Statistical Learning for Inductive Query Answering on OWL Ontologies

Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito

Dipartimento di Informatica, Università degli Studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
{fanizzi,claudia.damato,esposito}@di.uniba.it

Abstract. A novel family of parametric language-independent kernel functions defined for individuals within ontologies is presented. They are easily integrated with efficient statistical learning methods for inducing linear classifiers that offer an alternative way to perform classification w.r.t. deductive reasoning. A method for adapting the parameters of the kernel to the knowledge base through stochastic optimization is also proposed. This enables the exploitation of statistical learning in a variety of tasks where an inductive approach may bridge the gaps of the standard methods due the inherent incompleteness of the knowledge bases. In this work, a system integrating the kernels has been tested in experiments on approximate query answering with real ontologies collected from standard repositories.

1 Ontology Mining: Learning from Metadata

In the context of the Semantic Web (henceforth SW) many applications require the accomplishment of data-intensive tasks that can effectively exploit machine learning methods [1]. However, while a growing amount of metadata is being produced, most of the research effort addresses the problem of learning *for* the SW (mostly from structured or unstructured text [2]). Less attention was devoted to the advantages (and problems) of learning *from* SW data and metadata expressed in *Description Logics* (DLs) [3].

Classification is a central task for many applications. However, classifying through logic reasoning may be both too demanding because of its complexity and also too weak because of inconsistency or (inherent) incompleteness in the knowledge bases [4]. So far, for the sake of tractability, only simple DL languages have been considered in the development of logic-based learning methods [5, 6]. On the other end, efficient machine learning methods, that were originally developed for simple data, can be effectively upgraded to work with richer structured representations [7]. These methods have been shown to effectively solve *unsupervised* and *supervised* learning problems in DLs [8, 9], particularly those based on classification, clustering and ranking of individuals.

A. Sheth et al. (Eds.): ISWC 2008, LNCS 5318, pp. 195–212, 2008.

Although the inductive methods that will be presented are general and could in principle be exploited in various scenarios, we will focus on methods for inducing efficient classifiers from examples and use them to carry out forms of approximate query answering (and concept retrieval). This task is normally performed by recurring to standard deductive reasoning procedures [3]. Hence it may turn out to be ineffective when (inconsistent or) incomplete knowledge is available, which is not infrequent with heterogeneous and distributed data sources.

As discussed in previous works [9], besides of approximated retrieval and query answering, alternative classification methods can be as effective as deductive reasoning, even suggesting new knowledge (membership assertions) that was not previously logically derivable. As an example, considering the well-known WINE ontology, a statistical classifier induced by machine learning methods presented in the following, is able to infer assertions that cannot be logically derived by a reasoner such as that KathrynKennedyLateral, which is known as a Meritage, is a CaliforniaWine and an AmericanWine as well as that CotturiZinfandel, which is only known as a Zinfandel, is not a CabernetSauvignon (a non-disjoint sibling class). This feature of inductive classifiers can be exploited during the time-consuming *ontology completion* task [10] since the knowledge engineer has only to validate such assertions.

Among the other learning methods, *kernel methods* [11] represent a family of very efficient algorithms, that ultimately solve linear separation problems (finding an optimal hyperplane in between positive and negative instances) in high-dimensional feature spaces whereto a kernel function implicitly maps the original feature space of the considered dataset (*kernel trick*). *Ad hoc* kernel functions allow for learning classifiers even when the instances are represented in rich languages.

In this work, we demonstrate the exploitation of a kernel method for inducing classifiers for individuals in OWL ontologies. Indeed, kernel functions have been recently proposed for languages of average expressiveness, such as the family of kernels for $\mathcal{ALC}$ [12, 13]. However, the scope of their applicability was limited because of two factors: the definition in terms of a normal form for concept descriptions and the employment of the notion of (approximations of) *most specific concepts* [3] in order to lift instances to the concept-level where the kernels actually work.

In order to overcome such limitations, we propose a novel parametric family of kernel functions for DL representations which is inspired to a semantic pseudo-metric for DLs [8]. These functions encode a notion of similarity between individuals, by exploiting only semantic aspects of the reference representation. Their definition is also related to other simple kernels that were recently proposed [14]. Yet, while each of these kernels acts separately on a different level of similarity [15], based on the concepts and properties of the ontology, ours may integrate these aspects being parametrized on a set of features (concept descriptions). Furthermore, these features are not fixed but may be induced enforcing the discernibility of different instances. Similarly to metric-learning procedures based on stochastic search [8], a method for optimizing the choice of the feature sets is also proposed. This procedure, based on genetic programming, can be exploited in case the concepts in the ontologies would turn out to be weak for *discriminative* purposes.

The basics of kernel methods are presented in the next section jointly with related works about kernels for complex representations. In Sect. 3 the new family of kernels is proposed together with an algorithm for optimizing the choice of its parameters. Then, the query answering problem and its solution through our inductive method are formally defined in Sect. 4 and experimentally evaluated in Sect. 5. Conclusions and further applications of ontology mining methods are finally outlined in Sect. 6.

2 Inducing Classifiers with Kernel Methods

Given the learning task of inducing classifiers from examples, kernel methods are particularly well suited from an engineering point of view because the learning algorithm (*inductive bias*) and the choice of the kernel function (*language bias*) are almost completely independent [1]. While the former encapsulates the learning task and the way in which a solution is sought, the latter encodes the hypothesis language, i.e. the representation for the target classes. Different kernel functions implement different hypothesis spaces (representations). Hence, the same kernel machine can be applied to different representations, provided that suitable kernel functions are available. Thus, an efficient algorithm may be adapted to work on structured spaces [7] (e.g. trees, graphs) by merely replacing the kernel function with a suitable one. Positive and negative examples of the target concept are to be provided to the machine that processes them, through a specific kernel function, in order to produce a definition for the target concept in the form of a decision function based on weights.

2.1 Learning Linear Classifiers with Kernel Methods

Most machine learning algorithms work on simple representations where a training example is a vector of boolean features x extended with an additional one y indicating the membership w.r.t. a target class: $(\boldsymbol{x}, y) \in \{0,1\}^n \times \{-1,+1\}$. Essentially these algorithms aim at finding a vector of coefficients $\boldsymbol{w} \in \mathbb{R}^n$ which is employed by a linear function (i.e. a *hyperplane* equation) to make a decision on the y label for an unclassified instance $(\boldsymbol{x}, \cdot)$:

$$\mathrm{class}(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w} \cdot \boldsymbol{x})$$

if $\boldsymbol{w} \cdot \boldsymbol{x} \geq 0$ then predict $\boldsymbol{x}$ to be positive $(+1)$ else it is classified as negative (-1).

As an example, the Perceptron is a well-known simple algorithm to learn such weights [1]. In the training phase, for each incoming training instance, the algorithm predicts a label according to mentioned decision function and compares the outcome with the correct label. On erroneous predictions, the weights $\boldsymbol{w}$ are revised depending on the set of examples that provoked the mistake (denoted by

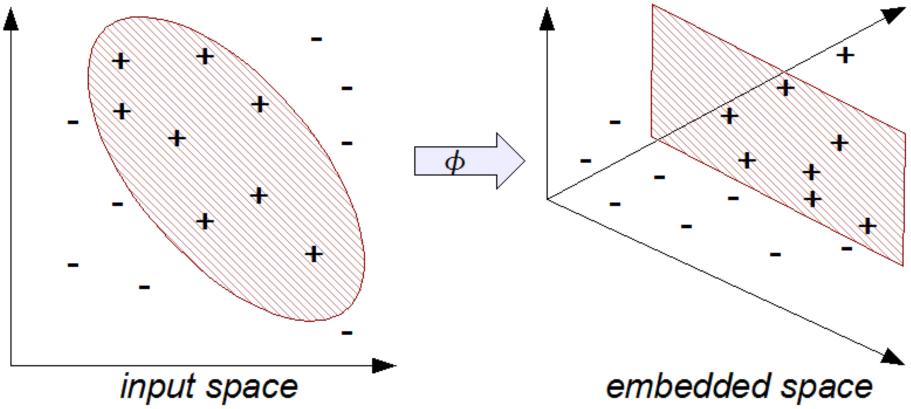

Fig. 1. The idea of the kernel trick

M): $\boldsymbol{w} = \sum_{\boldsymbol{v} \in M} l(\boldsymbol{v})\boldsymbol{v}$, where function l returns the label of the input example. Then, the resulting decision function can be written $\boldsymbol{w} \cdot \boldsymbol{x} = \sum_{\boldsymbol{v} \in M} l(\boldsymbol{v})(\boldsymbol{v} \cdot \boldsymbol{x})$. The dot product in these linear functions is the common feature of these methods.

Separating positive from negative instances with a linear boundary may be infeasible as it depends on the complexity of the target concept [1]. The *kernel trick* consists in mapping the examples onto a suitable different space (likely one with many more dimensions), allowing for the linear separation between positive and negative examples (*embedding space*, see Fig. 1). For example, the decision function for the perceptron becomes: $\sum_{\boldsymbol{v} \in M} l(\boldsymbol{v})(\phi(\boldsymbol{v}) \cdot \phi(\boldsymbol{x}))$, where ϕ denotes the transformation. Actually, such a mapping is never explicitly performed; a *valid* (i.e. definite positive) kernel function, corresponding to the inner product of the transformed vectors in the new space, ensures that an embedding exists [11]: $k(\boldsymbol{v}, \boldsymbol{x}) = \phi(\boldsymbol{v}) \cdot \phi(\boldsymbol{x})$. For instance, the decision function above becomes (*kernel perceptron*): $\sum_{\boldsymbol{v} \in M} l(\boldsymbol{v})k(\boldsymbol{v}, \boldsymbol{x})$.

Any algorithm for learning linear classifiers which is ultimately based on a decision function that involves an inner product could in principle be adapted to work on non-linearly separable cases by resorting to valid kernel functions which implicitly encode the transformation into the embedding space. Even more so, often many hyperplanes can separate the examples. Among the other kernel methods, the *support vector machines* (SVMs) aim at finding the hyperplane that maximizes the *margin*, that is the distance from the areas containing positive and negative training examples. The classifier is computed according to the closest instances w.r.t. the boundary (support vectors).

These algorithms are very efficient (polynomial complexity) since they solve the problem through quadratic programming techniques once the kernel matrix is produced [11]. The choice of kernel functions is very important as their computation should be efficient enough for controlling the complexity of the overall learning process.

2.2 Kernels for Structured Representations

When examples and background knowledge are expressed through structured (logical) representations, a further level of complexity is added. One way to solve the problem may involve the transformation of statistical classifiers into logical ones. However, while the opposite mapping has been shown as possible (e.g. from DL knowledge bases to artificial neural networks [16]), direct solutions to the learning problem are still to be investigated.

An appealing quality of the class of valid kernel functions is its closure w.r.t. many operations. In particular this class is closed w.r.t. the *convolution* [17]:

$$k_{\mathrm{conv}}(x, y) = \sum_{\substack{\overline{x} \in R^{-1}(x) \\ \overline{y} \in R^{-1}(y)}} \prod_{i=1}^{D} k_i(\overline{x}_i, \overline{y}_i)$$

where R is a composition relationship building a single compound out of D simpler objects, each from a space that is already endowed with a valid kernel. Note that the choice of R is a non-trivial task which may depend on the particular application.

Then new kernels can be defined for complex structures based on simpler kernels defined for their parts using the closure property w.r.t. this operation. Many definitions have exploited this property, introducing kernels for strings, trees, graphs and other discrete structures. In particular, [7] provide a principled framework for defining new kernels based on type construction where types are defined in a declarative way.

While these kernels were defined as depending on specific structures, a more flexible method is building kernels as parametrized on concepts described with another representation. Such kernel functions allow for the employment of algorithms, such as the SVMs, that can simulate feature generation. These functions transform the initial representation of the instances into the related active features, thus allowing for learning the classifier directly from structured data. As an example, Cumby & Roth propose kernels based on a simple DL representation, the *Feature Description Language* [18].

Kernels for richer DL representations have been proposed in [12]. Such functions are actually defined for comparing $\mathcal{ALC}$ concepts based on the structural similarity of the AND-OR trees corresponding to a normal form of the input concept descriptions. However these kernels are not only structural since they ultimately rely on the semantic similarity of the primitive concepts (on the leaves) assessed by comparing their extensions through a set kernel. Although this proposal was criticized for possible counterintuitive outcomes, as it might seem that semantic similarity between two input concepts were not fully coped with, the kernels are actually applied to couples of individuals, after having lifted them to the concept level by means of (approximations of) their *most specific concept* [3]. Since these concepts are constructed on the grounds of the same ABox and TBox, it is likely that structural and semantic similarity tend to coincide.

A more recent definition of kernel functions for individuals in the context of the standard SW representations is reported in [14]. The authors define a set of kernels for individuals and for the various types of assertions in the ABox (on concepts, datatype properties, object properties). However, it is not clear how to integrate such functions which cope with different aspects of the individuals; the preliminary evaluation on specific classification problems regarded single kernels or simple additive combinations.

3 A Family of Kernels for Individuals in DLs

In the following we report the basic DL terminology utilized for this paper (see [3] for a thorough and precise reference).

Ontologies are built on a triple $\langle N_C, N_R, N_I \rangle$ made up by a set of concept names N_C, a set of role names N_R and a set of individual names N_I, respectively. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps (via $\cdot^{\mathcal{I}}$) such names to the corresponding element subsets, binary relations, and objects of the domain $\Delta^{\mathcal{I}}$. A DL language provides specific constructors and rules for building complex concept descriptions based on these building blocks and for deriving their interpretation. The *Open World Assumption* (OWA) is made in the underlying semantics, which is convenient for the SW context.

A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. $\mathcal{T}$ is the set of terminological axioms of concept descriptions $C \sqsubseteq D$, meaning $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, where C is the concept name and D is its description. $\mathcal{A}$ contains assertions on the world state, e.g. $C(a)$ and $R(a, b)$, meaning that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Subsumption w.r.t. the models of the knowledge base is the most important inference service. Yet in our case we will exploit *instance checking*, that amounts to decide whether an individual is an instance of a concept [3].

The inherent incompleteness of the knowledge base under open-world semantics may cause reasoners not to be able to assess the target class-membership. Moreover this can be a computationally expensive reasoning service. Hence we aim at learning efficient alternative classifiers that can help answering these queries effectively.

3.1 Kernel Definition

The main limitations of the kernels proposed in [12] for the space of $\mathcal{ALC}$ descriptions are represented by the dependency on the DL language and by the approximation of the most specific concept which may be computationally expensive. The use of a normal form has been also criticized since this is more a structural (syntactic) criterion that contrasts notion of semantic similarity.

In order to overcome these limitations, we propose a different set of kernels, based on ideas that inspired a family of inductive distance measures [8, 9], which can be applied directly to individuals:

Definition 3.1 (DL-kernels). *Let* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *be a knowledge base. Given a set of concept descriptions* $\mathsf{F} = \{F_1, F_2, \ldots, F_m\}$, *a family of kernel functions* $k_p^{\mathsf{F}} : \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \mapsto [0, 1]$ *is defined as follows:*

$$\forall a, b \in \mathsf{Ind}(\mathcal{A}) \qquad k_p^{\mathsf{F}}(a,b) := \left[\sum_{i=1}^{m} \left| \frac{\kappa_i(a,b)}{m} \right|^p \right]^{1/p}$$

where $p > 0$ *and* $\forall i \in \{1, \ldots, m\}$ *the* simple concept kernel function κ_i *is defined:* $\forall a, b \in \mathsf{Ind}(\mathcal{A})$

$$\kappa_i(a,b) = \begin{cases} 1 & (F_i(a) \in \mathcal{A} \wedge F_i(b) \in \mathcal{A}) \vee (\neg F_i(a) \in \mathcal{A} \wedge \neg F_i(b) \in \mathcal{A}) \\ 0 & (F_i(a) \in \mathcal{A} \wedge \neg F_i(b) \in \mathcal{A}) \vee (\neg F_i(a) \in \mathcal{A} \wedge F_i(b) \in \mathcal{A}) \\ \frac{1}{2} & \textit{otherwise} \end{cases}$$

or, model-theoretically:

$$\kappa_i(a,b) = \begin{cases} 1 & (\mathcal{K} \models F_i(a) \wedge \mathcal{K} \models F_i(b)) \vee (\mathcal{K} \models \neg F_i(a) \wedge \mathcal{K} \models \neg F_i(b)) \\ 0 & (\mathcal{K} \models \neg F_i(a) \wedge \mathcal{K} \models F_i(b)) \vee (\mathcal{K} \models F_i(a) \wedge \mathcal{K} \models \neg F_i(b)) \\ \frac{1}{2} & \textit{otherwise} \end{cases}$$

The rationale for these kernels is that similarity between individuals is determined by their similarity w.r.t. each concept in a given committee of features. Two individuals are maximally similar w.r.t. a given concept F_i if they exhibit the same behavior, i.e. both are instances of the concept or of its negation. Conversely, the minimal similarity holds when they belong to opposite concepts. Because of the OWA, sometimes a reasoner cannot assess the concept-membership, hence, since both possibilities are open, we assign an intermediate value to reflect such uncertainty.

As mentioned, instance-checking is to be employed for assessing the value of the simple similarity functions. Yet this is known to be computationally expensive (also depending on the specific DL language of choice). Alternatively, especially for ontologies that are rich of explicit class-membership information (assertions), a simple look-up may be sufficient, as suggested by the first definition of the κ_i functions.

The parameter p was borrowed from the form of the Minkowski's measures [19]. Once the feature set is fixed, the possible values for the kernel function are determined, hence p has an impact on the granularity of the measure.

3.2 Discussion

The most important property of a kernel function is its validity (it must correspond to a dot product in a certain embedding space).

Proposition 3.1 (validity). *Given an integer* $p > 0$ *and a committee of features* F*, the function* k_p^{F} *is a valid kernel.*

This result can be assessed by proving the function k_p^{F} definite-positive. Alternatively it is easier to prove the property by showing that the function can be obtained by composing simpler valid kernels through operations that guarantee the closure w.r.t. this property [17]. Specifically, since the simple kernel functions κ_i ($i = 1, \ldots, n$) actually correspond to *matching kernels* [7], the property follows from the closure w.r.t. sum, multiplication by a constant and kernel multiplication [17].

One may note that such functions extend (and integrate) the kernels defined in [14]. For instance, the *common class* kernels may constitute a simplified version of the DL-kernels. They are essentially based on the intersection of the sets of common classes, considering only those occurring in the ontology. The new kernels, in principle, can be parametrized on any set of complex concept descriptions, including negated concepts. Moreover, they take also into account uncertain membership cases. As regards the data-property and object-property kernels, again the similarity is assessed by comparing (restrictions of) domains and ranges of defined relations related to the assertions on the input individuals. These may be encoded by further concepts to be added to the committee, especially when they can determine the separation of different individuals.

Furthermore, the uniform choice of the weights assigned to the various features in the sum ($1/m^p$) may be replaced by assigning different weights reflecting the importance of a certain feature in discerning the various instances. A good choice may be based on the amount of *entropy* related to each feature (then the weight vector has only to be normalized) [9].

It is worthwhile to note that this is indeed a family of kernels parametrized on the choice of features. Preliminary experiments regarding instance-based classification, demonstrated the effectiveness of the kernel using the very set of both primitive and defined concepts found in the knowledge bases. However, the choice of the concepts to be included in the committee F is crucial and may be the object of a preliminary learning problem to be solved (*feature selection*).

3.3 Optimizing the Feature Set

As for the pseudo-metric that inspired the kernel definition [8], a preliminary phase may concern finding an optimal choice of features. This may be carried out by means of randomized optimization procedures, similar to the one developed for the pseudo-distance. However, the integration of the algorithm in suitable kernel machines guarantees that the feature construction job is performed automatically by the learning algorithm (the features correspond to the dimensions of the embedding space).

The underlying idea in the kernel definition is that similar individuals should exhibit the same behavior w.r.t. the concepts in F. Here, one may make the assumption that the feature-set F represents a sufficient number of (possibly redundant) features that are able to discriminate different individuals (in terms of a discernibility measure).

Namely, since the function is strictly dependent on the committee of features F, two immediate heuristics arise:

- The *number* of concepts of the committee,
- Their discriminating power in terms of a *discernibility factor*, i.e. a measure of the amount of difference between individuals.

```
GPOptimization($\mathcal{K}$, maxGenerations, fitnessThr, FeatureSet)
input   $\mathcal{K}$: current knowledge base
        maxGenerations: maximal number of generations
        fitnessThr: minimal required fitness threshold
output FeatureSet: set of concept descriptions
static  currentFSs, formerFSs: arrays of feature sets
        currentBestFitness, formerBestFitness = 0: arrays of fitness values
        offsprings: array of generated feature sets
        fitnessImproved: improvement flag
        generationNo = 0: number of current generation
begin
currentFSs = makeInitialFS($\mathcal{K}$,INIT_CARD)
formerFSs = currentFSs
repeat
    fitnessImproved = false
    currentBestFitness = bestFitness(currentFSs)
    while (currentBestFitness < fitnessThr) and (generationNo < maxGenerations) do
        begin
        offsprings = generateOffsprings(currentFSs)
        currentFSs = selectFromPopulation(offsprings)
        currentBestFitness = bestFitness(currentFSs)
        ++generationNo
        end
    if (currentBestFitness > formerBestFitness) and (currentBestFitness < fitnessThr) then
        begin
        formerFSs = currentFSs
        formerBestFitness = currentBestFitness
        currentFSs = extendFS(currentFSs)
        end
    else
        fitnessImproved = true
    end
until not fitnessImproved
return selectBest(formerFSs)
end
```

Fig. 2. Feature set optimization algorithm based on genetic programming

Finding optimal sets of discriminating features, should also profit by their composition, employing the specific constructors made available by the representation language.

These objectives can be accomplished by means of randomized optimization techniques, especially when knowledge bases with large sets of individuals are available. For instance in [8] we have proposed a metric optimization procedures based on stochastic search. Namely, part of the entire data can be drawn in order to learn optimal feature sets, in advance with respect to the successive usage for all other purposes.

A specific optimization algorithm founded in *genetic programming* has been devised to find optimal choices of discriminating concept committees. The resulting algorithm is shown in Fig. 2. Essentially, it searches the space of all possible feature committees, starting from an initial guess (determined by the call to the makeInitialFS() procedure) based on the concepts (both primitive and defined) currently referenced in the knowledge base $\mathcal{K}$, starting with a committee of a given cardinality (INIT_CARD). This initial cardinality may be determined as a function of $\lceil \log_3(N) \rceil$, where $N = |\mathsf{Ind}(\mathcal{A})|$, as each feature projection can categorize the individuals in three sets.

The outer loop gradually augments the cardinality of the candidate committees until the threshold fitness is reached or the algorithm detects some fixpoint: employing larger feature committees would not yield a better feature set w.r.t. the best fitness recorded in the previous iteration (with fewer features). Otherwise, the EXTENDFS() procedure extends the current committee by including a newly generated random concept.

The inner while-loop is repeated for a number of generations until a stop criterion is met, based on the maximal number of generations maxGenerations or, alternatively, when a minimal fitness threshold fitnessThr is crossed by some feature set in the population, which can be returned.

As regards the BESTFITNESS() routine, it computes the best fitness of the feature sets in the input vector. Fitness can be determined as the *discernibility factor* yielded by the feature set, as computed on the whole set of individuals or on a smaller sample. For instance, given the fixed set of individuals $IS \subseteq \mathsf{Ind}(\mathcal{A})$ the fitness function may be:

$$\textsc{discernibility}(\mathsf{F}) := \nu \sum_{(a,b)\in IS^2} \sum_{i=1}^{|\mathsf{F}|} \mid 1 - \kappa_i(a,b) \mid$$

where ν is a normalizing factor that depends on the overall number of couples involved.

As concerns finding candidate sets of concepts to replace the current committee (the GENERATEOFFSPRINGS() routine), the function was implemented by recurring to some transformations of the current best feature sets:

- Choose $\mathsf{F} \in$ currentFSs;
- Randomly select $F_i \in \mathsf{F}$;
  - Replace F_i with $F_i' \in \textsc{randomMutation}(F_i)$ randomly generated, or
  - Replace F_i with one of its refinements $F_i' \in \textsc{ref}(F_i)$

The possible refinements of concept description are language-specific. E.g. for the case of $\mathcal{ALC}$ logic, refinement operators have been proposed in [6, 5].

This is iterated till a number of offsprings is generated (another parameter which determines the speed of the search process). Then these offspring feature sets are evaluated and the best ones are included in the new version of the currentFSs array; the best fitness value for these feature sets is also computed.

When the while-loop is over, the current best fitness is compared with the best one recorded for the former feature set length; if an improvement is detected then the outer repeat-loop is continued, otherwise (one of) the former best feature set(s) is selected and returned as the result of the algorithm.

4 Approximate Classification and Retrieval

SVMs based on kernel functions can efficiently induce classifiers that work by mapping the instances into an embedding feature space, where they can be discriminated by means of a linear classifier.

Given the kernel function for DLs defined in the previous section, we intend to use an SVM to induce a linear classifier which can be efficiently employed to solve the following problem:

Definition 4.1 (classification problem). *Let* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *be a knowledge base, let* $\mathsf{Ind}(\mathcal{A})$ *be the set of all individuals occurring in* $\mathcal{A}$ *and let* $\mathcal{C} = \{C_1, \ldots, C_s\}$ *be the set of (both primitive and defined) concepts in* $\mathcal{K}$.
The classification problem *can be defined as follows:*
given *an individual* $a \in \mathsf{Ind}(\mathcal{A})$,
determine $\{C_1, \ldots, C_t\} \subseteq \mathcal{C}$ *such that:* $\mathcal{K} \models C_i(a)\ \forall i \in \{1, \ldots, t\}$.

In the general setting of the kernel algorithms, the target classes for the classification problem are normally considered as disjoint. This is unlikely to hold in the SW context, where an individual can be an instance of more than one concept. Then, a different setting has to be considered. The multi-class classification problem is decomposed into smaller binary classification problems (one per class). Therefore, a simple binary value set ($V = \{-1, +1\}$) may be employed, where $+1$ indicates that an individual x_i is instance of the considered concept C_j and -1 indicates that x_i is not instance of C_j.

This multi-class learning setting is valid when an implicit *Closed World Assumption* (CWA) is made. Conversely, in a SW context, where the OWA is adopted, this is not sufficient because of the uncertainty brought by the different semantics. To deal with this peculiarity, the absence of information on whether a certain instance x_i belongs to the extension of the concept C_j should not be interpreted negatively; rather, it should count as neutral information. Thus, a larger valued set has to be considered, namely $V = \{+1, -1, 0\}$, where the three values denote, respectively, class-membership, non-membership and uncertain assignment. Hence, given a query instance x_q, for every concept $C_j \in C$, the classifier will return $+1$ if x_q is an instance of C_j, -1 if x_q is an instance of $\neg C_j$, and 0 otherwise.

The classification is performed on the grounds of the linear models built from a set of training examples whose correct labels are provided by an expert (or a reasoner). For each concept, classifiers for membership and non-membership have to be learned.

Dually, statistical classifiers can be used to perform an approximate retrieval service. Considered a knowledge base $\mathcal{K}$ and a query concept Q, a learning problem can be solved providing a limited set of individuals that are (examples) and are not (counterexamples) in the concept extension. The learning algorithm will produce a classifier for deciding the class-membership of other individuals; then all other individuals in $\mathcal{A}$ can be classified w.r.t. Q, thus solving the concept retrieval problem inductively.

The classifier is generally very efficient (simple mathematical computation is carried out). As regards the effectiveness (see also the next section), its performance on query answering or retrieval tasks may be compared to that of a logic reasoner. Moreover, the classifier may be able, in some cases, to answer queries when the reasoner cannot; that is the classifier may be able to induce knowledge that is likely to hold but that is not logically derivable. One may also consider using binary classifiers only, in order to force the answer to belong to $\{+1, -1\}$, or provide a measure of likelihood for this answer [9], yet this goes beyond the scope of this work.

5 Experimental Evaluation

The new kernel functions were implemented and integrated with the support vector machines in the LIBSVM library[1]. They can be easily integrated also in the SVM^{light} extension[2] proposed in [14]. The experimental session was designed in order to evaluate the learning method on a series of query answering problems.

5.1 Setup

A number of different OWL ontologies were selected from the Protégé library[3]: Newspaper, Wines, Surface-Water-Model (S.W.M.), Science, and New Testament Names (N.T.N.). Details about them are reported in Tab. 1 (upper part).

Table 1. Facts about the ontologies employed in the experiments

ontology	DL lang.	#concepts	#obj. prop.	#data prop.	#individuals
Newspaper	$\mathcal{ALCF}(D)$	29	28	25	72
S.W.M.	$\mathcal{ALCOF}(D)$	19	9	1	115
Wines	$\mathcal{ALCIO}(D)$	112	9	10	149
Science	$\mathcal{ALCIF}(D)$	74	70	40	331
N.T.N.	$\mathcal{SHIF}(D)$	47	27	8	676
BioPax	$\mathcal{ALCIF}(D)$	74	70	40	323
LUBM	$\mathcal{ALR}_+\mathcal{HI}(D)$	43	7	25	118
SWSD	$\mathcal{SHIF}(D)$	47	27	8	732
Financial	$\mathcal{ALCIO}(D)$	112	9	10	1000

For each ontology, all concepts in their turn were considered as queries. A ten-fold cross validation design[4] was adopted in order to overcome the variability in the composition of the training and test sets of examples. Examples were labeled according to the reasoner response; the classifier was then induced by the SVM exploiting the kernel matrix computed by the use of the DL-Kernel[5], for the subset of the training examples selected in each run of the experiment. The classifier was then tested on the remaining individuals assessing its performance with respect to the correct theoretical classification provided by the reasoner.

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm

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

[3] http://protege.stanford.edu/plugins/owl/owl-library

[4] The set of examples is randomly divided into ten parts then, in each fold, one part is used to validate the classifier induced using the instances in the other parts as training examples [1].

[5] The feature set for the DL-kernel was made by all concepts in the ontology and parameter p was set to 1 for simplicity and efficiency purposes.

A different setting has been considered in [14] with a simplified version of the GALEN ontology. There, the ontology was randomly populated and only seven selected concepts have been considered while no roles have been taken into account. We considered only populated ontologies with their genuine composition, with no change on their population. Differently from the mentioned experiments, the population was not randomly generated in order to avoid that the classifier resulting from the learning process were influenced by the specific generating algorithm.

The performance of the classifier was evaluated by comparing its responses on test instances to those returned by a standard reasoner[6] used as baseline. As mentioned, the experiment has been performed by adopting the ten-fold cross validation procedure. The results (percentages) presented in the following tables are averaged over the folds and over all the concepts occurring in each ontology. Particularly, for each concept in the ontology, the following parameters have been measured for the evaluation [9]:

- *match rate*: number of cases of individuals that got exactly the same classification by both classifiers with respect to the overall number of individuals;
- *omission error rate*: amount of unlabeled individuals while they actually were to be classified as instances or as counterexamples for the concept;
- *commission error rate*: amount of individuals labeled as instances of a concept, while they (logically) belong to the negation of that concept or vice-versa;
- *induction rate*: amount of individuals that were found to belong to a concept or its negation, while this information is not logically derivable by the reasoner.

The experiment is aimed at showing that statistical classification is comparably effective w.r.t. logic classification. Meanwhile it is very efficient (because of the simple linear function it is based on) and is also able to suggest (by analogy) assertions that are not logically derivable from the ontologies.

5.2 Outcomes

The outcomes of the experiments regarding the classification of all the concepts occurring in each ontology are reported in Tab. 2. By looking at the table, it is important to note that, for every ontology, the commission error was null. This means that the classifier did not make critical mistakes, i.e. cases when an individual is deemed to be an instance of a concept while it really is an instance of another disjoint concept. At the same time it is important to note that very high match rates were registered for each ontology. Particularly, it is interesting to observe that the match rate increases with the increase of the number of individuals in the considered ontology. This is because the performance of statistical methods is likely to improve with the availability of large numbers of training examples, which means that there is more information for better separating the example space.

[6] Pellet 1.5.1: `http://pellet.owldl.com`

Table 2. Results (average rates ± standard deviation) of the experiments on classification using the SVM with the DL-kernel

ontology	match	induction	omission	commission
NewsPaper	**90.3** ± 8.3	**0.0** ± 0.0	**9.7** ± 8.3	**0.0** ± 0.0
S.W.M.	**95.9** ± 4.1	**0.0** ± 0.0	**4.1** ± 4.1	**0.0** ± 0.0
Wines	**95.2** ± 8.8	**0.6** ± 5.2	**4.2** ± 7.5	**0.0** ± 0.0
Science	**97.1** ± 2.0	**1.8** ± 2.5	**1.1** ± 1.6	**0.0** ± 0.0
N.T.N.	**98.2** ± 1.7	**0.2** ± 0.9	**1.6** ± 1.6	**0.0** ± 0.0

Table 3. Results (average rates ± standard deviations) of the experiments on classification using the SVM with the $\mathcal{ALC}$ kernel ($\lambda = 1$)

ontology	match	induction	omission	commission
NewsPaper	**90.3** ± 8.3	**0.0** ± 0.0	**9.7** ± 8.3	**0.0** ± 0.0
S.W.M.	**87.1** ± 15.8	**6.7** ± 16.0	**6.2** ± 9.1	**0.0** ± 0.0
Wines	**95.6** ± 7.8	**0.4** ± 3.4	**4.0** ± 7.3	**0.0** ± 0.0
Science	**94.2** ± 7.8	**0.7** ± 7.8	**5.1** ± 7.8	**0.0** ± 7.8
N.T.N.	**92.5** ± 24.7	**2.6** ± 8.4	**0.1** ± 3.9	**4.7** ± 11.3

A conservative behavior has been also observed, indeed the omission error rate was not null (although it was very low). This was probably due to a high number of training examples classified as unknown w.r.t. certain concepts. To decrease the tendency to a conservative behavior of the method, a threshold could be introduced for the consideration of the training examples with an *unknown* classification.

In almost all cases, the classifier was able to induce class-membership assertions that were not logically derivable. For example, in the NTN ontology JesusChrist was found to be an instance of the concepts Man and $\overline{\mathsf{Woman}}$, while this could not be determined by deductive reasoning (it is known to be an instance of SonOfGod). However, the assessment of the quality of the induced knowledge is not possible because the correct answer to the inferred membership assertions is known by the experts that built and populated the ontologies.

The experiment has been repeated on the same ontologies, applying classifier induced using the SVM jointly with the $\mathcal{ALC}$ kernel [12, 13]. Since the languages of the ontologies are generally more complex than $\mathcal{ALC}$, we considered the individuals to be represented by approximations of the most specific concepts of such individuals w.r.t. the ABox [3]. Note that a separate random new ten-fold experiment was generated, hence the training / test subsets were different w.r.t. the previous run.

The outcomes of the experiments are reported in Tab. 3. By comparing the outcomes reported in the two tables, it is possible to note that the classifiers induced by the SVM with the new DL-kernel generally improve both match rate and omission rate with respect to the $\mathcal{ALC}$ kernel (in the cases where they do not improve the difference is not large). The observed induction rates are generally in favor of the classifiers induced with the $\mathcal{ALC}$ kernel. This can be explained with the higher precision of the classifiers induced by the DL-kernels, which increased the match rate in many cases when the reasoner was not able to give a certain classification. The commission rate for the experiments with the $\mathcal{ALC}$ kernel is null like in the experiments with the DL-kernel (but for one case). Finally, one may also observe that the outcomes of the classifiers induced by adopting the new kernel showed a more stable behavior as testified by the limited deviations reported in the tables (with some exceptions where the difference is limited).

5.3 Experiments on Query Answering

Another experimental session has been designed for evaluating the performance of the classifiers induced with the new kernels on solving query answering problems with randomly generated concepts.

Further larger ontologies were selected (see Tab. 1, lower part): the BioPax glycolysis ontology[7] (BioPax), an ontology generated by the Lehigh University Benchmark (LUBM), the Semantic Web Service Discovery dataset[8]

(SWSD) and FINANCIAL ontology[9] employed as a testbed for PELLET. This was to increase the diversity of the domain (as well as source and population) of the ontologies and to provide learning problems with many classified training instances (yet this also depends on the generality of query concepts).

Table 4. Results (average rates ± standard deviation) of the experiments on random query answering

ONTOLOGY	match	induction	omission	commission
S.W.M.	**82.31** ± 21.47	**9.11** ± 16.49	**8.57** ± 8.47	**0.00** ± 0.00
SCIENCE	**99.16** ± 4.35	**0.44** ± 3.42	**0.39** ± 2.76	**0.00** ± 0.00
N.T.N.	**80.38** ± 17.04	**8.22** ± 16.87	**9.98** ± 10.08	**1.42** ± 2.91
BIOPAX	**84.04** ± 14.55	**0.00** ± 0.00	**0.00** ± 0.00	**15.96** ± 14.55
LUBM	**76.75** ± 19.69	**5.75** ± 5.91	**0.00** ± 0.00	**17.50** ± 20.87
FINANCIAL	**97.85** ± 3.41	**0.42** ± 0.23	**0.02** ± 0.07	**1.73** ± 3.43
SWSD	**97.92** ± 3.79	**0.00** ± 0.00	**2.09** ± 3.79	**0.00** ± 0.00

Preliminarily, a number of individuals (30% of the entire number) was uniformly sampled; then the method for generating optimal feature sets was run for each ontology to better define the final kernel function (p was set again to 1). Random queries were also preliminarily generated for each ontology combining (2 through 8) atomic concepts or universal and existential restrictions (maximal depth 3), using the union and intersection operators. In order to be able to induce the classifier, the generated queries were required also to represent satisfiable concepts and that some individuals could be recognized as their examples and counterexamples.

The outcomes are reported in Tab. 4, from which it is possible to observe that the behavior of the classifier on these concepts is not very dissimilar with respect to the outcomes of the previous experiments. These queries were expected

[7] `http://www.biopax.org/Downloads/Level1v1.4/`

[8] `https://www.uni-koblenz.de/FB4/Institutes/IFI/AGStaab/Projects/xmedia/dl-tree.htm`

[9] `http://www.cs.put.poznan.pl/alawrynowicz/financial.owl`

to be harder than the previous ones which correspond to the very primitive or defined concepts for the various ontologies. Specifically, the commission error rate was low for all but two ontologies (BioPax and LUBM) for which some very difficult queries were randomly generated which raised this rate beyond 10% and consequently also the standard deviation values. The difficulty arose from the very limited number of training classified instances available for the target random concept (many unclassified training instances).

As for all methods that learn from examples, the number of positive and negative instances has an impact on the quality of the classifier, which is likely shown when their quality is assessed against the test set.

6 Conclusions and Future Work

Inspired from previous works on dissimilarity measures in DLs, a novel family of semantic kernel functions for individuals has been defined based on their behavior w.r.t. a number of features (concepts). The kernels are language-independent being based on instance-checking (or ABox look-up) and can be easily integrated with a kernel machine (a SVM in our case) for performing a broad spectrum of activities related to ontologies.

In this paper we focused on the application of statistical methods for inducing classifiers based on the individuals in an ontology. The resulting classifiers can be used to perform alternative classification and query answering in a more efficient yet effective way, compared with the standard deductive procedures. It has been experimentally shown that its performance is not only comparable to the one of a standard reasoner, but the classifier is also able to induce new knowledge, which is not logically derivable (e.g. by using a DL reasoner). Particularly, an increase in predictive accuracy was observed when the instances are homogeneously spread, as expected from statistical methods. The induced classifiers can be exploited for predicting / suggesting missing information about individuals, thus completing large ontologies. Specifically, it can be used to semi-automatize the population of an ABox. Indeed, the new assertions can be suggested to the knowledge engineer that has only to validate their acquisition.

This constitutes a new approach in the SW context, since the efficiency of the statistical-numerical approaches and the effectiveness of a symbolic representation have been combined [16]. As a next step, a more extensive experimentation of the proposed method has to be performed besides of a comparison with similar existing methods [9].

Further ontology mining methods can be based on kernels such as conceptual clustering which allows the discovery of interesting subgroups of individuals which may require the definition of a new concept or to track the drift of existing concepts over time (with the acquisition of new individuals) or even to detect new emerging concepts [8].

References

[1] Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

[2] Buitelaar, P., Cimiano, P., Magnini, B. (eds.): Ontology Learning from Text: Methods, Evaluation and Applications. IOS Press, Amsterdam (2005)

[3] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)

[4] Hitzler, P., Vrandečić, D.: Resolution-based approximate reasoning for OWL DL. In: Gil, Y., Motta, E., Benjamins, V., Musen, M.A. (eds.) Proceedings of the 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland. LNCS, vol. 3279, pp. 383–397. Springer, Heidelberg (2005)

[5] Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the Semantic Web. Applied Intelligence 26, 139–159 (2007)

[6] Lehmann, J., Hitzler, P.: Foundations of refinement operators for description logics. In: Blockeel, H., Ramon, J., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS (LNAI), vol. 4894, pp. 161–174. Springer, Heidelberg (2008)

[7] Gärtner, T., Lloyd, J., Flach, P.: Kernels and distances for structured data. Machine Learning 57, 205–232 (2004)

[8] Fanizzi, N., d'Amato, C., Esposito, F.: Randomized metric induction and evolutionary conceptual clustering for semantic knowledge bases. In: Silva, M., Laender, A., Baeza-Yates, R., McGuinness, D., Olsen, O., Olstad, B. (eds.) Proceedings of the ACM International Conference on Knowledge Management, CIKM 2007, Lisbon, Portugal, pp. 51–60. ACM, New York (2007)

[9] d'Amato, C., Fanizzi, N., Esposito, F.: Query answering and ontology population: An inductive approach. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 288–302. Springer, Heidelberg (2008)

[10] Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Veloso, M. (ed.) Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, pp. 230–235 (2007)

[11] Schölkopf, B., Smola, A.: Learning with Kernels. The MIT Press, Cambridge (2002)

[12] Fanizzi, N., d'Amato, C.: A declarative kernel for $\mathcal{ALC}$ concept descriptions. In: Esposito, F., Raś, Z.W., Malerba, D., Semeraro, G. (eds.) ISMIS 2006. LNCS (LNAI), vol. 4203, pp. 322–331. Springer, Heidelberg (2006)

[13] Fanizzi, N., d'Amato, C.: Inductive concept retrieval and query answering with semantic knowledge bases through kernel methods. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, pp. 148–155. Springer, Heidelberg (2007)

[14] Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 58–71. Springer, Heidelberg (2007)

[15] Ehrig, M., Haase, P., Hefke, M., Stojanovic, N.: Similarity for ontologies - a comprehensive framework. In: Proceedings of the 13th European Conference on Information Systems, ECIS 2005 (2005)

[16] Hammer, B., Hitzler, P. (eds.): Perspectives of Neural-Symbolic Integration. Studies in Computational Intelligence, vol. 77. Springer, Heidelberg (2007)

[17] Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Department of Computer Science, University of California – Santa Cruz (1999)

[18] Cumby, C., Roth, D.: On kernel methods for relational learning. In: Fawcett, T., Mishra, N. (eds.) Proceedings of the 20th International Conference on Machine Learning, ICML 2003, pp. 107–114. AAAI Press, Menlo Park (2003)

[19] Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search – The Metric Space Approach. In: Advances in database Systems. Springer, Heidelberg (2007)