.41	97.51
sim-ctat-pc	0.75	99.65	94.48	97.00

papers2affiliation, $c = 10$				
kernel config	err	prec	rec	F_1
sim-t-p	6.21	91.11	63.08	74.55
sim-cta-p	0.63	99.74	95.22	97.43
sim-cta-pc	0.63	99.17	94.95	97.02
sim-cta-pg1	6.09	96.77	61.23	75.01
sim-cta-pg3	0.69	99.79	94.91	97.29
sim-cta-pgc	0.58	99.03	95.68	97.33
sim-ctap-p	0.69	99.71	95.03	97.31
sim-ctap-pc	0.71	98.81	94.92	96.83
sim-ctap-pg1	6.63	96.34	60.11	74.03
sim-ctap-pg3	0.77	99.56	94.58	97.01
sim-ctap-pgc	0.67	98.80	95.48	97.11
sim-ctat-p	0.61	99.74	95.38	97.51
sim-ctat-pc	0.77	99.68	94.46	97.00

weight 0.1), with the isAbout and author object properties (each with weight 1) pointing to associated topics and authors. The *sim-ctap* kernels additionally consider the hasProject property whereas the The *sim-ctat* kernels additionally consider the title datatype property, again in conjunction with the bow cosine kernel. Again, we have employed different kernel modifiers, this time additionally the *pg1* and *pg3* kernel modifiers which correspond to the gaussian modifiers applied to the plain sum with parameters σ equal to 1 or 3.

As an example, Figure 2 visualizes the kernel matrix (the matrix of all kernel evaluations on all pairs) for the case of the *sim-ctpp-p* kernel in the *person2affiliation* task. Table 2 and table 3 illustrate the macro-averaged results of the classification experiments, estimated via the Leave-One-Out cross-validation strategy, for two different choices of the SVM soft margin parameter c. Not surprisingly, the *papers2affiliations* task has achieved virtually optimal results that are stable over the different kernel variants. These good results can be traced to the fact that the object properties pointing to associated authors have been included in the kernel computation, inherently bearing a strong correspondence to the research groups.

5 Conclusion

In this paper, we have investigated the question how kernel methods can be used to directly apply machine learning algorithms on Semantic Web-type instance data. This problem has to the best of our knowledge only been considered to a minor extend. We believe that this type of learning will become increasingly important in future research both from the machine learning as well as from the Semantic Web communities.

We have introduced a generic and principled framework for designing valid kernels on this type of input data that also allows to implicitly exploit the schema-level knowledge encoded in the ontology. This approach is in line with other work on kernels for

structured data [10,12]. We have presented some initial experiments in which we have exemplified the instantiation of the proposed framework for the specific purpose by classifying researchers to their respective research groups based on information represented by the well-known SWRC ontology and associated instance data and in a scenario using the GALEN upper ontology. While simple and artifically posed, the experiments enjoyed promising results and illustrated the overall approach.

Judged by the aim, namely mining data described by means of logical predicates, this research brings in some connections to Inductive Logic Programming (ILP) and Multi-Relational Data Mining [9]. While the results will always be conceptually similar to results achieved by adequate propositionalization approaches [18], the use of kernels avoid extensive pre-processing efforts and the definition of adequate kernels or the choice of adequate kernel components as a notion of similarity may often be more intuitive to the involved users.

A challenge for future work will, however, be to design means to aid the user in the choice of the various kernels, kernel modifiers and parameters. One such research trail is the automated optimization of kernel parameters such as the weights for kernel components for example by means of kernel alignment techniques [19]. As another line for future work, we aim at augmenting our framework by other kernel types that can exploit the relation structure of an instance space such as diffusion kernels [20]. Such kernels can also help to finding means for a more intuitive approach to avoiding cyclic kernel definitions. As much of the research in Description Logics is occupied with the properties of complex class descriptions, yet a different research trail would be the consideration of DL-type class descriptions as input for kernel-based learning algorithms. Last but not least, these initial conceptual results need to be carried over to the emerging Semantic Web application domains.

References

1. Brickley, D., Guha, R.: RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, 10 February 2004, Published online (2004), at `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`
2. McGuinness, D.L., van Harmelen, F.: OWL web ontology language overview. W3C Recommendation, 10 February 2004, Published online (2004), at `http://www.w3.org/TR/2004/REC-owl-features-20040210/`
3. Buitelaar, P., Cimiano, P., Magnini, B.: Trends in Information Processing Systems. Frontiers in Artificial Intelligence, vol. 123, p. 180. IOS Press, Amsterdam (2005)
4. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis (Hardcover). Cambridge University Press, Cambridge (2004)
5. Staab, S., Studer, R.: Handbook on Ontologies. Springer, Aix-en-Provence, France (2003)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
7. Vapnik, V., Golowich, S.E., Smola, A.J.: Support vector method for function approximation, regression estimation and signal processing. In: NIPS, pp. 281–287 (1996)
8. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany (1996)

9. Muggleton, S.H., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming 19,20, 629–679 (1994)
10. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explorations 5(1), 49–58 (2003)
11. Haussler, D.: Convolution kernels on discrete structures. Technical Report Technical Report UCS-CRL-99-10, UC Santa Cruz (1999)
12. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and distances for structured data. Machine Learning 57(3), 205–232 (2004)
13. Raedt, L.D., Passerini, A.: Kernels on prolog proof trees: Statistical learning in the ILP setting. Journal of Machine Learning Research 7, 307–342 (2006)
14. Ehrig, M., Haase, P., Stojanovic, N., Hefke, M.: Similarity for ontologies - a comprehensive framework. In: ECIS 2005. Proceedings of the 13th European Conference on Information Systems, Regensburg, Germany, May 26-28, 2005 (2005)
15. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. Journal of Machine Learning Research 2, 419–444 (2002)
16. Joachims, T.: Making large-scale SVM learning practical. In: Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge (1999)
17. Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., Oberle, D.: The SWRC ontology - semantic web for research communities. In: Jajodia, S., Mazumdar, C. (eds.) ICISS 2005. LNCS, vol. 3803, pp. 218–231. Springer, Heidelberg (2005)
18. Krogel, M.-A., Rawles, S., Zelezny, F., Flach, P., Lavrac, N., Wrobel, S.: Comparative evaluation of approaches to propositionalization. In: Horváth, T., Yamamoto, A. (eds.) ILP 2003. LNCS (LNAI), vol. 2835, pp. 194–217. Springer, Heidelberg (2003)
19. Cristianini, N., Shawe-Taylor, J.: On kernel-target alignment. In: Advances in Neural Information Processing Systems 14 - Proceedings of NIPS 2001, Vancouver, Canada, December 3-8, 2001, pp. 367–373 (2001)
20. Kondor, R.I., Lafferty, J.D.: Diffusion kernels on graphs and other discrete input spaces. In: ICML 2002. Proceedings of the Nineteenth International Conference on Machine Learning, pp. 315–322. Morgan Kaufmann, San Francisco (2002)