Semantics-based Profiles Modeling and Matching for Resources Access

Max Chevalier *Chantal Soulé-Dupuy* *Pascaline Laure Tchienehom*
Max.Chevalier@irit.fr *Chantal.Soulé-Dupuy@irit.fr* *Pascaline.Tchienehom@irit.fr*
05 61 55 74 43 *05 61 12 87 90* *05 61 12 88 82*
IRIT, 118 route de Narbonne, 31062 Toulouse cedex 4, France

Abstract

Heterogeneity of resources (information, users, hardware devices, etc.) placed at disposal has raised the problem of defining a generic model, which could be used as a basis for describing resources in various applications. In this article, we propose a profile generic model, which describes the logical structure, the contents and the semantics of non predefined resources. The goal is to clarify the semantics of profiles descriptive elements and to use this semantics for the definition of means for an automatic deduction of elements with compatible semantics between profiles described differently. The defined semantics is not limited to a particular application (it could be shared) and should then allow interoperability between profiles models coming eventually from different applications. In order to evaluate profiles interoperability, we define a flexible matching algorithm and we discuss the results of its implementation.

Keywords: *Semantics, interoperability, profile, generic model, matching, resources access.*

Résumé

L'hétérogénéité des ressources (informations, utilisateurs, dispositifs matériels, etc.) mises à disposition a soulevé le problème de la définition d'un modèle générique, qui pourrait être utilisé comme base de description de ressources dans différentes applications. Dans cet article, nous proposons un modèle générique de profil qui décrit la structure logique, le contenu et la sémantique de ressources non prédéfinies. Le but est d'expliciter la sémantique des éléments descriptifs d'un profil et d'utiliser cette sémantique pour définir des moyens de déduction automatique d'éléments de sémantique compatible entre profils décrits différemment. La sémantique définie n'est pas limitée à une application particulière (elle peut être partagée) et doit donc permettre de faire interopérer des profils issus éventuellement d'applications différentes. Afin d'évaluer l'interopérabilité de profils, nous définissons un algorithme d'appariement flexible et nous discutons des résultats de son implémentation.

Mots-clés : *Sémantique, interopérabilité, profil, modèle générique, appariement, accès à des ressources.*

1. Introduction

The development of internet, especially the World Wide Web, as well as intranets and numeric environments have led to a heterogeneous important amount of resources placed at disposal. A resource can be of various natures like: hardware device (mobile phones, PDA, captors, etc.), software device, information (a document for instance), collection of information, user, and so on. There is a real need of interoperability or cooperation between resources descriptions (called profiles) in applications, in order to realize a specific task: finding information that meet a user's need; sending messages (e-mail, sms, mms) to a user for advertisements or for preventing him from a breaking entrance of his house; and so forth. In this context, one solution could be to homogenously describe the different resources that have to cooperate. However, the diversity and heterogeneity of resources certainly lead to a great disparity in their description.

In order to improve this cooperation, it is essential to defined models which have at the same time the properties of *extensibility, flexibility, re-usability and interoperability*. For that, semantics have to be associated to the description of resources and should enable a coherent cooperation between different models, by using metadata. For instance, users interested in "recent information" could have their own definition of this notion. Thus, their search results should be strongly linked to the interpretation of the semantics they have associated to this notion.

In this paper, we propose a model for the description of non predefined resources (profile model) that has a double dimension: *generic and semantic*. The generic aspect provides us with a homogeneous framework of description and the semantic part enables us to mitigate the disparities or even the discrepancies which remain at the profiles instances level, in order to improve profiles interoperability in resources access applications. Let us note that the semantics is also defined generically and then instantiate in profiles examples. Moreover, we also defined a method for profiles matching, based on their

semantics. This method deduces couples of descriptive profiles elements, which have compatible semantics.

2. Literature review

Resources access, in this paper, represents a broader view of information access [2] where resources are not limited to information (documents) and users but can be extended to any kind of elements (person, thing or action) depending on the application: documents parts, documents collections, journals articles, hardware devices, user's context, users' group, etc.

Information access techniques help an individual to find information that meets his needs. We can gather them in two main groups: pull techniques or information retrieval techniques [18] [4], which need an explicit request of an individual and push techniques or filtering or recommendation techniques [15] [17], which do not need an explicit demand to return information to users.

These techniques are based on resources models that we called profiles. In traditional information access applications, profiles semantics is implicit or strongly linked to the application where they have been defined. In general, profiles models can be divided in three categories:

- *attribute-value model* [21] where attributes are independent and not structured. In this model, we cannot have two attributes with the same name;
- *logical structure based model* with an associated contents [5] [7] where attributes are structured and identified by a path. Two attributes can then have the same name but different associated paths;
- *semantics-based model* [6] [13] where an explicit semantics is associated to the logical structure elements by using a metadata language.

The first two types of models do not give any explicit information on profiles semantics and hence reduce or even exclude any cooperation between profiles, especially described by different applications. In order to solve those problems of interoperability, we do need extensible, flexible, re-usable and interoperable models [3]. These properties could be achieved with generic [14] and semantic [11] profiles models. For Instance, there are models that aim at describing the semantics of user's context through the capabilities of their devices like: CC/PP (Composite Capability/Preference Profiles) [13] and CSCP (Comprehensive Structured Context Profiles) [6]. The basic idea is to use metadata languages, which will bridge two different resources descriptions thanks to the analysis of their semantics defined by metadata.

The specificity of the profile model proposed in this paper is its generic and semantic aspect. Moreover, the model allows the description of non predefined resources. We also define a flexible method for profiles matching in resources access. This method enables an automatic deduction of profiles elements pairs of similar or compatible semantics between two profiles. It will also allow the evaluation of a similarity measure between elements pairs and afterward between profiles. This method aim at reducing incoherence and silence related to a similarity based only on logical structure elements, especially at an inter applications level.

3. Modeling of profiles for resources access

In this section, we present the proposed profile generic model for non predefined resources: the logical structure, the contents and the related semantics. Thereafter, we describe profiles instances and explain the interest of profiles interoperability for profiles matching in resources access.

3.1. Profile generic model

The *figure 1* presents the profile generic model (in UML [16]) proposed. It results from the analysis of various systems, in order to deduce a general model from them. The existing systems are conceived to achieve particular goals according to specificities of their context: recommendation of Web pages according to bookmarks [20], mails filtering [12], electronic trade [10], exploitation of user's context [6] [13], etc. Contrary to these systems, our model is enough general to be used by various applications.

The profile generic model of *figure 1* is subdivided into four levels: *the profile logical structure, the profile contents, the profile logical structure semantics and the profile contents semantics*.

3.1.1. Logical structure and contents

The *logical structure* presents the general structure of a profile. This structure is in the form of a hierarchy of re-usable elements (*ReusableElement* class) that characterize it. This hierarchy is a tree

where nodes or profile elements can be either profiles (instances of class *Profile*) or attributes (instances of class *Attribute*) that describes characteristics of a given profile in the hierarchy (which is the nearest profile in the hierarchy). There are two types of attributes: the class *NonLeafAttribute* that represents categories of profiles elements (for example the attribute *user's preferences* can be composed of others attributes like: *language, size and date*) and the class *LeafAttribute* that describes leaf attributes of the logical structure to which one can affect values. Hence, a re-usable element can be an instance of either class *Profile*, *NonLeafAttribute* or *LeafAttribute*.

The *profile contents* (see class *ContentsElt*), associated to leaf attributes, is composed of lists of *value-weight* pairs. These lists can contain only one pair of *value-weight* (for example the attribute *document size*) or several pairs of *value-weight* (for instance the attribute *document key word*). The *value* is the real contents of the attribute and the *weight* is a numeric value that describes at which point the *value* represents the attribute. For instance, if a user prefers english documents to french documents, we should define weights that represent that preference. Note that contents elements representation here follows the vector space model.

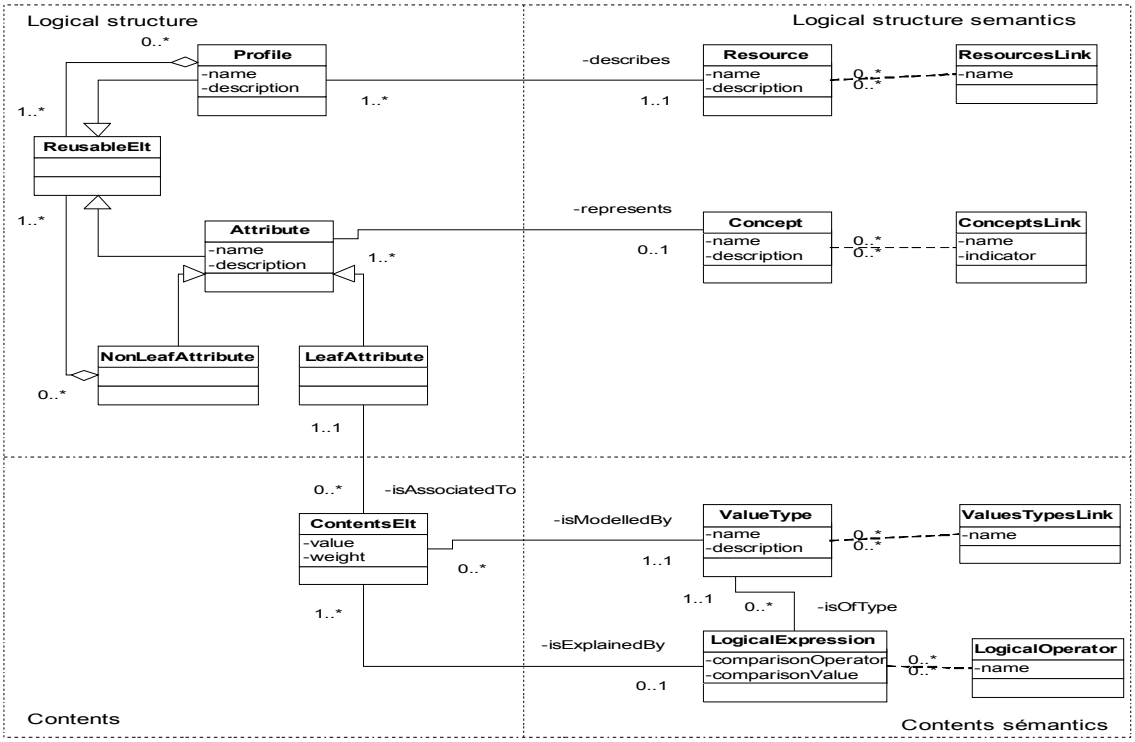

Figure 1. Profile generic model

3.1.2. Logical structure semantics and contents semantics

The generic model will also enable us to clarify the semantics of a profile logical structure and contents. The *logical structure semantics* of the generic model clarifies what a profile and an attribute represent. *Profile semantics* is the description of a category of resource (information, user, hardware devices, etc.) in a given context represented by the class *Resource*. The interest of this class *Resource* is to enable the re-usability of existing profiles models of a given category for the description of new ones. *Attribute semantics* clarifies the characteristic that an attribute describes represented by the class *Concept*. Thus, attributes will be related to generic concepts that come generally from existing metadata languages like: Dublin Core, Wordnet, etc. For instance, the attribute *information_language* of a document can be related to the metadata *dc:language* of the Dublin Core.

On the other hand, *contents semantics* of a profile clarifies the representation model or datatype (instance of class *TypeElement*) of a leaf attribute contents: integer, string, date, dates patterns (ddmmyyyy, mmyyyy, etc.), addresses patterns, and so on (*see also* XMLSchema element type). Moreover, contents semantics can refine the meaning of a contents element thanks to logical expressions. For example, we can express the fact that a user is interested in information published before a given date and after another one. We can then combine logical expressions with the logical operators: AND, OR.

Semantics is represented in the generic model by the classes: *Resource, Concept, ValueType and LogicalExpression* as well as by classes of associations (*ResourcesLink, ConceptsLink, ValuesTypesLink, LogicalOperator*) whose instances clarify the semantic links (*subsumption, equivalence*, for instance) that exist between instances of classes previously cited. This semantics is based on metadata languages (Dublin Core, RDF, RDFS, OWL, XMLSchema, etc.) that could be shared among profiles.

The interest of using a generic model for defining a given profile is that it proposes a basic framework for the description of several classes of profiles. Instances of parts of the proposed generic model are illustrated in [8]. In the following section, we describe characteristics of complete profiles instances and we also present the specificities of profiles interoperability based on the semantic part of the profiles.

3.2. Profiles instances and profiles interoperability

UML is a semi-formal language that has helped us to get a better visual aspect of our generic model. On the other hand, we have chosen RDF-like technologies (RDF/RDFS/OWL: *cf.* www.w3c.org) for the description of profiles instances. These technologies are formal languages that have similar objectives than description logic languages: reasoning on terminologies or taxonomies. RDF-like formalisms are more adapted for semantic descriptions because they provide us with existing and re-usable predicates like: disjunction (*owl:disjointWith*), equivalence (*owl:equivalentClass*), equality (*owl:sameClass*), subsumption (*rdfs:subClassOf* et *rdf:type*), etc. Moreover, using these technologies enable, afterwards, the validation of our model with an existing application programming interface (API) for the semantic web called *Jena* that are able to interpret RDF/RDFS/OWL languages. Note that UML and RDF are not disjoint languages. The basic triple *[subject, predicate, object]* of RDF exists implicitly in UML and in any other language. Thus, associations between classes and relations between a class and its properties can be clarified with RDF triples.

3.2.1. Profiles instances

The *figure 2* illustrates two profiles: a user profile and an information profile. For each profile, we describe its logical structure, its contents, its logical structure semantics, and its contents semantics. This visualisation is done thanks to a tool that we have implemented for the construction, the visualisation and the matching of profiles that correspond to our generic model.

The logical structure and the contents of a profile are always described by a tree. The graph structure is obtained only when semantic elements are added. For instance, several logical structure or contents elements can be linked to the same semantic element. Moreover, semantic elements can be linked to each other by following a non hierarchical form. In *figure 2*, attributes *interests_centers*, *music* and *films* of the profile *user_profile_y* represent the same concept *dc:subject*.

Figure 2 also illustrate the re-usability of metadata coming from different namespaces like: *Dublin Core, Semantic Profile Namespace* (which is the namespace that we have created for this paper proposed framework), etc. These namespaces are respectively represented by the prefixes *dc:, sp:,* etc.

3.2.2. *Profiles interoperability*

The re-usability of metadata will ease models interpretation by mitigating the re-definition of concepts or datatypes and hence by reducing creation of semantic links of equivalence or equality between concepts and datatypes. Moreover, those metadata and links between them will define a shared semantics among profiles. This semantics will help to deduce elements of similar semantics, even if they do not have the same name. It will also allow reducing incoherence due to elements that have different meanings but identical names. Moreover, the semantics will define a level of interoperability between profiles (or interaction capacity), which will correspond here to the number of elements of compatible semantics. In *figure 2* for instance, if we look at the logical structure semantics of the two illustrated profiles, we note that there are attributes that shared the same concepts between those profiles. Let us take the leaf attributes *datesPreferences* and *information_date* of profiles *user_profile_y* and *information_profile_x* respectively, these attributes represent the same concept *dc:date* but if we look at their contents semantics, they are different. What should be interesting here, is to be able to deduce that those attributes can be compared thanks to some transformations that should have been clearly identified: here the transformation of a date format *mmyyyy* to a year format *yyyy*; and contents representation change: the contents element *(02/2003, 1.0)* should be transformed to contents element *(recent, 1.0)*. These transformations are described in the following section.

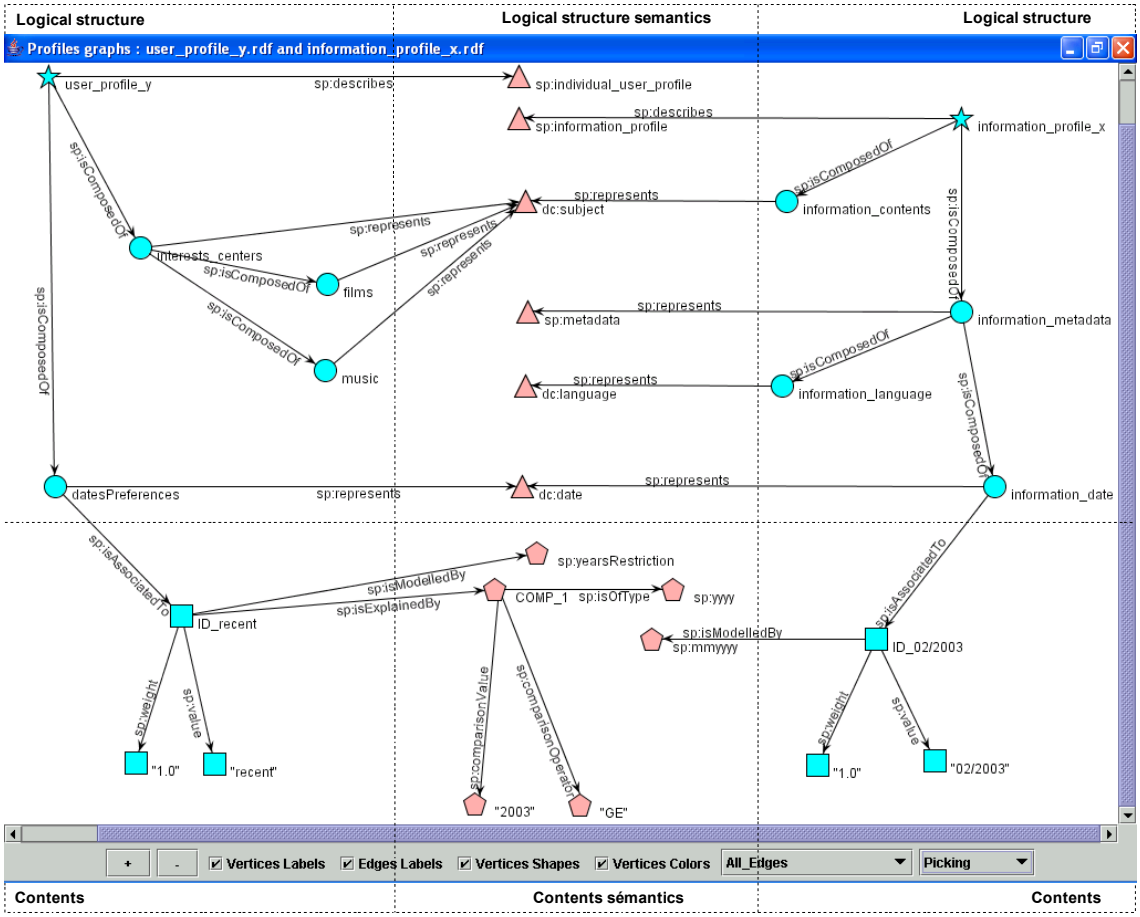

Figure 2. *Profiles instances and interoperability: logical structure, contents and semantics*

4. Matching flexibility for resources access

Describing the semantics of profiles logical structure and contents is a necessary step to an optimal exploitation of these last. In resources access, this exploitation is mainly based on profiles matching. In order to guarantee a coherent matching, we have to automatically deduce elements pairs of compatible semantics between the profiles to be compared. For that purpose, we have defined and implemented an algorithm, which is presented in the next section.

4.1. *Algorithm for compatible semantics elements detection*

This algorithm is based on profiles interoperability that itself is linked to the analysis of logical structure and contents semantics of profiles to be compared. Analysing the semantics related to logical structure and contents are both necessary because leaf attributes can represent the same concept while being associated to contents elements of different datatypes. For instance, attributes *datesPreferences* and *information_date* of *figure 2*. All the same, leaf attributes where elements are of the same datatype can represent different concepts. For instance, the *language of a piece of information* and the *user's interests centres* contents elements, can be related to the same datatype (string, for example) even though they describe different characteristics: *dc:language vs dc:subject*, for example.

The prior step to profiles matching is the selection of elements of compatible semantics between them that can be compared. *Figure 3* shows the general algorithm for this purpose, with its different steps and its different inputs and outputs. The steps of the algorithm are related: to the logical structure and its associated semantics analysis; and to the contents and its associated semantics analysis. These steps can also be divided in three classes: steps related only to graph patterns or paths (steps 1, 2, 4, 5); steps related to reasoning thanks to some verification of compatibility (steps 3, 6, 7) and finally the transformations step (step 8). All these steps use queries (or rules) written in SPARQL [19], an RDF query language.

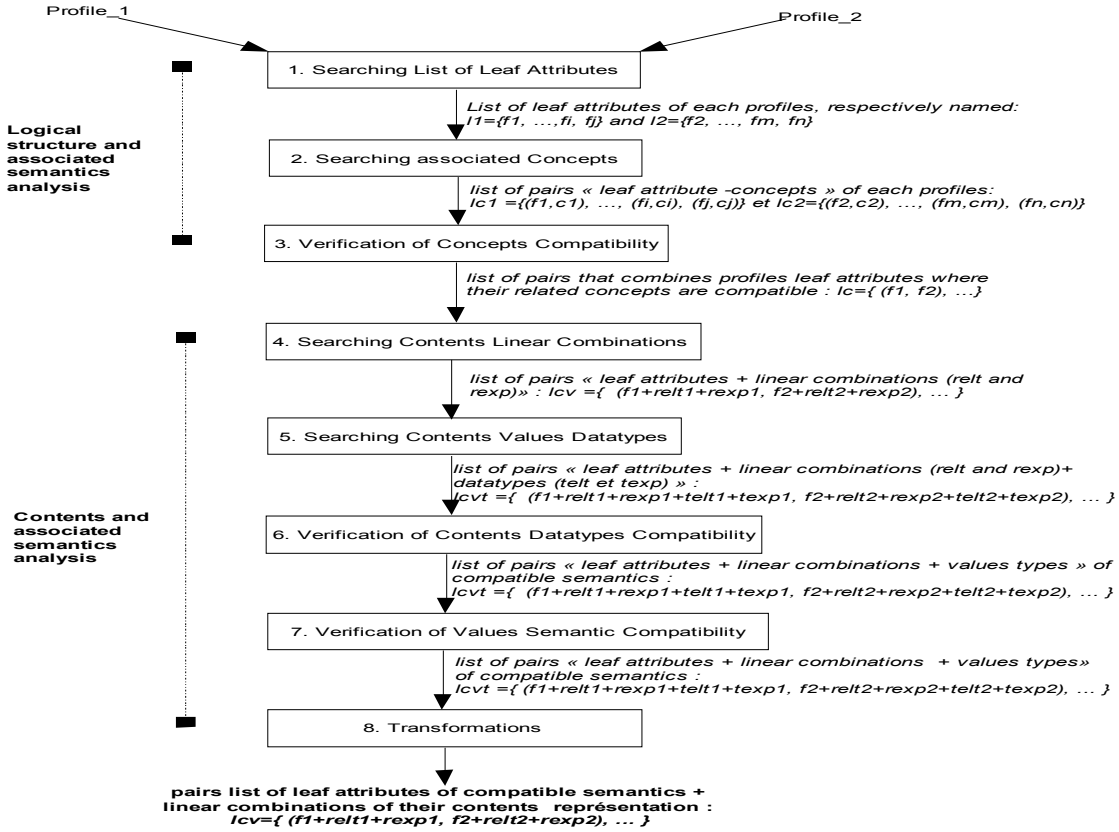

Figure 3. *General algorithm for profiles analysis: logical structure, contents and semantics*

Inputs of the algorithm are two profiles to be compared and outputs are a list of attributes pairs of compatible semantics with the linear combination of contents of each attribute. These attributes are always leaf attributes because they are the one related to contents elements. The matching of profiles or non leaf attributes is the result of an aggregation of leaf attributes matchings that compose it.

The logical structure and the associated semantics analysis allows the deduction of leaf attributes pairs that represent compatible concepts. This analysis defines a kind of *necessary rule* for the matching of two attributes coming from different profiles. Afterwards, we proceed to contents and associated semantics analysis that completes the previous one and makes it somehow sufficient. Finally, the linear combinations obtained at the end of the algorithm, will be used to measure the similarity degree between attributes of each couples by using *cosine formula* for instance.

4.1.1. *Logical structure and associated semantics analysis*

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX sp: <http://www.irit.fr/.../SemanticProfile#>

SELECT ?c1 ?c2
FROM Concepts.rdf
WHERE {
          ?c1 rdf:type sp:Concept .
          ?c2 rdf:type sp:Concept .
          { ?c1 owl:equivalentClass ?c2 }  UNION { ?c1 owl:sameAs ?c2 } .
FILTER (  (?c1=<"cf1"> || ?c1=<"cf2">) && (?c2=<"cf1"> || ?c2=<"cf2">)  ) .
        };
```

Table 1. *Verification of concepts compatibility.*

Analysing the logical structure and the associated semantics consists in:

a. *searching the list of leaf attributes with the associated concepts* for each profile to be compared;

b. *verifying the concepts compatibility* associated to any possible pair of leaf attributes of profiles to be matched. When leaf attributes concepts of a pair are different (come from different namespaces and/or have different names), one can check whether there are equivalent or identical. For that, we use a SPARQL query, similar to the one describe in *table 1*. Note that the key word *UNION* shows an alternative match, which means that at least one of the graph pattern (triple) involved, must be found for the query results. The key word *FILTER* defines a selection. On the other hand, properties *owl:equivalentClass* and *owl:sameAs* are defined as being symmetric and the values c_{f1} and c_{f2} are concepts names of given leaf attributes *f1* and *f2* of profiles *profile_1* and *profile_2* respectively. The file *Concepts.rdf* is a file that contains concepts and relations between them. It can be the result of a union of different files describing concepts related to a specific domain.

4.1.2. *Contents and associated semantics analysis*

For contents and associated semantics analysis, we consider that all contents element of a leaf attribute are of the same datatype, which means that they are modelled by the same instance of class *ValueType*. All the same, instances of class *LogicalExpression* that clarify contents elements of a given attribute are also of the same datatype. This is due to the fact that a leaf attribute is an elementary descriptive element and hence, has a homogeneous content. Thus, analysing contents and the associated semantics of a *necessarily compatible* (which means that the semantics of their concepts is compatible) pairs of leaf attributes consists in:

a. *searching linear combinations* (or vectorial representation) for representing contents associated to each attribute of a given pair: one representation is related to contents elements and the other to associated logical expressions that clarify (or explain) the meaning of these contents elements;

b. *searching contents datatypes* associated to each attribute of a pair: one is related to contents elements values and the other to the associated logical expressions;

c. *verifying contents datatypes compatibility* associated to each leaf attribute of a *necessarily compatible* pair: this verification is based on a library of datatypes transformations methods. Given two datatypes t_{f1} and t_{f2} of leaf attributes *f1* and *f2*, we check whether there is a method for changing a value of datatype t_{f1} to a value of datatype t_{f2} or conversely. The method name is a concatenation of the names of the two datatypes. We can also verify that compatibility by analysing semantic links that exist between datatypes. However, this alternative can be time consuming and useless if the transformation method does not exist;

d. *verifying values semantic compatibility* of contents elements associated to each leaf attribute of a *necessarily compatible* pair: this verification improve the flexibility of the analysis by using a file named *ContentsValues.rdf*, which is a result of the union between various taxonomies or terminologies described with RDF formalism like Wordnet RDF/OWL representation [1]. An example of that kind of verification is described in *table 2*. Note that the properties *sp:isATranslationOf, sp:isASynonymOf* and *sp:isAnAbbreviationOf* are defined as being symmetric. The symmetry helps to find symmetric triples, even if those triples do not exist physically in the queried file. val_1 and val_2 are two values of contents elements associated to leaf attributes *f1* and *f2* of profiles *profile_1* and *profile_2* respectively.

```
PREFIX sp: <http://www.irit.fr/.../SemanticProfile#>

SELECT ?v1 ?v2
FROM ContentsValues.rdf
WHERE {
                { ?v1 sp:isATranslationOf ?v2}
        UNION { ?v1 sp:isASynonymOf ?v2 }
        UNION { ?v1 sp:isAnAbbreviationOf ?v2 } .
FILTER (  (?v1=<" val1"> || ?v1=<" val2">) && (?v2=<" val1"> || ?v2=<" val2">)  ) .
        };
```

Table 2. *Verification of contents values compatibility.*

This type of query is useful because we can have for example the values *fr* and *french* that represent the same semantics, since *fr* is an abbreviation of *french*. This verification will allow defining a vectorial representation that is *semantically common* to attributes to be compared and that will enable us to evaluate, more faithfully, the similarity between these attributes.

e. *identifying transformations* to be performed on contents elements for the matching: the verification of contents datatypes compatibility can lead to some transformations, necessary for the matching. For example, if we want to compare attributes *datesPreferences* and *information_date* of *figure 2*, we do need to use a method that will extract a year from a date format.

On the other hand, we generally need to verify the contents vectorial represention of each leaf attribute as well as their dimension. We can have a *disjunction*, an *inclusion* or an *overlapping* between values of the vectorial representations of attributes to be matched. In order to perform this matching, it could be necessary to proceed to:

- a vectorial representation change by an *extension of values* of the different attributes contents elements, in order to describe them in the same dimension;

- a vectorial representation change by a *change of values* if at least one of the attribute to be compared can be represented in two different vectorial spaces. This is the case when attributes are clarified with logical expressions.

An example of those vectorial transformations is illustrated in the next section.

4.1.3. *Example of contents representation transformations*

Given two attributes semantically compatible (according to the compatibility of their concepts and datatypes):

- *datesPreferences* which contents is *{(recent, 1.0), (notRecent, 0.5)}*. Where the value *recent* represents a restriction to dates greater or equal to the year 2003 and the value *notRecent* represents a restriction to dates less than the year 2003;

- *information_date* which contents is the date (02/2003, 1.0).

The results of the contents analysis returns two linear combinations for these attributes: one related to contents elements and the other to logical expressions. Thus:

- for attribute *datesPreferences* (named $a_{fs)}$, we obtain:

$a_{fsexp} = 1.LT2003 + 1.GE2003$
$a_{fselt} = 0,5.notRecent + 1.recent$

- for attribute *information_date* (named a_m), we initially get a_{mexp} = 1.GE2003 after the extraction of the year from the date 02/2003. Then, we proceed to a change of this vectorial representation and we finally obtain:

$a_{mexp} = 0.LT2003 + 1.GE2003$
$a_{melt} = 0.notRecent + 1.recent$

Note that GE stands for *« >= »* and *LT* for *« < »*.

These two attributes will be compared by using the representation related to contents elements a_{fselt} and a_{melt} and by applying the cosine formula for instance. An aggregation method of leaf attributes matching in order to compare profiles are described in [9] and evaluated in [22].

4.2. *Algorithm evaluation*

We have evaluated the proposed algorithm based on our *profile generic model,* in comparison to methods based on an *attribute-value profile model* and on a *logical structure profile model*.

For that, we have defined 10 users' profiles describing users' interests, preferences (dates, languages, sizes of documents) and demographic data (name, gender, professional institution). Users' profiles are described by logical structure, contents and semantics.

We have also described documents of the CLEF evaluation campaign which contains articles of the year 1994, for journals: Swiss National News Agency (*SDA*), Le Monde (*LeMonde*) and Los Angeles Times (*LaTimes*). *Table 3* describes some properties of these collections.

Collections	***SDA 94***	***LeMonde 94***	***LaTimes 94***
Size	82.1 Mo	156 Mo	420 Mo
Number of documents	43 178	44 013	113 005
Language	French	French	English

Table 3. *Description of collections SDA 94, LeMonde 94 and LaTimes 94*

The RDF description of the CLEF documents follows their Document Type Definition (*DTD*) for the logical structure. Contents have been extracted from documents and the semantics related to logical structure and contents elements have been defined after looking at some documents of each collection. The DTD of each collection is different and described several aspects of the documents (identification, title, authors, date, paragraphs, and so on) but the semantics of some elements are similar. For example, authors of articles are defined by the words: *AU* for SDA articles, *AUTHOR* for LeMonde articles and *BYLINE* for LaTimes articles. These logical structure elements (attributes) can be related to the concept *dc:creator* of the Dublin Core for instance.

The database of profiles obtain is enough heterogeneous for our experiments. The experimentations consist in detecting the number of leaf attributes pairs of compatible semantic among those profiles. For that, we first apply our algorithm. Then, we consider the case of an *attribute-value model* where attributes of similar semantics are leaf attributes that have the same name. Finally, we consider the case of a *logical structure model* where attributes of similar semantics are leaf attributes that have the same name and that have also the same path in the logical structure. The comparative results of experiments are presented in *table 4*. Note that the model of profile proposed in this paper is called *Semantic Profile*.

Profiles models	***Semantic Profile***	***Attribute-value***	***Logical structure***
Average Number of attributes pairs of similar semantics	4,84	0,92	0,79

Table 4. *Comparative results for the detection of attributes of similar semantics*

We can notice that our algorithm outperforms the others. The semantics added acts as a shared part between profiles and allows detecting pairs that do not have the same name but share the same semantics, which is not possible with an attribute-value or logical structure model. On the other hand, attributes that have the same name, do not share automatically the same semantics. So, with respect to that and to the manual verification of some attributes pairs returned, we can say that the pairs detected with our algorithm are more trustworthy than those obtained with the others models.

4. Conclusion and discussions

In this paper, we have proposed a profile generic model and an analysis method for profiles matching based on semantics. The aim is to provide applications with flexibility for profiles modeling and for profiles matching as well. We have evaluated the proposed analysis algorithm for profiles matching with a java API called *Jena*, which enable us to combine a programming language *java* with an RDF query language *SPARQL*. The results shows that we can automatically deduce more trustworthy attributes pairs of similar semantics compare to methods based on an attribute-value model or a logical structure model.

For the future work, we plan to define how we can take into account semantic links, not always symmetric, for compatibility verification like the subsumption link with predicates *rdf:type* and *rdfs:subClassOf*. These predicates are transitive. For instance, the substitution of the value *vehicle* by the value *car* should imply a weight modification. All the same, taking into account subsumption links (*rdf:type* and *rdfs:subClassOf*) on concepts instances that represent attributes, require to define a particular reasoning procedure for their interpretation in profiles analysis.

10. References

[1] M. v. Assem, A. Gangemi, G. Schreiber, editors. RDF/OWL Representation of WordNet. *Editor's Draft*, http://www.w3.org/2001/sw/BestPractices/WNET/wn-conversion.html#querying, 2006.

[2] R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval*, First edition, Addison Wesley, ISBN 0-201-39829-X, 1999.

[3] T. Berners-Lee, J. Hendler, O. Lassila. The semantic web, *Scientific American*. 2001.

[4] M. Boughanem, C. Chrisment, C. Soulé-Dupuy. Query modification based on relevance backpropagation in adhoc environment, *Information Processing \& Management Journal*, Elsevier Science, vol. 35, p. 121-139, 1999.

[5] M. Bouzeghoub, D. Kostadinov. Personnalisation de l'information: aperçu de l'état de l'art et définition d'un modèle flexible de profils, *COnférence en Recherche d'Informations et Applications (CORIA'05)*, p.201-218, 2005.

[6] S. Buchholz, T. Hamann, G. Hubsch. Comprehensive Structured Context Profiles (CSCP): Design and Experiences, *Workshop on Context Modeling and Reasoning (CoMoRea'04)*, p. 43-47, 2004.

[7] B. Chang, J. Kesselman, R. Rahman. Document Object Model (DOM) Level 3 Validation Specification Version 1.0, *W3C Recommendation*, http://www.w3.org/TR/2004/REC-DOM-Level-3-Val-20040127/, 2004.

[8] M. Chevalier, C. Soulé-Dupuy, P. L. Tchienehom. Profiles Semantics and Matchings Flexibility for Resource Access, *International IEEE conference on Signal-Image Technology & Internet-based Systems (SITIS'05)*, p. 224-231, 2005.

[9] M. Chevalier, C. Soulé-Dupuy, P. L. Tchienehom. A profile-based architecture for a flexible and personalized information access, IADIS International Conference (IADIS/WWW Internet), vol 2, p. 1017-1022, IADIS - ISBN 972-99353-0-0, 2004.

[10] Y. H. Cho, J. K. Kim, S. H. Kim. A personalized recommender system based on web usage mining and decision tree induction, *Expert System with Applications*, vol. 23, n°3, p. 329-342, 2002.

[11] P. Dolog, W.Nejdl. Challenges and benefits of the Semantic Web for User Modelling, *In proceeding of AH'03*, 2003.

[12] D. Goldberg, D. Nichols, B. M. Oki, D. Terry. Using Collaborative Filtering to weave an Information Tapestry, *Communications of the ACM, Information Filtering*, vol. 35, n°12, p. 61-70, 1992.

[13] G. Klyne, F. Raynolds, C. Woodrow, H. Ohto, J. Hjelm, M. H. Butler, L. Tran, editors. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, *W3C Recommendation*, http://www.w3.org/TR/CCPP-struct-vocab/, 2004.

[14] A. Kobsa. Generic User Modelling Systems, *User Modelling and User-Adapted Interaction*, vol. 11, p. 49-63, 2001.

[15] M. Montaner, B. Lopez, J. L. D. L Rosa. A Taxonomy of Recommender Agents on the Internet, *Artificial Intelligence Review*, vol. 19, pages 285-330, Kluwer Academic Publishers, 2003.

[16] P. A. Muller, N. Gaertner. *Modélisation objet avec UML,* Deuxième édition, Eyrolles, ISBN 2-212-09122-2, 2000.

[17] M. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering, *Artificial Intelligence Review,* 1999.

[18] J. Pitkow, N. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, T. Breuel. Personalized Search: A contextual computing approach may prove a breakthrough in personalized search efficiency, *Communications of the ACM*, vol. 45, No. 9, p. 50-55, 2002.

[19] E. Prud'hommeaux, A. Seaborne. SPARQL Query Language for RDF, *W3C Working Draft*, http://www.w3.org/TR/2004/rdf-sparql-query/, 2005.

[20] J. Rucker, M. J Polanco. Siteseer: Personalized Navigation for the Web, *Communications of the ACM*, vol. 40, n°3, pp. 73-75, 1997.

[21] B. N. Schilit, N. L. Adams, R. Want. Context-aware computing applications, *In IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

[22] P. L. Tchienehom. Modèle générique de profils pour la personnalisation de l'accès à l'information, *23ième Congrès National Inforsid'04*, p. 269-284, 2005.