Yet Another Semantic Data Model, but to Improve Ontology Matching

Ivan Bedini [1], Benjamin Nguyen [2], Georges Gardarin [2]

[1] Orange Labs,
42 Rue des Coutures,
14000 Caen, France
ivan.bedini@orange-ftgroup.com

[2] PRiSM Laboratory, University of Versailles,
45 Avenue des Etats-Unis,
78035 Versailles, France
{benjamin.nguyen, georges.gardarin}@prism.uvsq.fr

Abstract. Adoption of semantic models, based on ontologies, is nowadays widely recognized to be particularly useful for integrating multiple data sources. It means that more and more net applications are challenged to real time matching problems. As Current algorithms for matching are time consuming and do not provide a model for storing and reuse found matchings, their adoption in such use cases is difficult. In this paper, we propose a new Semantic Data Model which is able to provide an extensive machine interpretable underground structure for storing and quickly recognize alignments between a set of input sources. As a result we are able to improve application knowledge matching performances and to provide a reusable global concept view for a given domain.

1 Introduction

In the area of application integration the adoption of ontologies can improve and simplify the software development and interoperability. Nevertheless current approaches for building ontologies involve in general a difficult human task that produces a static view of a specific domain, which does not fit the adequate flexibility and evolution required by several use cases.

As already pointed out in [1,2] classical ontology mapping approaches focus on alignment precision and recall, which is of course the primary behaviour to target when mapping ontologies, however, they lack efficiency. This can be explained by three main reasons: (i) the algorithm computational complexity order, as already exposed in [2]; (ii) the fact that algorithms compute measures between every couple of items of ontologies to map, even when they do not have anything in common (like looking for similarities between "*umbrella and sewing machine*"[1]); (iii) the lack of

[1] Comte de Lautréamont, *Les Chants de Maldoror, VI, Roman*, 1869

memorisation that implies that a comparison is done every time two items are met (like a *"Sisyphean task"*[2]), regardless of what has already been calculated.

In this paper we present the Semantic Data Model which aims at providing mechanisms to organize extracted information into a reusable semantic network of concepts.

2 Context and Motivations

The discovery of possible *matchings* (mappings) between items of different ontologies is a complex task that requires the application of several algorithms. As shown in [4,5] these algorithms are of different nature and can be classified in three main categories: syntactic, semantic and structural. A good process for similarity discovery should cover these three categories also implement a combination of these categories of algorithms in order to be able to discover as many matchings as possible. As result, a lot of time is spent computing these algorithms during the matching process.

2.1 Simple Example

Current approaches to matching discovery usually adopt algorithms with exponential computational complexity order [2]. The simple example below shows how algorithms often proceed in order to look for similarities. Let be C_1, C_2 and C_3 three sets of concepts that we want to align:

- C_1 = *{person, address, account}*
- C_2 = *{organization, location, manager}*
- C_3 = *{umbrella, washing machine, stove}*

Let $Sym(x,y) \rightarrow [0,1]$ represent a function that either measures a distance between two concepts or calculates in some way the pertinence (similarity) between them.

The process normally implements algorithms of different nature that must be executed for each couple of concepts belonging to different sets. Thus if we consider the two sets C_1 *and* C_2 we must calculate the similarity between the following set of possible matchings $M_{1,2}$ before discovering that there are only two mappings with real meanings: $A_{1,2}$.

$M_{1,2}$=*{(person,organization),(person,location),(person,manager),(address,organization),(address,location),(address,manager),(account,organization),(account,location),(account, manager)}*
$A_{1,2}$=*{(person,manager), (address,location)}*

The problem becomes even more evident when adding $M_{1,3}$ and $M_{2,3}$ matchings because the global alignment *A* is still composed by the same two matchings, while the similarity algorithm has been executed 27 times (=3^3). Thus if we consider *n* to be

[2] In Greek mythology Sisyphus was compelled to roll a huge rock up a steep hill, but before he reached the top of the hill, the rock always escaped him and he had to begin again (Odyssey, xi. 593).

the average number of concepts for each set and *m* the number of sets to match, then the resulting computational complexity order is $O(n^m)$.

2.2 The matching process

Different definitions of matching process have been already proposed in the literature [4,5] and the one provided in [4] well fits our need.

The process of ontology matching can be summarized in three main steps. The first step is the acquisition of the ontologies to be matched. The problem here is to deal with different ontology representations. The second step is given by the analysis of the ontologies and by the execution of the matching procedures. This step is different depending on the set of adopted algorithms. For this reason, this step is often iterated several times in order to refine the results obtained in the previous executions. In the third step, the mappings (author's note: also called alignments) *among ontology elements are determined. Here we can have different tasks depending on the type of matching process that has been performed.…Finally, a set of mappings are determined between the input ontology elements.*

Depending on the use case, at the end of this process a supplementary step can be added in order to produce the merging of equivalent concepts.

2.3 Web Search Engine Integration to the Matching Process

The introduction of a system able to maintain and reuse the matchings discovered can improve the second step of the matching process and notably reduce the computational time of the whole process.

As shown in [6] this approach, the integration of an external knowledge, has been already adopted by several tools and its usefulness has been demonstrated. For example in [7] authors use the Web to discover new matchings between concepts in order to enrich an ontology. But this experience also shows the limits of this approach. In fact classical search engine results are based on keywords and ranking, which certainly produces good quality results, but so far this traditional methodology requires human intervention to sort the query results due to the heterogeneity of retrieved information. On the contrary results produced by a semantic search engine give us additional information that can be directly usable by machines.

For example querying a classical search engine like Google[3] for "Purchase Order", returns 231000000 links; regardless of the number of links which is very high, the first is a link to a Wikipedia[4] page, the second to a PDF file, the third to Excel format and the fourth to a DOC file format, and so on. While this set of documents is certainly very relevant, it seems difficult to apply a reasoning system; a search on Watson[5] [3] would produce a modest number of results, where documents are semantically structured, and directly interpretable automatically.

[3] Google, moteur de recherche – http://www.google.com
[4] Wikipédia, l'encyclopédie libre – http://www.wikipedia.org
[5] Watson, the Semantic Web Gateway – http://watson.kmi.open.ac.uk

The lack of description of the content produced by classical search engines limits the applicability of other refinement research and automatic interpretation.

The aim of our Semantic Data Model (SDM) is to further improve the results of a query for machine interpretation with the adoption of an organised knowledge and a "memory". The SDM is able to offer the most probable similarities and at the same time, discard improbable matchings within a specific domain. In other words it tries to return a meaningful result set rather than a set of semantics triples.

The next sections introduce this model and provide an informal and a formal description.

3 Informal description of the Semantic Data Model (SDM)

The SDM does not target a high level generic modelling of a world view, rather the aim of this model is to capture and maintain several kinds of information about *application data for information exchange*, in order to be able to discover as many similarities as possible between them and discard matchings irrelevant to a given domain. An example of its use is the matching of output data parameters of a Web Service with input data parameters of another. Another example is to provide a view of the concepts sent in a B2B message (like invoices or purchase orders).

The section below details how this data is represented.

3.1 Objects

The basis of the model is the representation of various object structures following their use within application data. There are three kinds of objects called: *classes*, *properties* and *data types*.

The most basic object in the SDM structure is the **data type**. This kind of object can be also considered as the *printable* data that serves as the basis for applications' input and output. It can be a conventional basic type, like basic XML data type (such as *xs:string* or *xs:integer*) or a more complex representation of a printable data type like *measure, amount* or *text*. In a SDM graphical view these objects are represented as regular rectangle, like in **Figure** 1(a).

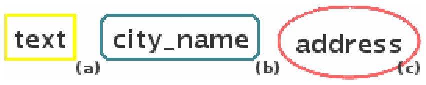

Figure 1 - Three basic SDM object types

The second basic object type is the **property**. This kind of objects represents a specific and atomic characteristic of a class and typically corresponds to objects in the world (of data exchange) that have no underlying structure, except one or more **data type** representations. For example the *first name* and *last name* of a *person*. As shown in **Figure** 1(b), this kind of object structure is represented in the SDM graph as a rounded rectangle.

The last, main object type is called **class** and corresponds intuitively to non atomic objects, thus to object characterised by a finite set of **properties** and optionally one or more **data types**. As shown in **Figure** 1(c), this kind of object structure is represented in the SDM graph as an ellipse.

Figure 2 shows an example of a simple graph representing the three basic SDM objects, where *string* is a data type for *latitude* and *longitude* which are properties for *geographical_coordinate* class.

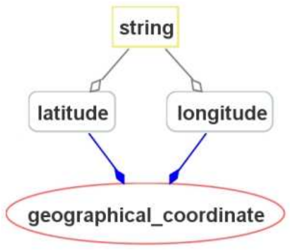

Figure 2 - SDM basic object structures

A special case of objects is the **object property** SDM structure, which is a non atomic object, thus a class, which is also a property of another object class. **Figure** 3 below shows an excerption of an address definition extracted from a B2B standard message. As we can see *address* is a class as well as *country* which is at the same time a property for *address*. In this case we say that *country* is an **object property** for *address*, while *address* is a **father** for *country*. This kind of objects often represents the **role** of a class in a specific instance.

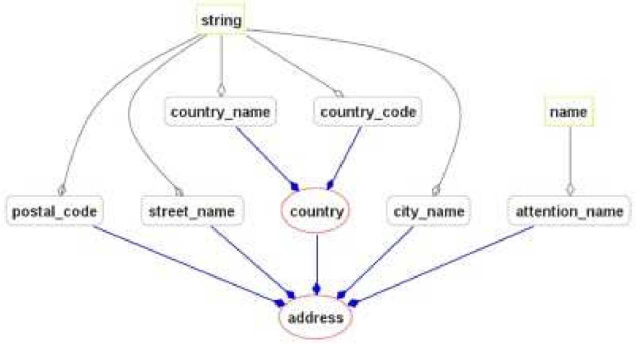

Figure 3 - SDM Object Property

All SDM object structures instances are simply considered **concepts** of the model and each concept can be one of the types described above depending on their behaviour (see Section 3.3).

3.2 Relationships

The second main set of SDM components is the representation of relationships between concepts. Capacity to describe and maintain relationships information within the model is a primary characteristic. In fact, as explained above, the aim of this model is to provide a way to find, with a simple query, all similarities considered for a given concept, or all relationships between two or more concepts. The model is also designed in order to store this information, discovered as automatically as possible, by the application of mining and learning techniques.

Relationships are qualified following the natural associations between formalized concepts: **semantic**, **structure** and **syntax**.

Semantic SDM relationships aim to build a graph of neighbourhood of concepts having a common meaning. In reality only few algorithms are capable of making sense-based similarity choices [8] and few results and tools are available. We currently describe a semantic relationship on the basis of **synonymy** and **shared term** relationships.

Shared term relationship targets compound words like *PostalAddress* and *ShippingAddress* having *Address* as common term.

This kind of association is relevant when we consider XML tag names for instance: it reflects the common practice when building tag names with a sequence of terms. This practice is usually adopted for data definition [9].

This XML tag annotation has the advantage of providing a human readable format but can not be exploited by machine as is. The construction of a lattice of Shared Terms (see Section 4.3 for more details) provides a machine readable format that can find relationships between concepts with similar names like *attention_name* and *attention* of **Figure** 4.

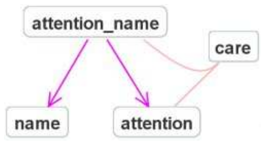

Figure 4 - Semantic Relationships example

S**ynonym** relationship relies on common dictionary based synonymy between terms, like *attention* and *care*, represented by a simple line in the SDM graph of **Figure** 4.

Structural relationship provides hierarchical associations between concepts. These associations define **properties of** a concept, **data-types** of properties and classes or also **equivalence** and "**is a**" relationships.

Figure 2 and **Figure** 3 above show the **data type** structural relationships, which is represented in the SDM graph as a line with empty diamond, like the relationship between *string* and *latitude*.

Property of relationship defines if a concept is a property of another concept or not. It is represented in the SDM graph as line with filled diamond, like the relationship between *geographical_coordinate* and *latitude*.

Is a relationship defines if a concept is considered as specialization or inversely a generalization of another. Intuitively such an association can be used to qualify possible roles of a concept in a specific context, or in a specific usage. For example a *student* can be a *person*, or a *delivery location* can be an *address*. In the SDM graph it is represented as a line with a filled circle at the source end.

The final structural association of the SDM is the representation of the **equivalence** relationships. This kind of association naturally relates concepts having the same meaning in a defined application context. In the SDM it is represented as a line with filled circle at both ends.

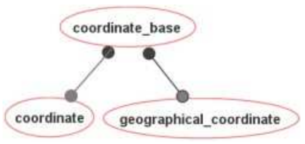

Figure 5 - Equivalence relationship example

Figure 5 shows an example of equivalence relationships between *geographical_coordinate*, *coordinate* and *coordinate_base*.

Syntax groups of relationships aim to maintain associations between retrieved concepts having common abbreviations, stem or a close value using a relevant syntax distance measure (for example up to a specified threshold measured with algorithms like N-Gram or Levenstein distance).

3.3 The Concept

The aim of the SDM is to provide an efficient knowledge representation for automatic data integration. Flexibility and simplicity still remain a primary requirement.

The SDM considers any object instance as a concept independently by its type and source. This view simplifies the representation of objects and provides a more dynamic evolution of the model. In fact when a new input source is added and information is retrieved, concepts already present within the model should find new uses therefore producing new roles for a concept.

For example looking at Figure 6 below we have the SDM graphical representation of two definition of the same core concept *Coordinate* extracted by two XSD files listed below in Listing 1 and Listing 2. Considering the two concepts *latitude* and *longitude*, intuitively in Listing 1 they are **object properties** with a **data type property** *string*, while in Listing 2 they **are a** *position*, thus **classes**, but always **properties of** the *coordinate* main concept. These two views for both concepts do not produce any conflict if we consider the most refined solution as the SDM final definition for *coordinate* that in this case coincides with the definition from Listing 2.

```
<xs:element name="GeographicalCoordinate"
type="GeographicalCoordinateType"/>
<xs:complexType name="GeographicalCoordinateType">
    <xs:sequence>
        <xs:element name="longitude" type="xs:string"/>
        <xs:element name="latitude" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

Listing 1 - Geographical Coordinate definition (excerpt from GS1 standard)

```
<xs:element name="Coordinate" type="CoordinateBaseType"/>
<xs:complexType name="CoordinateBaseType">
    <xs:sequence>
```

```
        <xs:element name="Longitude" type="PositionType"/>
        <xs:element name="Latitude" type="PositionType"/>
        <xs:element name="AltitudeMeasure" type="MeasureType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="PositionType">
    <xs:sequence>
        <xs:element name="DegreeMeasure" type="MeasureType"/>
        <xs:element name="MinuteMeasure" type="MeasureType"/>
    </xs:sequence>
</xs:complexType>
<xs:simpleType name="MeasureType">
    <xs:restriction base="xs:decimal"/>
</xs:simpleType>
```

Listing 2 - Coordinate definition (excerpt from OAGIS standard)

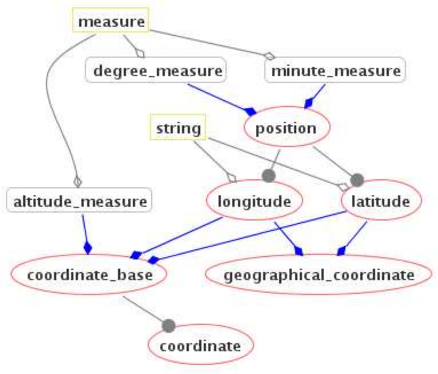

Figure 6 – SDM Structural Relationships example

The above example means that SDM is **flexible** and adding refinements to the model concepts can change their behaviour (e.g.: from SDM property to SDM class). At the same time it is able remain compliant with previous representations. This is also called **dynamicity** of the model. **Figure** 7 provides a graphical view of the basic building block, the **concept**, with its main relationships for representing the structure of stored data in the SDM model as described in previous sections.

The SDM is also **extensible**, for example if we wish to add a multilingual view of the domain it is possible to add new association types to accomplish the new similarity relationship between concepts.

The SDM provides a natural framework for bottom-up approach of domain knowledge design, beginning with information extraction from a source, then refining associations, and then making main concepts arise by adding more sources.

A final task of the design phase should specify functions for object types' transformation. This phase is needed for cases where automation is not applicable, like functions mapping a data-type representation to another (an *address* defined as a simple string to another representation requiring *street*, *city* and *country*).

The SDM provides slots to maintain information about the origin source of a concept. This kind of attribute is called **source** and it is represented as a list of URIs.

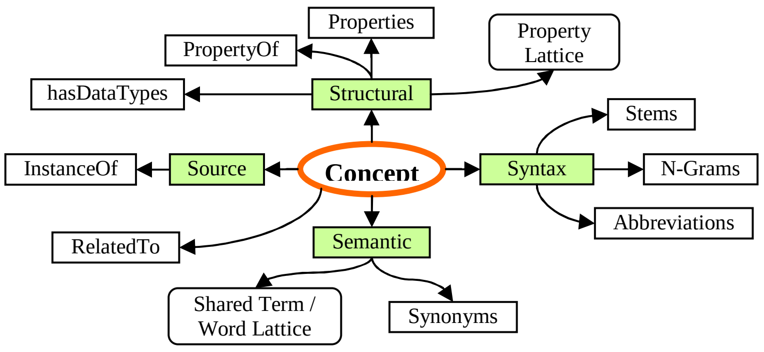

Figure 7 - Concept view

Others attributes provide information about the concept frequency and attendee (i.e. how many sources contain the concept). Such information is useful for statistical and probabilistic measure about the use of a concept in a domain and provides a detailed view to perform choices.

3.4 Graphical Representation

The SDM model graphical representation provides a global view of model's concepts storage with their relationships.

In first instance, by supporting subtype and property relationships the model achieves a *structurally object-oriented* model, i.e. one which is able to represent data types and attributes that are found in Object-Oriented languages like UML. Secondly by supporting semantic and syntax relationships the model realizes a *semantic model*, i.e. one which is able to represent associations meaning based that are found in ontological language like OWL. Thus the model has sufficient expressiveness to maintain automatic information extraction for the task of application data integration.

Table 1 gives the graphical syntax we use to describe SDM.

Table 1 - SDM graphical syntax

Symbol	Name	Description
label	Class Object	Non atomic objects, thus to object characterised by a finite set of properties and/or one or more data type
label	Property Object	Specific and atomic characteristic of a class
label	Data type Object	Printable data that serves as the basis for application's input and output
—◇	Data type relationship	Link to the printable concept
—◆	Property relationship	Links with concepts which are part of a non atomic concept, a property
—●	Is a relationship	Specialization or inversely a generalization of another concept
●—●	Equivalence relationship	Concepts having the same sense in a defined application context

→	Shared term relationship	Compound words having common terms
↔	Syntax relationship	Associations between concepts having common abbreviations, stem or also with syntax distance measure up to a specified threshold (N-Gram, Levenstein distance, …)
—	Synonymy relationship	Dictionary based synonymy between concept labels

Figure 8 shows an example of an excerpt of the address definition extracted from the GS1 B2B standard. This picture shows that the concept *address* emerges as main concept.

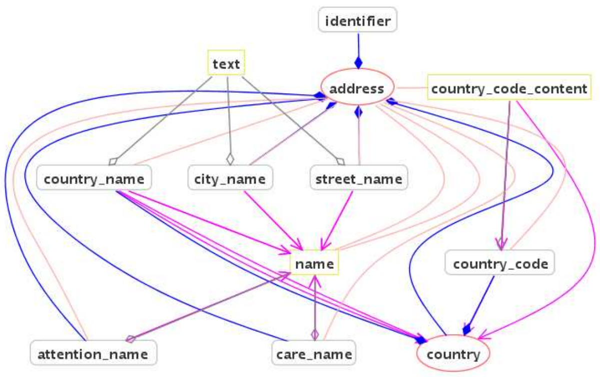

Figure 8 - SDM Graphical Representation

4 A Formal Definition of the Semantic Data Model

In this Section we provide a formal definition for the Semantic Data Model.

4.1 Model

Definition 1. A concept is the basic element of the model and is defined as a quadruple $c = <l, Hc, Rc, Inst>$ where:

- the label l is a common word (simple or compound) that best represents the concept. It is selected from a set of names extracted from the corpus source that can be associated to the concept (e.g.: in **Figure** 6 the classes *geographical_coordinate* and *coordinate* can be associated to the same concept and the common name can be one of them).
- *Hc* is the set of structural relationship, which correspond to the subsumption hierarchy.
- *Rc* is the set of relations and is partitioned into t wo subsets. One is the set of all assertions in which the relation is a semantic relation and the other one is the set of all assertions in which the relation is a non-semantic relation.

Structural relations should also be considered as semantic relationships, similarly to the WordNet [10] approach with meronymy and hyponymy relationships. But in the SDM it is preferable to maintain these two types separately.

- *Inst* is the set of originating instances of a concept.

Definition 2. A model is defined as a tuple $<C, R>$ where C is the set of concepts extracted from a given corpus source and R is the set of binary relationships between concepts of C.

4.2 Objects

Let O be the set of all concepts extracted from a corpus source belonging to a domain of interest.

Definition 3. Let C be a set of concepts called set of concept **classes**, $C = \{c_1, \ldots, c_m\}$ a finite subset of O. A concept is considered to be a **class** if it has more than one property. $c \in C$ *if* $P(c)= \{c_1, \ldots, c_m\}$, *for* $m > 1$.
$P(c) \neq \emptyset$ is called the set of properties for a given concept c.

Definition 4. Let P be a set of concepts, called the set of **properties**, which is a finite subset of O. A concept c_p is a **property** if exists at least one super class of which it is a property. A concept $c_p \in P$ *if* $\exists c_j \in C \mid c_p \in P(c_j)$

Definition 5. Let D be a set of concepts, called the set of **printable** concepts or **data-type**, a finite subset of O. A concept c_{dt} is considered as a **data-type** if it has no properties and it is directly related to a printable type.

As defined above a class is a non atomic object, which implicitly implies that a class must have more than one property. Thus if a class has only one property we assume that the property is just a representation of a class because it does not provide further information. We define the following rule:
Rule 1. Atomic classes equivalence:
$\forall c_i, c_j \in C$ *with* $i \neq j$, *if* $P(c_i) = \{c_j\} \Longrightarrow c_j \approx c_i$

4.3 Shared Term Lattice

The Shared Term relationship is particularly useful when the input ontology uses compound words for concepts' names. In fact normally naming similarities are often based on algorithms adopting string matching or distance measure. It is obvious that in this case these kinds of algorithms can fail. With the construction of a lattice based on shared terms, semantics and syntactic matching can quickly discover and also discard those concepts with/without naming similarities. In this section we define the lattice built over this kind of relationship.

Definition 6. Let *w* a short sequence of terms t_i that for simplicity we formalise in the common set format: $w = \{t_1, ..., t_n\}$, *for1* $\geq n \geq 6$. We define w as a **compound word**.

We limit to 6 the upper bound value of *n*, the number of terms of a compound word, because heuristically more than 6 terms loose sense and the compound word could be considered like a "sentence" itself.

Definition 7. Shared term relationship *St* is a directed association from a compound word w_1 to another compound word w_2 composed by a subset of terms of w_1.

Let *D* be a set of all compound words extracted by a given corpus source for a domain, $D = \{w_1, w_2, \ldots, w_n\}$.

Let be $w_1, w_2 \in D \mid |w_1| \cap |w_2| \neq \varnothing \Rightarrow w1\ \mathbf{St}\ w2$.

Shared term relationships derived properties:

i) $|w_1| \cap \varnothing = \varnothing$ *and* $|w_1| \cup \varnothing = w_1$

ii) $|w_1| \cap |w_1| = w_1$ *and* $|w_1| \cup |w_1| = w_1$

iii) if $w_1 = \{t_1, \ldots, t_m\}$, $w_2 = \{t_1, \ldots, t_m, t_{m+1}, ..., t_n\}$ *with* $m < n$, *in this case we say that* w_1 *is a direct subsequence of* $w_2 \Rightarrow |w_1| \cap |w_2| = w_1$ and $|w_1| \cup |w_2| = w_2$

iiii) if $w_1 = \{t_1, \ldots, t_m\}$, $w_2 = \{t_1, \ldots, t_m, t_{m+1}, ..., t_n\}$, $w_3 = \{t_1, \ldots, t_m, t_{m+1}, ..., t_h\}$ *with* $m < n$ *and* $m < l \Rightarrow |w_2| \cap |w_3| = w_1$ *and* $|w_2| \cup |w_3| = w_1$, *in this case we say that* w_1 *is the root word for* w_1 *and* w_2

iiiii) if $\exists w_1, w_2 \in D, \exists w_3 \notin D \mid |w_1| \cap |w_2| = w_3$, *In his case we say that the root word is an extension of D and that D is a non complete set of compound word for the domain.*

iiiiii) if $\forall\ w_1, w_2 \in D \mid |w_1| \cap |w_2| = w_3 \Rightarrow w_3 \in D$, *in this case we say that D is a complete set of compound word for the domain*

Definition 8. We define the **words lattice** (**WL)** as the main part of a domain "semantic network" built over **St** relationships. The WL is based on a complete set of compound words, which means that if the starting set is not complete, than we have to add extensions to complete the original set.

Let be *Dc* the completed set of words extracted from a domain of interest, $Dc = \{w1, \ldots, wn\}$, than **WL** is defined as a tuple of words and St relationships: $WL = <w, St>$, where $w \in$ Dc and St is the set of binary associations between words, $w_1, w_2 \in D \mid w1\ \mathbf{St}\ w2$.

Definition 9. We define the **root nodes** of the WL those words belonging to Dc, the completed set of words D, such that for each wx belonging to the set, wt intersection wx = wt for each not empty intersection. $\forall w_i \in D,\ R = \{wr \in Dc \mid w_i \cap wr = wr\}$

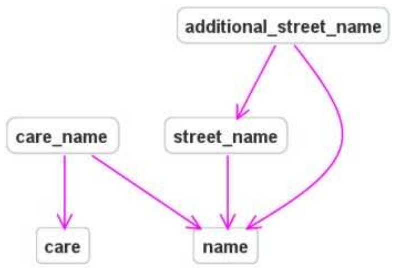

Figure 9 – Words Lattice example

Example 1. Given a set of compound words derived from the following four tags: *CareOfName*, *AdditionalStreetName*, *StreetName* and *CareOf*, we obtain the set of words *D = {care_name, street_name, additional_street_name, care}*. The completed set of *D*, *Dc* is the following *Dc = {care_name, street_name, additional_street_name, street, name}*, where *name* has been added in order to fill the intersection between *care_name* and *street_name*. While *R*, the set of root words is composed by *care* and *name* (See Figure 9).

4.4 Special Concern About Input Source

When retrieving information it is important to know how sources are built in order to be able to apply the best algorithm to get better results. In fact we have heuristically observed that different practices on XML Schema files construction can provide different results. In our use case we are building a semantic network of concepts, thus it is obvious that if input files have a correct semantics and structure, it is possible to simply get good quality results.

In this paper we introduce the concept of **documents semantically well structured** in order to define a classification of documents that permit to decide as automatically as possible the adoption of specific algorithms depending on the input source quality.

Definition 10. A concept *c* is **semantically valid** if its label belongs to a standard common dictionary of words (like the English oxford dictionary for the English language), rather than abbreviations or acronyms.

Definition 11. Let *C* be a set of concepts, we say that C is **well structured** if it is possible to bring back objects relationships to a hierarchy structure by symmetric functions.

Definition 12. Let $S = \{c_1, \ldots, c_n\}$ be a non empty set of concepts input sources for the SDM, we define *S* **semantically well structured** if $\forall c \in S$, *c* is well structured and semantically valid.

5 SDM to Integrate B2B Messages

In order to test our model we have implemented a prototype which automatically retrieves information from a set of XML Schema files as explained in [6]. In this experience we collect a consistent set of XSD files defined by B2B standard bodies and we make the hypothesis that each standard body produces files containing enough information to be considered an ontology itself. It implies that the merging operation of such a corpus is equivalent to the ontology merging.

As defined in Section 2.2 the ontology matching is composed by three main steps. In this experience we integrate SDM in order to improve the analysis and discovery of possible matchings between concepts of input ontologies, corresponding to the second step.

Listing 3 shows the overall algorithm that produces the ontology merging. Given a concept *ci*, the *GetSimilarityFunction* firstly queries the SDM WL (line 2/3) in order to retrieve the list of those concepts having semantics correlations. The parameter *WAssLevel* specifies the distance between nodes of the lattice to be retrieved. In second instance we go through the retrieved list *cxList* to look for structure similarity to decide if two concepts are equivalent and thus to be merged. Finally we modify the algorithm and we do not use the SDM WL to look for similar concepts, but we calculate a string distance measure between a given concept and all other concepts of the input ontologies (i.e. just changing the *cxList* with the list of all concepts to be merged).

```
1.   Function GetSimilarConcepts(){
2.     cxList<concept> =
3.       BuildListOfSimilarConcepts(ci, WLattice, WAssLevel)
4.     for each (cx belonging to cxList) do
5.       affinityValue = lookStructAffinity(ci,cx)
6.       If affinityValue > getStructThreshold() then
7.         MergeConcepts(ci, cx);
8.   }
9.
10.  Function MergeConcepts(ci, cx) {
11.    calculateMostFrequentConcept(ci, cx);
12.    calculate(ci.properties U cx.properties);
13.    merge(ci.relationships, cx.relationships);
14.  }
```

Listing 3 – Merging algorithm with SDM STLattice

Table 2 provides some results of the test applied to three different sets of input sources. The first one is a set of *Address* definitions, the second is a sub-set of the *Invoice* definition and the last is more complete set of *Invoice* definition. *Families* column represents the number of B2B standards for each input source, with the correspondent number of files and the number of concepts for the merged ontology. *Lattice/Term Sharing* column says if the SDM has been adopted for the test and when it is not used, the *Term Sharing* value says if the structure affinity analysis is done for each couple of concepts or only for those concepts having some semantic similarities calculated to each iteration.

Table 2 – SDM WL relationships adoption, experimental results

Corpus	Families	Files	Concepts	Lattice/Term Sharing	ST Lattice ass. level	Execution Time (ms)
Address	8	12	195	Yes	1	188
	8	12	195	Yes	2	328
	8	12	195	No/No	/	937
	8	12	195	No/Yes	/	391
Small Invoice	3	55	1183	Yes	1	603
	3	55	1183	Yes	2	3373
	3	55	1183	No/Yes	/	4217
Invoice	8	187	5808	Yes	1	5283
	8	187	5808	Yes	2	38130
	8	187	5808	No/Yes	/	68586

As we can see from Table 2 the adoption of such a model produces better performances for the matching process. In fact it facilitates the detection of similar concepts and allows to discard those concepts having no shared meaning. In the example of Section 2.1 corresponds to discard several useless correspondences and applying matching algorithms only on those matching with more sense (e.g. discarding the research of similarities between *person* and *umbrella*).

More than this we can observe that greater is the number of concepts better is the gain of execution time.

6 Open Issues and Conclusion

In this paper we have shortly presented the Semantic Data Model which is in agreement with the “Next Generation Semantic Web Applications" vision as described in [11].

The SDM is an organised knowledge representation for a specific domain, like a memory, which aims to provide an important building block to improve the automatic matching of different ontologies.

The formal framework provided by the SDM has been used to test the merging of few ontologies and results arising from its adoption show that it furnishes an important element able to improve performances in the ontology matching process.

However, even if the SDM adoption to the B2B domain shows encouraging results, few problems remain to be solved. The main complexity resides in the fulfillment of model instances with the complete and correct set of relationships between concepts.

The quality of instances of the model is highly dependent from the algorithms used to discover all possible similarities. The lost of some links could lead to low precision and high recall measures. This problem is common to all ontology matching applications and once again the adoption of the SDM can provide the right way for implementing more complex algorithms, thus more time consuming, and to resolve this general issue. In fact it permits to build a background knowledge that can be stored and further reused.

We plan on continuing this work with the further development of the SDM and of a more complete tool for ontology matching, capable to associate semantic concepts on the basis of their meaning and context.

Bibliography

1. Sabou M., d'Aquin M., Motta E. Using the Semantic Web as Background Knowledge for Ontology Mapping. In Proc. of the International Workshop on Ontology Matching, collocated with ISWC'06
2. Ehrig M., Staab S. QOM - Quick Ontology Mapping. In Proceeding of ISWC, 2004, pages 683-697
3. Mathieu d'Aquin, Claudio Baldassarre, Laurian Gridinoc, Sofia Angeletou, Marta Sabou, and Enrico Motta, 2007. Watson: A Gateway for Next Generation Semantic Web Applications. Poster session of the International Semantic Web Conference, ISWC 2007.
4. Castano S., Ferrar A., Montanelli S., Hess G. N., Bruno S. State of the Art on Ontology Coorination and Matching. Deliverable 4.4 Version 1.0 Final, March 2007. BOEMIE Project.
5. Euzenat J., Le Bach T., Barrasa J., Bouquet P., De Bo J., Dieng R., Ehrig M., Hauswirth M., Jarrar M., Lara R., Maynard D., Napoli A., Stamou G., Stuckenschmidt H., Shvaiko P., Tessaris S., Van Acker S. & Zaihrayeu I., State of the Art on Ontology Alignment. Knowledge Web Deliverable #D2.2.3, INRIA, Saint Ismier, 2004.
6. Bedini I., Nguyen B., Gardarin, G. Deriving Ontologies from XML Schema. To Appear in Proc. of EDA. June 2008, France.
7. Agirre E., Ansa O., Hovy E., Martinez E.. Enriching Very Large Ontologies Using the WWW. in Proc. of the Ontology Learning Workshop, ECAI, Berlin, Germany, 2000.
8. Giunchiglia F., Shvaiko P., Yatskevich M. S-Match: an algorithm and an implementation of semantic matching. In Proceedings of ESWS 2004, Heraklion (GR), pages 61–75, 2004.
9. Bedini I., Nguyen B., Gardarin G. B2B Automatic Taxonomy Construction. To Appear in Proc. of ICEIS. June 2008. p12 - 16.
10. Miller, G.A. (1995). WORDNET: A lexical database for English. Communications of ACM (11), 39-4**1**.
11. Motta, E., Sabou, M. Next Generation Semantic Web Applications. In Proc. of the 1st Asian Semantic Web Conference (ASWC), Beijing, China 2006