Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

KOSIMap: Use of Description Logic Reasoning to Align Heterogeneous Ontologies

Quentin Reul[1] and Jeff Z. Pan[2]

[1] VUB STARLab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
[2] University of Aberdeen, Aberdeen AB24 3FX, UK
Quentin.Reul@vub.ac.be; jeff.z.pan@abdn.ac.uk

Abstract. Semantic interoperability is essential on the Semantic Web to enable different information systems to exchange data. Such interoperability can be achieved by identifying similar information in heterogeneous ontologies. In this paper, we describe the Knowledge Organisation System Implicit Mapping (KOSIMap) framework, which differs from existing ontology mapping approaches by using description logic reasoning (i) to extract implicit information as background knowledge for every entity, and (ii) to remove inappropriate mappings from an alignment. The results of our evaluation show that the use of Description Logic in the ontology matching task increases coverage.

1 Introduction

Semantic interoperability enables distributed information systems to exchange data, knowledge, or resources based on common terminologies. These terminologies are often expressed in the form of ontologies as they provide a explicit and server-stored conceptualization of a domain based on well-defined semantics. However, the development of OWL ontologies relies on knowledge engineers to interpret data from domain experts. As a result, two knowledge engineers may interpret the same data differently. This leads to heterogeneity, such as differences in naming and conceptualization, that hinders interoperability among distributed information systems.

Semantic interoperability is essential on the Semantic Web to both provide and create services, and perform complex tasks without prior knowledge of available resources or how to acquire them. Ontology mapping has been recognised as a viable solution for this problem. Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, the task of mapping one ontology to another is that of finding an entity (i.e. classes, properties, and instances) in $\mathcal{O}_1$ that matches an entity in $\mathcal{O}_2$ based on their intended meaning. Although mappings can be derived manually, this process is time consuming and error prone especially as the size and complexity of ontologies increase. Therefore, it is necessary to develop methods to (semi-)automatically discover similar entities in heterogeneous ontologies. Several surveys reviewing ontology matching techniques, and tools have been carried over the years [4, 3]. These surveys show that most successful approaches combine different lexical and structural similarity measures to cover lexical descriptions as well as the descriptive information provided by semantic relations. For instance, Ctx-Match [2] (and its successor SMatch [7]) creates logical formulae by mapping classes in two ontologies to synsets in WordNet [6]. These logical formulae are then processed by a SAT solver to extract semantic mappings between classes in two ontologies. Alternatively, the OLA (OWL-Lite Alignment) framework [5] measures the similarity between two entities in OWL-Lite ontologies based on their features (e.g. labels, super-classes, properties). However, these approaches disregard the role of description logic reasoning to extract implicit information (i.e. logical consequences) about entities as a source of background knowledge.

In this paper, we describe the Knowledge Organisation System Implicit Mapping (KOSIMap) framework, which differs from existing approaches by using description logic reasoning (i) to extract implicit information as background knowledge for every entity, and (ii) to remove inappropriate mappings from an alignment. Note that this paper differs from [13] by describing the KOSIMap framework in detail rather than reporting the results of the Ontology Alignment Evaluation Initiative (OAEI) 2009 campaign. The rest of the paper is organised as follows. In section 2, we review some related approaches on ontology mapping. Section 3 presents how KOSIMap extracts mappings between entities from two ontologies. The results of the evaluation testing the core assumptions of KOSIMap are reported in section 4, while the final section discusses our results and outlines future work.

2 Related Work

OLA [5] relies on OWL-Lite constructs (e.g. `rdfs:label`, `rdfs:subClassOf`) to map entities from two ontologies. More specifically, it first calculates a local similarity score by combining the measure of different constructs through a weighted sum. For instance, the similarity between two classes aggregates the score for their names, superclasses, and properties. OLA then propagates the local similarity score to its neighbours. This iterative process ends when the approximated similarity score does not increase between two iterations. Similarly, Janowicz [9] computes the similarity between two classes based on the overlap of their respective $\mathcal{ALCNR}$ descriptions in the domain of GIScience. In SIM-DL, a similarity value of 1 indicates that compared concept descriptions are equal whereas 0 implies total dissimilarity. For example, the similarity between two classes is computed by the Jaccard similarity coefficient applied to their sets of subclasses, while the similarity between two restrictions $\exists r.\boldsymbol{C}$ and $\exists s.\boldsymbol{D}$ is based on the similarity between the involved roles (i.e. r and s) and fillers ($\boldsymbol{C}$ and $\boldsymbol{D}$). Note that none of these approaches use description logic reasoning to extract implicit information about entities.

Some approaches have also added a debugging component to improve the quality of the mappings. For example, ASMOV [10] first computes a pre-alignment from the matrix that results from the similarity calculation by adding mappings that are maximal within a threshold ζ. This pre-alignment is then subjected to semantic validation, which detects inappropriate mappings and their causes based on asserted axioms in the two ontologies. Alternatively, Meilicke et al. [12] provide non-standard reasoning based on Distributed Description Logic (DDL) to support the revision of mappings. In this case, mappings are encoded as bridge rules (e.g. $\mathcal{O}_1$:$\boldsymbol{A} \overset{\equiv}{\longrightarrow} \mathcal{O}_2$:$\boldsymbol{B}$) and DDL reasoning is applied on these bridge rules to determine inconsistent mappings.

3 KOSIMap

In this section, we present the KOSIMap framework that aligns entities in ontology $\mathcal{O}_1$ and ontology $\mathcal{O}_2$ based on the application of description logic reasoning. KOSIMap first extracts logical consequences embedded in both ontologies using a DL reasoner (§3.1). Next, KOSIMap computes three different types of similarities for every pair of entities (§3.2). We then build a matrix storing the combined values from which a pre-aligment is extracted (§3.3). Finally, we remove inappropriate mappings from the pre-alignment. Note that KOSIMap performs each step consecutively. The source ontology, denoted $\mathcal{O}_s$, is described in Example 1 and is based on the Pizza tutorial [8].

Example 1. Suppose we have an ontology $\mathcal{O}_s$ defined by the following OWL statements:

```
Namespace(pizzaA: <http://www.owl-ontologies.com/pizza#>)
Ontology (
  Class(pizzaA:Pizza_Topping partial)
  Class(pizzaA:Cheese_Topping partial pizzaA:Pizza_Topping)
  Class(pizzaA:Mozzarella_Topping partial intersectionOf(pizzaA:Cheese_Topping
      restriction(pizzaA:hasSpicyness someValuesFrom(pizzaA:Mild))))
  Class(pizzaA:Pepperoni_Topping partial intersectionOf(pizzaA:Pizza_Topping
      restriction(pizzaA:hasSpicyness someValuesFrom(pizzaA:Medium))))
  Class(pizzaA:Tomato_Topping partial intersectionOf(pizzaA:Pizza_Topping
      restriction(pizzaA:hasSpicyness someValuesFrom(pizzaA:Mild))))
  Class(pizzaA:Pizza partial restriction(pizzaA:hasBase someValuesFrom(pizzaA:Pizza_Base)))
  Class(pizzaA:Americana_Pizzas partial intersectionOf(pizzaA:Pizza
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Mozzarella_Topping))
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Tomato_Topping))
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Pepperoni_Topping))
      restriction(pizzaA:hasTopping allValuesFrom(
          unionOf(pizzaA:Mozzarella_Topping pizzaA:Tomato_Topping
              pizzaA:Pepperoni_Topping))))
  Class(pizzaA:Cheesy_Pizza complete intersectionOf(pizzaA:Pizza
      restriction(pizzaA:hasTopping someValuesFrom(pizzaA:Cheese_Topping))))
  ObjectProperty(pizzaA:hasBase)
  SubPropertyOf(pizzaA:hasBase pizzaA:hasIngredient)
  ObjectProperty(pizzaA:hasTopping domain(pizzaA:Pizza) range(pizzaA:Pizza_Topping))
  SubPropertyOf(pizzaA:hasTopping pizzaA:hasIngredient)
  ObjectProperty(pizzaA:topped inverseOf(pizzaA:hasTopping))
)
```

3.1 Pre-Processing

We first process lexical descriptions of each entity (e.g. labels and names) using Natural Language Processing (NLP) techniques. We apply three types of NLP techniques. Firstly, we remove characters, such as '.', '_', '-' and ' ', from the string. Secondly, We used the Pling-Stemmer from [16] to stem plural words.

Finally, we ensure that every string is in lower case. For example, the name of ***Americana_Pizzas*** is converted to "*americanapizza*".

Entities are not only defined by lexical descriptions, but also by the semantics provided by the axioms in the ontology. For example, the subsumption relation links two classes according to the genus/species classification. In KOSIMap, we extract implicit information about every class and object property from the asserted axioms in the ontologies using a DL reasoner (e.g. FaCT++ [18]). The set of all named classes occurring in an ontology $\mathcal{O}$ is denoted by $\texttt{CN}_{\mathcal{O}}$, while $\texttt{RN}_{\mathcal{O}}$ refers to the set of all named object properties in $\mathcal{O}$.

Classes. The set of subsumers of a class $\boldsymbol{A} \in \texttt{CN}_{\mathcal{O}}$, denoted by $\texttt{S}_c(\boldsymbol{A})$, contains every (implicit and explicit) super-classes and equivalent classes of $\boldsymbol{A}$ following the classification of the ontology $\mathcal{O}$ by a DL reasoner. In Example 1, $\texttt{S}_c$(***Americana_Pizzas***) = {***Americana_Pizzas***, ***Cheesy_Pizza***, ***Pizza***}, whereas the set of explicit parents is {***Americana_Pizzas***, ***Pizza***}.

Table 1. Rules to extract properties associated with classes.

PR1	If $\geq 1\ r \sqsubseteq \boldsymbol{C} \in \mathcal{O}$, $r \in \texttt{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \texttt{CN}_{\mathcal{O}}$, & $r \notin \texttt{P}_c(\boldsymbol{C})$ then $\texttt{P}_c(\boldsymbol{C}) := \texttt{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR2	if $\boldsymbol{C} \sqsubseteq \exists r.\boldsymbol{X} \in \mathcal{O}$, $r \in \texttt{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \texttt{CN}_{\mathcal{O}}$, & $r \notin \texttt{P}_c(\boldsymbol{C})$ then $\texttt{P}_c(\boldsymbol{C}) := \texttt{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR3	if $\boldsymbol{C} \sqsubseteq \forall r.\boldsymbol{X} \in \mathcal{O}$, $r \in \texttt{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \texttt{CN}_{\mathcal{O}}$, & $r \notin \texttt{P}_c(\boldsymbol{C})$ then $\texttt{P}_c(\boldsymbol{C}) := \texttt{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR4	if $\boldsymbol{C} \sqsubseteq = 1r.\boldsymbol{X} \in \mathcal{O}$, $r \in \texttt{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \texttt{CN}_{\mathcal{O}}$, & $r \notin \texttt{P}_c(\boldsymbol{C})$ then $\texttt{P}_c(\boldsymbol{C}) := \texttt{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR5	if $\boldsymbol{C} \sqsubseteq \geq 1r.\boldsymbol{X} \in \mathcal{O}$, $r \in \texttt{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \texttt{CN}_{\mathcal{O}}$, & $r \notin \texttt{P}_c(\boldsymbol{C})$ then $\texttt{P}_c(\boldsymbol{C}) := \texttt{P}_c(\boldsymbol{C}) \bigcup \{r\}$
PR6	if $\boldsymbol{C} \sqsubseteq \leq 1r.\boldsymbol{X} \in \mathcal{O}$, $r \in \texttt{RN}_{\mathcal{O}}$, $\boldsymbol{C} \in \texttt{CN}_{\mathcal{O}}$, & $r \notin \texttt{P}_c(\boldsymbol{C})$ then $\texttt{P}_c(\boldsymbol{C}) := \texttt{P}_c(\boldsymbol{C}) \bigcup \{r\}$

Classes are also described in terms of properties, which provides information about the characteristics of a class. Most existing approaches only consider properties that are explicitly associated with classes. In more expressive ontology languages, such as OWL DL, class characteristics can be embedded in the axioms or be inherited from their subsumers. As a result, we have devised several rules to extract the properties associated with a class (Table 1). The first rule (**PR1**) states that the domain of a named property is a property of that class. In Example 1, this rule would infer that *hasTopping* is a property of ***Pizza***. Rules **PR2** to **PR6** process every general concept inclusion in the ontology of the form $\boldsymbol{A} \sqsubseteq \text{Rest}(r).\boldsymbol{B}$, where Rest($r$) is a restriction (e.g. `someValuesFrom`, `allValuesFrom`, and `minCardinality`). Based on these types of general concept inclusion, we are able to extract the implicit object properties associated with $\boldsymbol{A}$. In Example 1, the application of rule **PR2** infers that *hasBase* is a property of ***Pizza***.

Object Properties. The `rdfs:subPropertyOf` construct defines the property hierarchy by stating that a property is a subproperty of another. For example, ontology $\mathcal{O}_s$ states that *hasTopping* is a sub-property of *hasIngredient*. Thus, a reasoner can deduce that if an individual is related to another by the $\mathit{hasTopping}$, then it is also related to the other by the $\mathit{hasIngredient}$ property. The set of super-properties of a property $r \in \mathtt{RN}_{\mathcal{O}}$, denoted $\mathtt{S}_p(r)$, includes all the super-properties of r.

Table 2. Extension rules for binary relation

ER1	If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathtt{R}(r)$, $r \equiv r^- \in \mathcal{O}$, & $(\boldsymbol{Y}, \boldsymbol{X}) \notin \mathtt{R}(r)$ then $\mathtt{R}(r) := \mathtt{R}(r) \bigcup \{(\boldsymbol{Y}, \boldsymbol{X})\}$
ER2	If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathtt{R}(r)$, $r \equiv r_{\circ}^- \in \mathcal{O}$, & $(\boldsymbol{Y}, \boldsymbol{X}) \notin \mathtt{R}(r_{\circ}^-)$ then $\mathtt{R}(r_{\circ}^-) := \mathtt{R}(r_{\circ}^-) \bigcup \{(\boldsymbol{Y}, \boldsymbol{X})\}$
ER3	If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathtt{R}(r)$, $r \sqsubseteq s \in \mathcal{O}$, & $(\boldsymbol{X}, \boldsymbol{Y}) \notin \mathtt{R}(s)$ then $\mathtt{R}(s) := \mathtt{R}(s) \bigcup \{(\boldsymbol{X}, \boldsymbol{Y})\}$
ER4	If $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathtt{R}(r)$, $(\boldsymbol{Y}, \boldsymbol{Z}) \in \mathtt{R}(s)$, $r \circ s \sqsubseteq t \in \mathcal{O}$, & $(\boldsymbol{X}, \boldsymbol{Z}) \notin \mathtt{R}(t)$ then $\mathtt{R}(t) := \mathtt{R}(t) \bigcup \{(\boldsymbol{X}, \boldsymbol{Z})\}$

The set of binary relation of an object property r, denoted $\mathtt{R}(r)$, is a collection of ordered pairs of elements on $\mathtt{CN}_{\mathcal{O}}$. More specifically, the set $\mathtt{R}(r)$ is a subset of the Cartesian product $\mathtt{CN}_{\mathcal{O}} \times \mathtt{CN}_{\mathcal{O}}$. For instance, the statement $(\boldsymbol{A},\boldsymbol{B}) \in \mathtt{R}(r)$ is read as $\boldsymbol{A}$ is r-related to $\boldsymbol{B}$, and $\boldsymbol{A}$ and $\boldsymbol{B}$ is called the domain and the range of r respectively. In Example 1, $\mathtt{R}(\mathit{hasTopping})$ contains the binary relation (***Pizza***, ***Pizza_Topping***). In $\mathcal{EL}+$ [1], we can not rely on OWL semantics to express the domain and range of an object property r (i.e. $\geq 1\ r \sqsubseteq \boldsymbol{C}$ and $\top \sqsubseteq \forall r.\boldsymbol{C}$ respectively) as number and universal restrictions are not allowed. In this case, the binary relation $(\boldsymbol{A}, \boldsymbol{B})$ is contained in $\mathtt{R}(r)$ if and only if the axiom $\boldsymbol{A} \sqsubseteq \exists r.\boldsymbol{B}$ is found in the ontology. In Example 1, $\mathtt{R}(\mathit{hasBase})$ contains (***Pizza***, ***Pizza_Base***). We have extended standard reasoning with four rules to extract implicit binary relations (Table 2). The first rule covers symmetric object properties and adds the inverse binary relation to the set if it has not already been added. The second rule is similar to the first rule but deals with inverse object properties. In Example 1, $\mathtt{R}(\mathit{topped})$ contains (***Pizza_Topping***, ***Pizza***) as *topped* is the inverse property of *hasTopping* and that (***Pizza***, ***Pizza_Topping***) $\in \mathtt{R}(\mathit{hasTopping})$. The last two rules were proposed by [1] and cover role hierarchies and property chain axioms, such as transitive object properties.

3.2 Similarity Generator

The similarity generator computes three kinds of similarities; namely *label similarity*, *property-based similarity*, and *class-based similarity.* We describe each measure in more detail by calculating the similarity between entities in ontology $\mathcal{O}_s$ and entities in $\mathcal{O}_t$ (Example 2). Finally, we describe how individual scores are combined to provide an aggregated value for each pair of entities.

Example 2. Suppose we have an ontology $\mathcal{O}_t$ defined by the following OWL statements:

```
Namespace(p: <http://www.owl-ontologies.com/pizzaB#>)
Ontology (
  Class(pizzaB:Topping partial)
  Class(pizzaB:CheesyTopping partial pizzaB:Topping)
  Class(pizzaB:Mozzarella partial intersectionOf(pizzaB:CheesyTopping
       restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Mild))))
  Class(pizzaB:Tomatoes partial intersectionOf(pizzaB:Topping
       restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Mild))))
  Class(pizzaB:JalapenoPeppers partial intersectionOf(pizzaB:Topping
       restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Hot))))
  Class(pizzaB:Pepperoni partial intersectionOf(pizzaB:Topping
       restriction(pizzaB:hasSpicyness someValuesFrom(pizzaB:Medium))))
  Class(pizzaB:AmericanHot partial intersectionOf(pizzaB:Pizza
       restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:Mozzarella))
       restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:Tomatoes))
       restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:JalapenoPeppers))
       restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:Pepperoni))))
  Class(pizzaB:PizzaWithCheese complete intersectionOf(pizzaB:Pizza
       restriction(pizzaB:hasGarnish someValuesFrom(pizzaB:CheesyTopping))))
  ObjectProperty(pizzaB:hasBase domain(pizzaB:Pizza) range(pizzaB:Base))
  SubPropertyOf(pizzaB:hasBase pizzaB:hasIngredient)
  ObjectProperty(pizzaB:hasGarnish)
  SubPropertyOf(pizzaB:hasGarnish pizzaB:hasIngredient)
)
```

Label Similarity. The most basic feature of entities is their labels, which are defined through the `rdfs:label` construct. Labels are human identifiers (i.e. words) expressed in a vocabulary usually shared by experts in the same domain. Therefore, we assume that equivalent classes are likely to be modelled using similar labels (or names). In KOSIMap, we support several string similarity (e.g. Q-Gram [17] or SMOA [15]) to calculate the label similarity for each pair of entities. For example, the label similarity between ***Americana_Pizzas*** in $\mathcal{O}_s$ and ***AmericanHot*** in $\mathcal{O}_t$ based on the Q-Gram similarity (i.e. Q-Gram("*americanapizza*", "*americanhot*")) is 0.571.

Degree of Commonality Coefficient. The property-based and class-based similarity is calculated based on the *degree of commonality coefficient* (Definition 1). The DoCCoeff between two sets $\mathtt{S}_s$ and $\mathtt{S}_t$ is defined as the sum of the maximum similarity for each element in source set (i.e. $\mathtt{S}_s$). Note that the coefficient returns an **asymmetric** measure to reflect the coverage of the first set by the second. This follows the observations made by [19], who argues that the similarity between sets of complex objects is directional and asymmetric.

Definition 1 (Degree of Commonality Coefficient). *Given two sets* $\mathtt{S}_s$ *and* $\mathtt{S}_t$*, the degree of commonality coefficient between them, denoted DoCCoeff(*$\mathtt{S}_s$*,* $\mathtt{S}_t$*) is defined as:*

$$\mathit{DoCCoeff}(\mathtt{S}_s, \mathtt{S}_t) = \frac{1}{\max(|\mathtt{S}_s|, |\mathtt{S}_t|)} \sum_{e_i \in \mathtt{S}_s} \max_{e_j \in \mathtt{S}_t} \mathit{sim}(e_i, e_j) \qquad (1)$$

where $\mathtt{S}_s$ *is the source set,* $\mathtt{S}_t$ *is the target set, and* $\mathit{sim}(e_i, e_j)$ *computes the similarity between pair of elements in the two sets.*

Property-based similarity computes the similarity between two entities based on the set of properties associated with them. KOSIMap first calculates the overlap between set of super-properties for each pair of properties. Let's compute the property-based similarity between *hasTopping* in $\mathcal{O}_s$ and *hasGarnish* in $\mathcal{O}_t$. In this case, the sets of super-properties for each property is:

– $\mathtt{S}_p$(*hasTopping*) = {*hasIngredient*}
– $\mathtt{S}_p$(*hasGarnish*) = {*hasIngredient*}

We then calculate the DoCCoeff based on the label similarity (i.e. Q-Gram). In this case, the property-based similarity is 1. Secondly, we calculate the overlap between two classes based on their respective sets of inherited properties, which are generated based in the rules in Table 1. Let's now consider the similarity between ***Americana_Pizzas*** in $\mathcal{O}_s$ and ***AmericanHot*** in $\mathcal{O}_t$. In this case, the set of inherited properties for each class is:

– $\mathtt{P}_c$(***Americana_Pizzas***) = {*hasBase*, *hasTopping*}
– $\mathtt{P}_c$(***AmericanHot***) = {*hasBase*, *hasGarnish*}

We then calculate the DoCCoeff between the two sets. In this case, the similarity between two elements (i.e. object properties) is computed based on the property-based similarity between their sets of super-properties. Thus, the property-based similarity between ***Americana_Pizzas*** and ***AmericanHot*** is 1 as `sim`(*hasBase*, *hasBase*) = 1 and `sim`(*hasTopping*, *hasGarnish*) = 1.

Class-based similarity computes the similarity between two entities based on the set of classes associated with them. KOSIMap first computes the overlap between two classes based in their sets of subsumers. Let's compute the class-based similarity between ***Americana_Pizzas*** and ***AmericanHot***. In this case, the set of subsumers for each class is:

– $\mathtt{S}_c$(***Americana_Pizzas***) = {***Americana_Pizzas***, ***Cheesy_Pizza***, ***Pizza***}.
– $\mathtt{S}_c$(***AmericanHot***) = {***AmericanHot***, ***PizzaWithCheese***, ***Pizza*** }.

We then calculate the DoCCoeff between the two sets based on the label similarity (i.e. Q-Gram). In this case, the class-based similarity between ***Americana_Pizzas*** and ***AmericanHot*** is 0.706. Secondly, we calculate the class-based similarity between pairs of object properties based on their set of binary relations. Let's now consider the overlap between *hasTopping* in $\mathcal{O}_s$ and *hasGarnish* in $\mathcal{O}_t$. In this case, the set of binary relations for each object property is:

– R(*hasTopping*) = {(***Pizza***, ***Pizza_Topping***), (***Americana_Pizza***, ***Tomato_Topping***), (***Americana_Pizza***, ***Mozarella_Topping***), (***Americana_Pizza***, ***Pepperoni_Topping***)}.
– R(*hasGarnish*) = {(***AmericanHot***, ***Pepperoni***), (***AmericanHot***, ***JalapenoPeppers***), (***AmericanHot***, ***Mozzarella***), (***AmericanHot***, ***Tomatoes***)}

We then calculate the DoCCoeff between the two sets. As we are dealing with binary relations, the DoCCoeff between two sets of binary relations combines the similarity between the first element of two binary relations with the similarity between their second elements. In this case, the similarity between two elements (i.e. classes) is computed based on the class-based similarity calculated between two classes. For example, the degree of commonality coefficient between (***Pizza***, ***Pizza_Topping***) and (***AmericanHot***, ***Pepperoni***) is 0.333 as `sim`(***Pizza***, ***AmericanHot***) = 0.333 and `sim`(***Pizza_Topping***, ***Pepperoni***) = 0.333. The class-based similarity between *hasTopping* and *hasGranish* is 0.593.

Similarity Aggregation combines the score of the above three types of similarity to obtain a more complete measure of similarity. The combined score for each pair of entities is then stored into a *similarity* matrix (Definition 2), where each entity in the source ontology corresponds to a row and each entity in the target ontology corresponds to a column.

Definition 2 (Similarity Matrix). *A similarity matrix, denoted* $\mathtt{SIM}_{st}$*, is a matrix with dimension n*m, where n and m are the number of entities in the source and target ontology respectively. The entries* $r_{st} \in [0,1]$ *denotes the similarity between* e_s *and* e_t*, where* e_s *is an entity in the source ontology and* e_t *is an entity in the target ontology.*

In KOSIMap, the *global similarity* (i.e. sim_g) is computed through a linear function that balances the impact of each measure by giving it a weight w_k and is defined as:

$$sim_g(e_1, e_2) = \sum_{k=0}^{n} w_k * sim_k(e_1, e_2) \tag{2}$$

where n is the number of similarity measures considered and $w_k \in [0,1]$. Suppose we assign a weight of 0.3 to the label similarity, 0.2 to the property-based similarity and 0.5 to the class-based similarity, then the global similarity between ***Americana_Pizzas*** in $\mathcal{O}_s$ and ***AmericanHot*** in $\mathcal{O}_t$ is $0.3 * 0.571 + 0.2 * 1 + 0.5 * 0.706 = 0.724$. The global similarity for each pair of entities is then stored into a *similarity* matrix.

3.3 Mapping Extraction and Refinement

The goal of the final step is to extract a set of mappings from the similarity matrix. This is normally achieved by discarding all candidate mappings below a threshold ζ. However, this method may return multiple mappings for each entity in the source ontology.

As a result, we propose a two-step approach to extract mappings, where an entity in the source ontology is associated with at most one entity in the source ontology. The approach first extracts a pre-alignment (i.e. $\mathtt{A}_{pre}$) from the similarity matrix. A mapping $\langle e_s, e_t, \equiv, r_{st}\rangle$ is added to $\mathtt{A}_{pre}$ if r_{st} is the highest value for e_s and if it is bigger than a threshold ζ. Note that if two elements e_t^1 and e_t^2 have a similarity value such that $sim_g(e_s, e_t^1) = sim_g(e_s, e_t^2)$, then both $\langle e_s, e_t^1, \equiv, sim_g(e_s, e_t^1)\rangle$ and $\langle e_s, e_t^2, \equiv, sim_g(e_s, e_t^2)\rangle$ are added to A_{pre}.

This pre-alignment is then passed through a refinement process, which eliminates inappropriate mappings. In KOSIMap, we identify two types of inappropriate mappings, namely redundant and inconsistent mappings. Redundant mappings are encountered when mappings in a pre-alignment $\mathtt{A}_{pre}$ can be inferred from existing mappings, while inconsistent mappings occur when a class in the source ontology is mapped to several classes in the target ontology that are defined as disjoint. [14] argues that direct siblings (i.e. entities having the same parent) are disjoint unless it introduces conflicts. As KOSIMap assumes that the local ontologies are consistent, we consider direct siblings as disjoint entities. Table 3 shows the final set of mappings between $\mathcal{O}_s$ and $\mathcal{O}_t$ with a threshold $\zeta = 0.2$. This approach differs from ASMOV [10] in that it checks whether the information inferred by the mappings can be proven by both the explicit and implicit knowledge available in the local ontologies.

Table 3. The mapping resulting from the alignment between the two ontologies.

Entity1	**Relation**	**Entity2**	**Strength**
$\mathcal{O}_s$:***Americana_Pizza***	=	$\mathcal{O}_t$:***AmericanHot***	0.724
$\mathcal{O}_s$:***Cheese_Topping***	=	$\mathcal{O}_t$:***CheesyTopping***	0.567
$\mathcal{O}_s$:***Cheesy_Pizza***	=	$\mathcal{O}_t$:***PizzaWithCheese***	0.75
$\mathcal{O}_s$:***Medium***	=	$\mathcal{O}_t$:***Medium***	0.8
$\mathcal{O}_s$:***Mild***	=	$\mathcal{O}_t$:***Mild***	0.8
$\mathcal{O}_s$:***Mozarella_Topping***	=	$\mathcal{O}_t$:***Mozzarella***	0.487
$\mathcal{O}_s$:***Pepperoni_Topping***	=	$\mathcal{O}_t$:***Pepperoni***	0.533
$\mathcal{O}_s$:***Pizza***	=	$\mathcal{O}_t$:***Pizza***	1.0
$\mathcal{O}_s$:***Pizza_Topping***	=	$\mathcal{O}_t$:***Topping***	0.533
$\mathcal{O}_s$:*hasBase*	=	$\mathcal{O}_t$:*hasBase*	0.861
$\mathcal{O}_s$:*hasIngredient*	=	$\mathcal{O}_t$:*hasIngredient*	0.626
$\mathcal{O}_s$:*hasSpicyness*	=	$\mathcal{O}_t$:*hasSpicyness*	0.61
$\mathcal{O}_s$:*hasTopping*	=	$\mathcal{O}_t$:*hasGarnish*	0.534

4 Evaluation

In this section, we assess the impact of Description Logic reasoning on the computation of the similarity between two entities and on the extraction of appropriate mappings.

4.1 Method

This evaluation is carried out on a subset of the OAEI Conference track[3]. Note that we only consider the ontologies for which a reference alignment is provided (i.e. EKAW, SOFSEM, SIGKDD, IASTED, CMT, ConfOf and EDAS). The advantage of the conference track is that it includes ontologies that share the same domain of discourse (i.e. conference organisation) and that are rich in various types of axioms. The evaluation consists of two experiments:

1. **Explicit vs. Implicit Hierarchy:** This experiment compares the role of description logic in determining the class hierarchy. The first alignment is computed by applying the class-based similarity to the set of super-classes for each class in two ontologies. The set of super-classes for a class $\boldsymbol{A}$ is obtained by traversing the asserted hierarchy from the class itself to the root node of the ontology. The second alignment is obtained by applying the class-based similarity to the set of subsumers of each class (§3.1). Note that the reference alignment only includes mapping between classes.

[3] `http://nb.vse.cz/~svabo/oaei2009/`

2. **Disjointness vs. Siblings:** The alignment extraction method relies on disjointness to determine inconsistent mappings. This approach has been extended to consider direct siblings as disjoint entities. This evaluation focuses on the impact of siblings on the alignment extraction process. The first alignment is computed by only considering the explicit disjointness, while the second alignment is obtained by considering direct siblings. Note that the direct sibling are computed based on the classified ontology. In both cases, the weights for label similarity, property-based similarity, and class-based similarity are set to 0.4, 0.1 and 0.5 respectively. Note that because we are focusing on the extraction step the similarity measure does not have an impact on the results.

4.2 Results

The first experiment compares two methods to obtain the class hierarchy. The first method relies on the explicit class hierarchy, while the second method uses DL reasoning to extract set of subsumers. We applied the class-based similarity on the respective sets to compute 21 pairs of alignments. Generally, the highest f-measure is achieved at the same threshold for the implicit hierarchy as for the explicit hierarchy. The implicit hierarchy achieves better f-measure in 15 cases. In 6 of these 15 case, the recall achieved by the two methods is the same, but the implicit hierarchy yields a better precision. This suggests that the use of the implicit hierarchy (as background knowledge) improves the coverage of the ontology mapping task.

Table 4 shows the harmonic mean (H-Mean) f-measure score of each approach across different thresholds for the 21 tests in the conference test case. We can see that the use of the implicit hierarchy consistently yields better results than the use of the explicit hierarchy. Thus, this further suggests that the use of the implicit hierarchy improves the coverage of the ontology mapping task.

Table 4. H-Mean f-measure at different threshold for the Explicit vs. Implicit Hierarchy experiment.

Threshold	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Implicit Hierarchy	.34	.37	.40	.44	.45	.47	.43	.42	.38
Explicit Hierarchy	.29	.32	.35	.36	.40	.44	.42	.40	.37

The second experiment to evaluate the impact of using asserted disjointness (i.e. explicitly stated in axioms) compared to using implicit disjointness. The implicit disjointness is obtained by considering direct siblings as disjoint entities based on the classified ontology. In KOSIMap, the disjointness is only used during the alignment extraction process, and thus does not have an impact when calculating the similarity between two entities. In 15 out of the 21 tests, the use of direct siblings achieves the same results as those achieved by using the explicit disjointness. This can be explained by the fact that the set of direct siblings is identical to the set of disjoint entities for the entities being mapped.

Table 5 shows the harmonic mean f-measure score of each approach across different thresholds for the 21 tests in the conference test case. We can see that the use of direct siblings consistently yields better results than the use of disjointness. We have also performed this experiment by combining both approaches and have found that this approach always achieves the best result.

Table 5. H-Mean f-measure at different threshold for the Disjointness vs. Siblings.

Threshold	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Disjoint	.26	.33	.43	.52	.57	.57	.42	.34	0
Siblings	.27	.34	.45	.54	.59	.58	.43	.34	0

5 Conclusion

In this paper, we presented the KOSIMap framework, which uses Description Logic reasoning (i) to extract implicit information (i.e. logical consequences) about every entity, and (ii) to remove inappropriate mappings from an alignment. The framework first extracts logical consequences embedded in both ontologies using an OWL DL reasoner. Next, KOSIMap computes three different types of similarities for every pair of entities. We then build a matrix storing the aggregated values for every pair of entities from which mappings are extracted. Finally, we remove inappropriate mappings from the set of all possible mappings. Note that each step is performed in an ordered and consecutive manner.

The results of our evaluation showed that the use of the implicit hierarchy consistently yields better overall f-measure on the OAEI conference track. We also observed that the use of the implicit hierarchy improves the coverage. Secondly, we checked whether the use of direct siblings during the alignment extraction process had a negative impact on the coverage. The overall f-measure showed that the use of direct siblings consistently yields better results than the use of disjointness during the alignment extraction process.

Although these results are encouraging, we realise that the approach can be further improved. For example, the pre-alignment process phase could be improved by iteratively considering another entity in the target ontology when a mapping has been removed during the mapping refinement phase. Moreover, the refinement process could be improved by not only considering local logical inconsistencies, but by also considering logical inconsistencies in the distributed ontologies. For example, Meilicke et al. [12] provide non-standard reasoning based on DDL to support the mapping revision process.

6 Acknowledgements

The IPAS project is co-funded by the Technology Strategy Board's Collaborative Research and Development programme (www.innovateuk.org) and Rolls-Royce (project No. TP/2/IC/6/I/10292).

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.
2. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. In *Proc. 2nd International Semantic Web Conference (ISWC'03)*, 2003.
3. S. Castano, A. Ferrara, S. Montanelli, G. N. Hess, and S. Bruno. State of the art on ontology coordination and matching. Technical report, BOEMIE, March 2007.
4. J. Euzenat and P. Shvaiko. *Ontology Matching.* Springer-Verlag, Berlin, 2007.
5. J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *Proc. ISWC-2003 Workshop on Semantic Information Integration*, 2003.
6. C. Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.
7. F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review Journal (KER)*, 18(3):265–280, 2003.
8. M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. A practical guide to building OWL ontologies using the protégé-OWL plugin and CO-ODE tools. Technical report, University of Manchester, 2004.
9. K. Janowicz. Sim-DL: Towards a semantic similarity measurement theory for the description logic $\mathcal{ALCNR}$ in geographic information retrieval. In *On the Move to Meaningful Internet Systems*, 2006.
10. Y. R. Jean-Mary, E. P. Shironoshita, and M. R. Kabuka. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide*, 2009.
11. H. W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
12. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Reasoning support for mapping revision. In *Proc. 23rd Conference on Artificial Intelligence (AAAI-08)*, 2008.
13. Q. Reul and J. Pan. KOSIMap: Ontology Alignments Results for OAEI 2009. In *Proc. 4th International Workshop on Ontology Matching (OM-2009)*, 2009.
14. S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proc. 2nd European Semantic Web Conference (ESWC05)*, pages 226–240, 2005.
15. G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. In *Proc. 4th International Semantic Web Conference (ISWC 2005)*, 2005.
16. F. M. Suchanek, G. Ifrim, and G. Weikum. LEILA: Learning to extract information by linguistic analysis. In *Proc. 2nd Workshop on Ontology Population (OLP2)*, 2006.
17. E. Sutinen and J. Tarhio. On using Q-Gram locations in approximate string matching. In *Proc. 3rd European Symposium on Algorithms (ESA 95)*, 1995.
18. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. IJCAR 2006*, 2006.
19. A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.