Structural Similarity in Expressive Description Logics: An Extended Family of Kernels for OWL

Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito

LACAM – Dipartimento di Informatica, Università degli studi di Bari
{fanizzi|claudia.damato|esposito}@di.uniba.it

Abstract. In the context of the Semantic Web many applications of inductive inference ultimately rely on a notion of similarity for the standard knowledge representations of the ontologies. We have tackled the problem of statistical learning with ontologies proposing a family of structural kernels for $\mathcal{ALCN}$ that has been integrated with support vector machines. Here we extend the definition of the kernels to more expressive languages of the family, namely those backing OWL.

1 Concept Similarity in Ontologies

Although *machine learning* techniques may have a great potential for the Semantic Web (SW) applications, *ontology learning* tasks have focused mainly on methods for text [4]. Much less effort has been devoted to the application of machine learning methods to knowledge bases described in formal concept representations of the SW (formal *ontologies*) ultimately based on *Description Logics* (DL) [1].

In order to apply statistical learning some notion of similarity for the specific representation is needed [8]. It is particularly appealing to work with kernel methods that allow for decoupling the final learning algorithm (e.g. perceptrons, support vector machines, etc.) from the notion of similarity which depends on the particular instance space which is encoded in the kernel function. We have been investigating on the definition of kernel functions for instance spaces represented in relational languages such as those based on clauses and lately also on DLs. One of the first works concerns kernels for the *Feature Description Logic* [6] which proved particularly effective for relational structures elicited from text. More recently further proposals for working with more complex DL and ontology languages have been made [10, 2, 13].

In this work, we propose a family of declarative kernel functions that can be applied to DLs representations with different degrees of expressiveness. The kernels encode a notion of similarity of individuals in this representation, based on structural and semantic aspects of the reference representation. Specifically, we extend the definition of a family of kernels for the $\mathcal{ALCN}$ logic [12]. These kernel functions are based on both structural and also semantic aspects, namely a normal form and the extension of the concepts involved in the description, as elicited from the knowledge base.

As such the kernels are designed for comparing concept descriptions. In order to apply them to instance comparison w.r.t. real ontologies, that likely exploit the full extent of the most expressive DL languages, (upper) approximations of the most specific concepts that cover the single individuals had to be computed [5]. This exposes the method to a number of problems. Firstly, a (partially) structural normal form may fail to fully capture the semantic similarity between the individuals. Scaling up to more complex languages wold require specific normal forms. Moreover, the existence of a most specific concept is not guaranteed [1].

Coupling valid kernels with efficient algorithms such as the support vector machines, many tasks based on inductive classification can be tackled. Particularly, we demonstrate how to perform important inferences on semantic knowledge bases, namely concept retrieval and query answering. These tasks are generally grounded on merely deductive procedures which easily fail in case of (partially) inconsistent or incomplete knowledge. The methods has been shown to work comparably well w.r.t. standard reasoners [12, 13], allowing the suggestion of new knowledge that was not previously logically derivable.

Moreover, these kernels naturally induce distance measures which allow for extensions to further metric-based learning methods such as nearest-neighbor classification [7] and conceptual clustering [11]. However these extensions are beyond the scope of this work.

2 Preliminaries on Representation and Inference

The basics of the DLs will be briefly recalled. The reader may refer to the DL handbook [1] for a thorough reference. Such representations provide the basic constructors adopted by the standard ontology languages employed in the SW, such as the Ontology Markup Language OWL. We will focus on the $\mathcal{ALCQ}$ logic which may represent a tradeoff between expressiveness and reasoning efficiency.

2.1 Knowledge Bases in Description Logics

Let us consider a triple $\langle N_C, N_R, N_I \rangle$ made up. respectively, by a set of *primitive concept* names N_C, to be interpreted as sets of objects in a certain domain, a set of *primitive role* names N_R, to be interpreted as binary relationships between the mentioned objects, and a set of individual names N_I for the objects themselves.

The semantics of the descriptions is defined by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set, the *domain* of the interpretation, and $\cdot^{\mathcal{I}}$ is the *interpretation function* that maps each individual $a \in N_I$ to a domain object $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, each primitive concept name $A \in N_C$ to its *extension* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and for each $R \in N_R$ the extension is a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.

Complex descriptions can be built in $\mathcal{ALCQ}$ using the language constructors listed in Table 1, along with their semantics derived from the interpretation of atomic concepts and roles [1]. The *top* concept $\top$ is interpreted as the whole domain $\Delta^{\mathcal{I}}$, while the *bottom* concept $\bot$ corresponds to $\emptyset$. Complex descriptions can be built in $\mathcal{ALCQ}$ using the following constructors. The language supports

Table 1. Syntax and semantics of concepts in the $\mathcal{ALCQ}$ logic.

Name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
full concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
concept disjunction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}((x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}((x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$
qual. at least restriction	$\geq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \|\{y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\|\} \geq n\}$
qual. at most restriction	$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \|\{y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\|\} \leq n\}$

full negation: a concept negation $\neg C$ has an extension that amounts to the complement of $C^{\mathcal{I}}$ w.r.t. the domain. The *conjunction* and *disjunction* of two concepts are simply interpreted as the intersection and union of their extensions. Concepts can be also defined as restrictions on the roles. The *existential restriction* constrains the elements of its extension to be related via a certain role R to some instances of concept C while the *value* (or *universal*) *restriction* comprises those elements who are related through R only to instances of concept C (if any). Finally, qualified numeric restrictions define concepts by constraining its instances with the minimal or maximal number of instances of concept C related through R.

OWL offers further constructors that extend the expressiveness of this language. Its DL equivalent is $\mathcal{SHOIQ}(\mathbf{D})$ [1], that extends $\mathcal{ALCQ}$ with individual classes, role hierarchies, transitive and inverse roles ($\mathcal{I}$). Besides concrete domains ($\mathbf{D}$), i.e. well-founded external data types (such as numerical types, tuples of the relational calculus, spatial regions, or time intervals), can be dealt with.

The main inference employed with these representations is assessing whether a concept *subsumes* another concept based on their semantics:

Definition 2.1 (subsumption). *Given two descriptions C and D, C is* subsumed *by D, denoted by $C \sqsubseteq D$, iff for every interpretation $\mathcal{I}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. When $C \sqsubseteq D$ and $D \sqsubseteq C$ then they are* equivalent*, denoted with $C \equiv D$.*

Note that this naturally induces a generality relationship on the space of concepts.

Generally subsumption is not assessed in isolation but rather related to the models of a system of axioms representing the background knowledge and the world state:

Definition 2.2 (knowledge base). *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 $C \equiv D$, where C is the concept name and D is its description. $\mathcal{A}$ contains assertions on individuals $C(a)$ and $R(a,b)$.*

An interpretation that satisfies all its axioms is a *model* of the knowledge base.

Note that defined concepts should have a unique definition. However defining concepts through inclusions axioms ($C \sqsubseteq D$) is also generally admitted. Moreover, such definitions are assumed to be unfoldable.

Example 2.1 (royal family). This example shows a knowledge base modeling concepts and roles related to the British royal family:

$$
\begin{aligned}
\mathcal{T} = \{ \; & \texttt{Male} \equiv \neg\texttt{Female}, \\
& \texttt{Woman} \equiv \texttt{Human} \sqcap \texttt{Female}, \\
& \texttt{Man} \equiv \texttt{Human} \sqcap \texttt{Male}, \\
& \texttt{Mother} \equiv \texttt{Woman} \sqcap \exists\texttt{hasChild.Human}, \\
& \texttt{Father} \equiv \texttt{Man} \sqcap \exists\texttt{hasChild.Human}, \\
& \texttt{Parent} \equiv \texttt{Father} \sqcup \texttt{Mother}, \\
& \texttt{Grandmother} \equiv \texttt{Mother} \sqcap \exists\texttt{hasChild.Parent}, \\
& \texttt{Mother-w/o-daughter} \equiv \texttt{Mother} \sqcap \forall\texttt{hasChild.}\neg\texttt{Female}, \\
& \texttt{Super-mother} \equiv \texttt{Mother} \sqcap \geq 3\, \texttt{hasChild.Human} \; \}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{A} = \{ \; & \texttt{Woman(elisabeth)}, \texttt{Woman(diana)}, \texttt{Man(charles)}, \texttt{Man(edward)}, \\
& \texttt{Man(andrew)}, \texttt{Mother-w/o-daughter(diana)}, \\
& \texttt{hasChild(elisabeth, charles)}, \texttt{hasChild(elisabeth,edward)}, \\
& \texttt{hasChild(elisabeth, andrew)}, \texttt{hasChild(diana, william)}, \\
& \texttt{hasChild(charles, william)} \; \}
\end{aligned}
$$

Note that the *Open World Assumption* (OWA) is made in the underlying semantics, since normally knowledge representation systems are applied in situations where one cannot assume that the knowledge in the base is complete. Although this is quite different from the standard settings considered in machine learning and knowledge discovery, it is convenient for the Semantic Web context where new resources (e.g. Web pages, Web services) may be continuously made available.

2.2 Inference Services

The basic inference services for DL knowledge bases can be viewed as entailments as they amount to verifying whether a generic relationship is a logical consequence of the knowledge bases axioms.

Typical reasoning tasks for TBoxes regards the *satisfiability* of a concept, *subsumption*, *equivalence* or *disjointness* between two concepts. For example in the knowledge base reported in Ex. 2.1, Father is subsumed by Man and is disjoint with Woman. These inferences can be reduced to each of them, hence normally reasoners support only one of them. Reasoning is performed though *tableau* algorithms [9, 1].

Inference problems concerning ABoxes are mainly related to checking its *consistency*, that is, whether it has a model, especially w.r.t. those of the TBox. For example, an ABox like the one in Ex. 2.1 containing also the assertion Father(elisabeth) is not consistent.

Since we aim at crafting inductive methods that manipulate individuals, a prototypical inference is *instance checking* [1], that amounts to checking whether an assertion α is entailed by the knowledge base ($\mathcal{K} \models \alpha$). If α is a role assertion then it is quite easy to perform the inference as normally roles do not have a definition in the TBox. However, we will focus on deciding whether an individual is an instance of a given concept. This can be easily reduced to a consistency problem: $\mathcal{K} \models C(a)$ iff $\mathcal{K} \cup \{\neg C(a)\}$ is inconsistent.

The adopted open world semantics has important consequences on the way queries are answered. While the closed-world semantics identifies a database with a single model an ABox represents possibly infinitely many interpretations. Hence a reasoner might be unable to answer certain queries because it may be actually able to build models for both a positive ($\mathcal{K} \cup \{C(a)\}$) and a negative ($\mathcal{K} \cup \{\neg C(a)\}$) answer.

Example 2.2 (open world semantics).
Given the ABox is Ex. 2.1, while it is explicitly stated that diana is a Mother-w/o-daughter, it cannot be proven for elisabeth because she may have daughters that are simply not known in the knowledge base. Analogously, elisabeth is an instance of Super-mother while this cannot be concluded for diana as only two children are known in the current knowledge base. Although these instance checks fail because of the inherent incompleteness of the knowledge state, this does not imply that the individuals belong to the negated concepts, as could be inferred in case the CWA were made.

Another related inference is *retrieval* which consists in querying the knowledge base to know the individuals that belong to a given concept:

Definition 2.3 (retrieval). *Given an knowledge base $\mathcal{K}$ and a concept C, find all individuals a such that $\mathcal{K} \models C(a)$.*

A straightforward algorithm for a retrieval query can be realized via instance checking on each individual occurring in the ABox, testing whether it is an instance of the concept.

Dually, it may be necessary to find the (most specific) concepts which an individual belongs to. This is called a *realization problem.* One especially seeks for the most specific one (up to equivalence):

Definition 2.4 (most specific concept). *Given an ABox $\mathcal{A}$ and an individual a, the* most specific concept *of a w.r.t. $\mathcal{A}$ is the concept C, denoted $\mathsf{MSC}_{\mathcal{A}}(a)$, such that $\mathcal{A} \models C(a)$ and for any other concept D such that $\mathcal{A} \models D(a)$, it holds that $C \sqsubseteq D$.*

This is a typical way to lift individuals to the conceptual level [5].

For some languages, the MSC may not be expressed by a finite description [1], yet it may be approximated by a more general concept [17]. Generally approximations up to a certain depth k of nested levels are considered, denoted MSC^k. We will generically indicate a maximal depth approximation with MSC^*. For further details see also [12].

2.3 A Normal Form for $\mathcal{ALCQ}$

Many semantically equivalent (yet syntactically different) descriptions can be given for the same concept. Equivalent concepts can be reduced to a normal form by means of rewriting rules that preserve their equivalence [1]. We will adopt a normal form extending one given for $\mathcal{ALC}$ descriptions [3], for concepts that are already in negation normal normal form.

Preliminarily, some notation is necessary for naming the various nested subparts (levels) of a concept description D:

- $\mathsf{prim}(D)$ is the set of all the primitive concepts (or their negations) at the top-level of D;
- $\mathsf{val}_R(D)$ is the[1] concept in $\mathcal{ALCQ}$ normal form in the scope of the value restriction (if any) at the top-level of D (otherwise $\mathsf{val}_R(D) = \top$);
- $\mathsf{ex}_R(D)$ is the set of the descriptions C' appearing in existential restrictions $\exists R.C'$ at the top-level conjunction of D;
- $\mathsf{min}_{R.C}(D) = \max\{n \in \mathbb{N} \mid D \sqsubseteq \geq nR.C\}$ (always a finite number);
- $\mathsf{max}_{R.C}(D) = \min\{n \in \mathbb{N} \mid D \sqsubseteq \leq nR.C\}$ (if unlimited, $\mathsf{max}_{R.C}(D) = \infty$).

A normal form may be recursively defined as follows:

Definition 2.5 ($\mathcal{ALCQ}$ normal form). *A concept description C is in $\mathcal{ALCQ}$* normal form *iff $C = \bot$ or $C = \top$ or if $C = C_1 \sqcup \cdots \sqcup C_n$ with*

$$C_i = \prod_{P \in \mathsf{prim}(C_i)} P \sqcap \prod_{R \in N_R} \left\{ \forall R.\mathsf{val}_R(C_i) \sqcap \prod_{E \in \mathsf{ex}_R(C_i)} \exists R.E \ \sqcap \prod_{C \in N_C} \left[\geq m^i_{R.C} R.C \sqcap \leq M^i_{R.C} R.C \right] \right\}$$

where $m^i_{R.C} = \mathsf{min}_{R.C}(C_i)$, $M^i_{R.C} = \mathsf{max}_{R.C}(C_i)$ and, for all $R \in N_R$, $\mathsf{val}_R.C(C_i)$ and every sub-description in $\mathsf{ex}_R.C(C_i)$ are, in their turn, in $\mathcal{ALCQ}$ normal form.

Example 2.3 ($\mathcal{ALCQ}$ normal form). The concept description
$C \equiv (\neg A_1 \sqcap A_2) \sqcup (\exists R_1.B_1 \sqcap \forall R_2.(\exists R_3.(\neg A_3 \sqcap B_2)))$
is in normal form, whereas the following is not:
$D \equiv A_1 \sqcup B_2 \sqcap \neg(A_3 \sqcap \exists R_3.B_2) \sqcup \forall R_2.B_3 \sqcap \forall R_2.(A_1 \sqcap B_3)$
where A_i's and B_j's are primitive concept names and the R_k's are role names.

This normal form can be obtained by means of repeated applications of equivalence preserving operations, namely replacing defined concepts with their definition as in the TBox and pushing the negation into the nested levels (*negation normal form*). This normal form induces an AND-OR tree structure for the concept descriptions in this language. This structure can be used for comparing different concepts and asses their similarity.

[1] A single one because multiple value restrictions can be gathered using the equivalence $\forall R.C \sqcap \cdots \sqcap \forall R.C_n \equiv \forall R.(C_1 \sqcap \cdots \sqcap C_n)$. Then the nested descriptions can be transposed in normal form using further rewriting rules (distributiveness, etc.. . .) [1].

3 Structural Kernels for $\mathcal{ALCQ}$

A simple way to define a kernel function for concept descriptions in normal form would require to adapt a tree kernel [18] where similarity between trees depends on the number of similar subtrees (or paths unraveled from such trees) which does not fully capture the semantic nature of expressive DLs languages.

The kernel function definition should not be based only on structural elements but also (partly) on their semantics, since different descriptions may have similar extensions and vice-versa, especially with expressive DL languages such as $\mathcal{ALCQ}$.

Normal form descriptions can be decomposed level-wise into sub-descriptions. For each level, there are three possibilities: the upper level is dominated by the disjunction (1) of concepts that, in turn, are made up of a conjunction (2) of complex or primitive (3) concepts. In the following the definition of the $\mathcal{ALCQ}$ kernel is reported.

Definition 3.1 (family of $\mathcal{ALCQ}$ kernels). *Given an interpretation $\mathcal{I}$ of $\mathcal{K}$, the $\mathcal{ALCQ}$* kernel *based on $\mathcal{I}$ is the function*[2] $k_{\mathcal{I}} : \mathcal{X} \times \mathcal{X} \mapsto \mathbf{R}$ *structurally defined as follows: given two disjunctive descriptions $D_1 = \bigsqcup_{i=1}^{n} C_i^1$ and $D_2 = \bigsqcup_{j=1}^{m} C_j^2$ in $\mathcal{ALCQ}$ normal form:*
disjunctive descriptions:

$$k_{\mathcal{I}}^{d}(D_1, D_2) = \lambda \sum_{i=1}^{n} \sum_{j=1}^{m} k_{\mathcal{I}}^{c}(C_i^1, C_j^2)$$

with $\lambda \in]0, 1]$
conjunctive descriptions:

$$\begin{aligned}
k_{\mathcal{I}}^{c}(C^1, C^2) = & \prod_{\substack{P_1 \in \mathsf{prim}(C^1) \\ P_2 \in \mathsf{prim}(C^2)}} k_{\mathcal{I}}^{p}(P_1, P_2) \cdot \\
& \cdot \prod_{R \in N_R} k_{\mathcal{I}}^{d}(\mathsf{val}_R(C^1), \mathsf{val}_R(C^2)) \cdot \\
& \cdot \prod_{R \in N_R} \sum_{\substack{C_i^1 \in \mathsf{ex}_R(C^1) \\ C_j^2 \in \mathsf{ex}_R(C^2)}} k_{\mathcal{I}}^{d}(C_i^1, C_j^2) \cdot \\
& \cdot \prod_{R \in N_R} \prod_{C \in N_C} k_{\mathcal{I}}^{n}((\mathsf{min}_{R.C}(C^1), \mathsf{max}_{R.C}(C^1)), (\mathsf{min}_{R.C}(C^2), \mathsf{max}_{R.C}(C^2)))
\end{aligned}$$

numeric restrictions:
if $\min(M_C, M_D) > \max(m_C, m_D)$

$$k_{\mathcal{I}}^{n}((m_C, M_C), (m_D, M_D)) = \frac{\min(M_C, M_D) - \max(m_C, m_D) + 1}{\max(M_C, M_D) - \min(m_C, m_D) + 1}$$

[2] We use the superscripts $k^{\cdot}$ for more clarity.

otherwise $k^n_{\mathcal{I}}((m_C, M_C), (m_D, M_D)) = 0$.

primitive concepts:

$$k^p_{\mathcal{I}}(P_1, P_2) = k_{\text{set}}(P_1^{\mathcal{I}}, P_2^{\mathcal{I}}) = |P_1^{\mathcal{I}} \cap P_2^{\mathcal{I}}|$$

where k_{set} *is the kernel for set structures defined in [14]. This case includes also the negation of primitive concepts:* $(\neg P)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus P^{\mathcal{I}}$

Preliminarily, the extension of the concepts can be approximated by the cardinality of their retrieval. Note also that this is a family of kernels parameterized on an interpretation and on a real number λ.

The first kernel function (k^d) computes the similarity between disjunctive descriptions as the sum of the cross-similarities between all couples of disjuncts from the top-level of either description. The term λ is employed to lessen the contribution coming from the similarity of the sub-descriptions (i.e. amount of indirect similarity between concepts that are related to those at this level) on the grounds of the level where they occur.

The conjunctive kernel (k^c) computes the similarity between two input descriptions, distinguishing primitive concepts, those referred in value restrictions and those referred in existential restrictions. These values are multiplied reflecting the fact that all the restrictions have to be satisfied at a conjunctive level.

The similarity of the qualified numeric restrictions is simply computed (by k^n) as a measure of the overlap between the two intervals. Namely it is the ratio of the amounts of individuals in the overlapping interval and those the larger one, whose extremes are minimum and maximum. Note that some intervals may be unlimited above: $\max = \infty$. In this case we may approximate with an upper limit N greater than $|\Delta^{\mathcal{I}}| + 1$.

The similarity between primitive concepts is measured (by k^p) in terms of the intersection of their extension. Since the extension is in principle unknown, we will epistemically approximate it recurring to the notion of retrieval (see Def. 2.3). Making the *unique names assumption* on the names of the individual occurring in the ABox $\mathcal{A}$, one can consider the *canonical interpretation* [1] $\mathcal{I}$, using $\mathsf{Ind}(\mathcal{A})$ as its domain ($\Delta^{\mathcal{I}} := \mathsf{Ind}(\mathcal{A})$). Note that the ABox may be thought of as a (partially complete) graph structure where multiple instances are located accounting for a number of possible worlds.

Besides, the kernel can be normalized as follows: since the kernel for primitive concepts is essentially a set kernel it may be multiplied by a constant $\lambda_p = 1/\Delta^{\mathcal{I}}$ so that the cardinality of the intersection is weighted by the number of individuals occurring in the overall ABox. Alternatively, another choice could be parameterized on the primitive concepts of the kernel definition $\lambda_P = 1/|P_1^{\mathcal{I}} \cup P_2^{\mathcal{I}}|$ which would weight the rate of similarity (the extension intersection) measured by the kernel with the size of the concepts measured in terms of the individuals belonging to their extensions.

Discussion. Being partially based on the concept structure and only ultimately on the extensions of the concepts at the leaves of the tree, it may be objected that the proposed kernel functions may only roughly capture the semantic similarity of the concepts. This may be well revealed by the case of input concepts that are semantically almost equivalent yet structurally different. However, it must be also pointed out that the rewriting process for putting the concepts in normal form tends to eliminate these differences. More importantly, the ultimate goal for defining a kernel is comparing individuals rather than concepts. This will be performed recurring to the most specific concepts of the individuals w.r.t. the same ABox (see Sect. 4). Hence, it was observed that semantically similar individuals tend to share the same structures as elicited from the same source.

The validity of a kernel depends on the fact that the function is *definite positive*. Yet validity can be also proven exploiting some closure properties of the class of kernel functions w.r.t. several operations [15]. Namely, the multiplication of a kernel by a constant, the addition or multiplication of kernels yields valid kernels. Thus one can demonstrate that the functions introduced above are indeed valid kernels for the given space of hypotheses. Then, exploiting these closure properties it can be proven that:

Proposition 3.1. *Given an interpretation $\mathcal{I}$, the function $k_{\mathcal{I}}$ is a valid kernel for the space $\mathcal{X}$ of $\mathcal{ALCQ}$ descriptions in normal form.*

Observe that the core function is the one on primitive concept extensions. It is essentially a set kernel [14]. The kernel functions for top-level conjunctive and disjunctive descriptions are also positive definite being essentially based on the primitive kernel. Descending through the levels there is an interleaving of the employment of these function up the the basic case of the function for primitive descriptions.

As regards the computational complexity, it is possible to show that the kernel function can be computed in time $O(|N_1||N_2|)$ where $|N_i|$, $i = 1, 2$, is the number of nodes of the concept AND-OR trees. It can computed by means of dynamic programming. It is also worthwhile to note that Knowledge Base Management Systems, especially those dedicated to storing instances [16], generally maintain information regarding concepts and instances which may further speed-up the computation.

4 Extensions

It has been objected that the kernels in this familty would not work for concepts that are equivalent yet syntactically different. However, they are not intended for assessing a concept similarity: ultimately kernel machines employed in inductive tasks need to be applied to instances described in this representation, therefore the most important extension is towards the case of individuals.

Indeed, the kernel function can be extended to the case of individuals $a, b \in \mathsf{Ind}(\mathcal{A})$ by taking into account the approximations of their MSCs (see Sect. 2.2). In this way, we move from a graph representation like the ABox portion containing an individual to an intensional tree-structured representation:

$$k_{\mathcal{I}}(a, b) = k_{\mathcal{I}}(\mathsf{MSC}^*(a), \mathsf{MSC}^*(b))$$

Note that before applying the kernel functions a sort of *completion* of the input descriptions is necessary, substituting the defined concepts with the concept descriptions corresponding to their definitions, so to make explicit the relevant knowledge concerning either individual (example).

The extension of the kernel function to more expressive DL is not trivial. DLs allowing normal form concept definitions can only be considered. Moreover, for each constructor not included in the $\mathcal{ALCQ}$ logic, a kernel definition has to be provided.

Another extension for the kernel function could be made taking into account the similarity between different relationships in a more selective way. This would amount to considering each couple of existential and value restrictions with one element from each description (or equivalently from each related AND-OR tree) and the computing the convolution of the sub-descriptions in the restriction. As previous suggested for λ, this should be weighted by a measure of similarity between the roles measured on the grounds of the available semantics. We propose therefore the following weight: given two roles $R, S \in N_R$: $\lambda_{RS} = |R^{\mathcal{I}} \cap S^{\mathcal{I}}|/|\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}|$.

All of these weighting factors need not to be mere constants. Another possible extension is considering them as functions of the depth of the nodes to be compared: $\lambda : \mathbb{N} \mapsto]0, 1]$ (e.g. $\lambda(n) = 1/n$). In this way one may control the decay of impact of the similarity of related individuals/concepts located ad more deeply nested levels.

As suggested before, the intersection could be measured on the grounds of the relative role extensions with respect to the whole domain of individuals, as follows:

$$\lambda_{RS} = \frac{|R^{\mathcal{I}} \cap S^{\mathcal{I}}|}{|R^{\mathcal{I}} \cup S^{\mathcal{I}}|}$$

It is also worthwhile to recall that some DLs knowledge bases contain also an *R-box* [1] with axioms concerning the roles, one knows beforehand that, for instance, $R \sqsubseteq S$ and compute their similarity consequently.

In order to increase the applicability of precision of the structural kernels and tackle the DL languages supporting the OWL versions, they should be able to work with inverse roles and nominals. The former may be easily accommodated by considering, in the normal form and kernels, a larger set of role names $N_R^* = N_R \cup \{S \mid \exists R \in N_R\colon\ S = R^-\}$. The latter can be dealt with with a set kernel, as in the sub-kernel for primitive concepts. Given the set of individual names O_1 and O_2: $k^o_{\mathcal{I}}(O_1, O_2) = k_{\text{set}}(O_1^{\mathcal{I}}, O_2^{\mathcal{I}}) = |O_1^{\mathcal{I}} \cap O_2^{\mathcal{I}}|$.

Finally, as discussed in [10], related distance measures can also be derived from kernel functions which essentially encode a notion of similarity between concepts and between individuals. This can enable the definition of various distance-based methods for these complex representations spanning from relational clustering [11] to instance-based methods [7].

5 Conclusions and Outlook

Kernel functions have been defined for OWL descriptions which was integrated with a SVM for inducing a statistical classifier working with the complex representations. The resulting classifier could be tested on inductive retrieval and classification problems.

The induced classifier can be exploited for predicting or 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 inclusion. This constitutes a new approach in the SW context, since the efficiency of the statistical and numerical approaches and the effectiveness of a symbolic representation have been combined.

The derivation of distance measures from the kernel function may enable a series of further distance-based data mining techniques such as clustering and instance-based classification. Conversely, new kernel functions can be defined transforming newly proposed distance functions for these representations, which are not language dependent and allow the related data mining methods to better scale w.r.t. the number of individuals in the ABox.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] S. Bloehdorn and Y. Sure. Kernel methods for mining instance data in ontologies. In K. Aberer et al., editors, *In Proceedings of the 6th International Semantic Web Conference, ISWC2007*, volume 4825 of *LNCS*, pages 58–71. Springer, 2007.

[3] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel et al., editors, *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning, KR02*, pages 203–214. Morgan Kaufmann, 2002.

[4] P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation And Applications*. IOS Press, 2005.

[5] W. Cohen and H. Hirsh. Learning the CLASSIC description logic. In P. Torasso et al., editors, *Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning*, pages 121–133. Morgan Kaufmann, 1994.

[6] C. Cumby and D. Roth. On kernel methods for relational learning. In T. Fawcett and N.Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning, ICML2003*, pages 107–114. AAAI Press, 2003.

[7] C. d'Amato, N. Fanizzi, and F. Esposito. Query answering and ontology population: An inductive approach. In S. Bechhofer et al., editors, *Proceedings of the 5th European Semantic Web Conference, ESWC2008*, volume 5021 of *LNCS*, pages 288–302. Springer, 2008.

[8] C. d'Amato, S. Staab, and N. Fanizzi. On the influence of description logics ontologies on conceptual similarity. In A. Gangemi and J. Euzenat, editors, *Proceedings of the 16th EKAW Conference, EKAW2008*, volume 5268 of *LNAI*, pages 48–63. Springer, 2008.

[9] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. *Journal of Logic and Computation*, 4(4):423–452, 1994.

[10] N. Fanizzi and C. d'Amato. A declarative kernel for $\mathcal{ALC}$ concept descriptions. In F. Esposito et al., editors, *In Proceedings of the 16th International Symposium on Methodologies for Intelligent Systems, ISMIS2006*, volume 4203 of *Lecture Notes in Computer Science*, pages 322–331. Springer, 2006.

[11] N. Fanizzi, C. d'Amato, and F. Esposito. Randomized metric induction and evolutionary conceptual clustering for semantic knowledge bases. In M. Silva et al., editors, *Proceedings of the ACM International Conference on Knowledge Management, CIKM2007*, pages 51–60, Lisbon, Portugal, 2007. ACM.

[12] N. Fanizzi, C. d'Amato, and F. Esposito. Learning with kernels in Description Logics. In F. Železný and N. Lavrač, editors, *Proceedings of the 18th International Conference on Inductive Logic Programming, ILP2008*, volume 5194 of *LNAI*, pages 210–225. Springer, 2008.

[13] N. Fanizzi, C. d'Amato, and F. Esposito. Statistical learning for inductive query answering on OWL ontologies. In A. Sheth et al., editors, *Proceedings of the 7th International Semantic Web Conference, ISWC2008*, volume 5318 of *LNCS*, pages 195–212. Springer, 2008.

[14] T. Gärtner, J. Lloyd, and P. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, 2004.

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

[16] I. R. Horrocks, L. Li, D. Turi, and S. K. Bechhofer. The instance store: DL reasoning with large numbers of individuals. In V. Haarslev and R. Möller, editors, *Proceedings of the 2004 Description Logic Workshop, DL 2004*, volume 104 of *CEUR Workshop Proceedings*, pages 31–40. CEUR, 2004.

[17] T. Mantay. Commonality-based ABox retrieval. Technical Report FBI-HH-M-291/2000, Department of Computer Science, University of Hamburg, Germany, 2000.

[18] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.