Modeling and Querying Context-Aware Personal Information Spaces

Rania Khéfifi[1], Pascal Poizat[1,2], and Fatiha Saïs[1]

[1] LRI, CNRS and Paris Sud University,
[2] Evry Val d'Essonne University,
{rania.khefifi, pascal.poizat, fatiha.sais}@lri.fr,

Abstract. Personal information management (PIM) is the practice and analysis of the activities performed by people to acquire, organize, maintain, and retrieve information for everyday use. It is characterized by its heterogeneity, its dispersion and its redundancy. In this paper, our reflection focused on the development of a model that will be user-based. It help him create his data model depending on his needs.The meta-model that we propose allows users creating personal information and organizing them according to different points of view (ontologies) and different contexts. Contextual queries are defined to allow users to retrieve their personal information depending on context valuation.

Keywords: Personal Information Management, Ontologies, Contextual data

1 Introduction

Personal information management (PIM) is the practice and analysis of the activities performed by people to acquire, organize, maintain, and retrieve information for everyday use. PIM is a growing area of interest because, everyone is looking for better use of our limited personal resources of time, money and energy. There exist several personal information management system [2–4]. MEMEX (Memory Extender), that allows automatic creation of references between library objects and establishing links between pieces of information. SEMEX [3] offers users a flexible framework with two main goals: browsing personal information by semantically meaningful associations and enrich personal information space to increase users' productivity, it uses data annotation, similarity computation from ontology-based framework. PIMO [4] is a framework for representing personal information model. It is a framework using multiple ontologies to represent concepts and documents using by persons in their daily activities. The increasingly big amount of personal information (e.g., mails, contacts, appointments) managed by a user is characterized by its heterogeneity, its dispersion and its redundancy. Processing this amount of information is difficult and not obvious. Indeed, personal information is often managed by very specific and autonomous tools which deal with specific kinds of user data (***e.g.*** *iCal*

to manage appointments and meetings, *thunderbird* and *outlook* to manage e-mails). Despite the existence of various applications, there are no connections and no links between the pieces of personal information managed by the user applications. Indeed, many data present in one application can relate and concern other applications which leads to the redundancy of personal information. PIM users may create objects with the same name by referring to different entities and may create different objects with different names by referring them to the same entity. This kind of behavior may conduct to the semantic heterogeneity of personal information which makes the task of integration and interoperability between personal information more difficult. This is the reason why in this work we have developed an ontology based approach to deal with semantic heterogeneity as much as possible. Our system that we present in this paper, offers users the ability to organize their personal information in a way to ensure their semantic consistency, their reusability and their transparent querying. It is based on a meta-model allowing the user to create its own personal data model using different existing domain ontologies to describe and organize its personal information. We have studied the problem of designing model to represent data with context and selecting the most appropriate data according to a context asked in the query [1]. Martinenghi and Torlone in [5] have proposed a logical model as a foundation of context-aware database management system, and they have used many context to describe their data. Furthermore, user' personal information can be described according to different points of view thanks to the semantic Web technologies (***e.g.*** OWL, RDF, SPARQL) allowing a richer descriptions of personal data.

2 Preliminaries

We present in this section the semantic Web materials on which our approach is based: OWL ontology model (`http://www.w3.org/TR/owl2-profiles/`), RDF data model and SPARQL query language.

OWL ontology We consider an OWL ontology $\mathcal{O}$ that is defined as a tuple $\mathcal{O} = \{\mathcal{C}, \mathcal{P}, \mathcal{R}\}$ where $\mathcal{C}$ is a set of classes, $\mathcal{P}$ is a set of properties and $\mathcal{R}$ is a set of relations (subsumption, partOf, equivalence, disjointness).
Figure 2, gives an example of two ontologies O1 and O2. Each represents a part of the set of classes $\mathcal{C}$, partially ordered by the subsomption relation (pictured by '$\rightarrow$'). The properties are pictured by bubbles linked to their corresponding domain and range.

RDF data model RDF (Resource Description Framework) is a language which allows to describe semantically data in a understandable way for machines. Using RDF, data is represented as a set of triples expressed in the N-TRIPLE syntax as: <*subject property object*>. RDF language allows to use *reification* mechanism to add new elements to the descriptions of the RDF declarations, like data author, creation date, etc. For the reification semantics, see (`http://www.w3.org/TR/rdf-mt/#Reif`).

3 Personal Information Modelling

In this section, we will present the main components of the personal information representation model that we have designed. In Figure 1, we show the the three levels of this model: *(i) Point-of-View level*, *(ii) Personal Information Type Level*, and *(iii) Personal Information Instance level*. The *Personal Information*

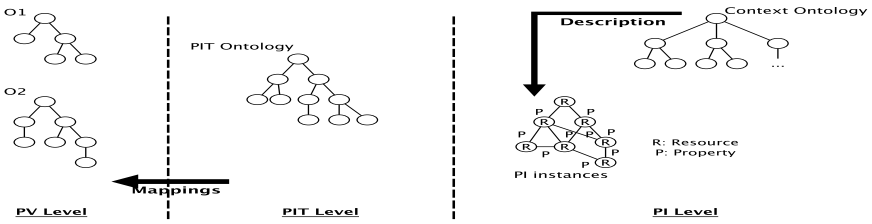


Fig. 1: Personal Information Model Architecture.

Type (PIT) level represents the model of a structured vocabulary that the user aims to use to describe his/here personal information. The *Personal Information Instance (PI) level* represents the personal information that are instances of the PIT model. A set of mappings is added between the user PIT model and the ontologies that are considered in the *Point-of-View (PV) level*. As it is shown in Figure 1, personal information can also be represented according to different contexts, e.g. geographical, social and temporal (for a survey on context-aware systems, see [1]). Indeed, personal information is context-aware, which means that the concrete values of the properties may variate in function of the considered context. For example, a person may have a personal address and a professional address.

3.1 Point-of-View level

We consider that a point of view is an ontology. It is composed of a set of classes (concepts) representing a specific domain. We allow declaring disjunction constraints between classes of a same point-of-view as well as between classes of different points-of-view.

3.2 Personal Information Type level

This level is dedicated to represent the *Personal Information Types* (PITs) defined by the user and represented in an OWL ontology. A PIT is defined by a class for which a set of properties P is associated. We define a set of mappings between PIT classes (resp. properties) and Point-of-View classes (resp. properties) in order to exploit existing knowledge that are declared in PV ontologies.

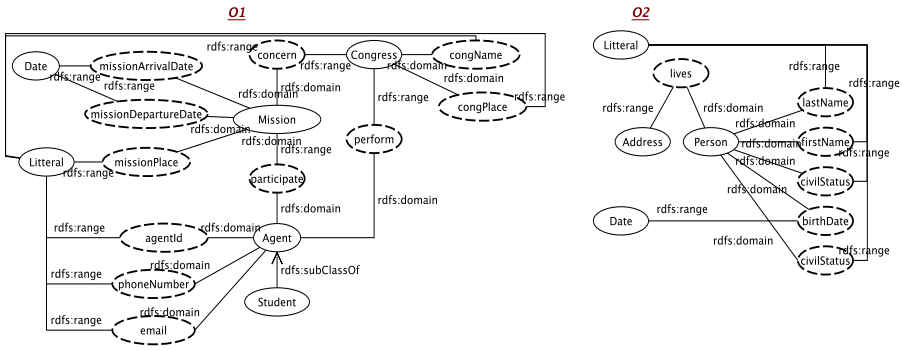


Fig. 2: Sample of some point-of-view ontologies.

Definition 1 *–**Personal Information Type (PIT)**. A PIT is defined by a tuple* $(c, \mathcal{P}, \mathcal{C}, M_p)$ *where:*

- c *is the class that defines the PIT. A label is associated to the class.*
- P *is a set of properties associated by the user to the class* c *from a set of candidate properties of the PV ontologies.*
- $\mathcal{C}$ *is the set of classes that are involved in the PIT property definition. These classes should not be pairwise disjoint, w.r.t. the disjunction constraints that are declared between classes of the different ontologies.*
- $M_p = \{(c.p_1 \equiv o_1 : c_2.p_2), (c.p_1 \preceq o_2 : c_3.p_3), \ldots, (c.p_j \equiv o_k : c_n.p_m.)\}$ *is the set of mappings between the properties* P *of the PIT and properties of the PV ontologies. These mappings correspond to a subset of the whole set of mappings* $\mathcal{M}_P$ *between the PIT model and the PV model.*

Example 1 *Let there two ontologies: a* work *ontology* O_1 *and a* family *ontology* O_2*. The* $Person$ *concept is described by the set of properties* {lastName, firstName, e-mail, job} *and by the set of properties* {lastName, firstName, birthDate, civilStatus,numberOfChildren} *in the ontology* O_2*. In spite of the existence of two different descriptions in ontologies* O_1 *and* O_2*, the user can choose to create a new concept (**e.g.** Friend) represented by a new PIT having as name* $Friend$*. The new PIT* $friend$ *can be described by a subset of properties derived from the two sets of properties of the concepts* $O_1 : Employee$ *and* $O_2 : person$*, (**e.g.** describe* Friend *by name and* $firstName$ *derived from* $O_1 : Employee$*, and civilStatus derived from* $O_2 : Person$*). A set of mappings is then generated between the user PIT ontology and the PV ontologies* O_1 *and* O_2*:*

$$\mathcal{M}_p = \{(Friend.firstName \equiv O_1 : Employee.firstName), (Friend.lastName \equiv O_1 : Employee.name), (Friend.email \equiv O_1 : Employee.email), (Friend.civilStatus \equiv O_2 : Person.civilStatus)\}.$$

3.3 Personal Information Instance level.

Once the different PITs needed by the user are defined, the personal information (***e.g.*** contacts, mails, publications, appointments) can be attached to these PITs and then instantiate the different properties describing these PITs. As it is

shown in Figure 1, our personal information instances are context aware. Hence, instead of having a value for each mono-valued property we consider a set triples *(property, context, value)*.

Definition 2 –***Personal Information(PI)*** *is an instance of a PIT which refers to a user information. Each PI is also represented by a subset of the property instances which describe its corresponding PIT. We consider a PI as defined by a tuple* $(T, Reference, \mathcal{P}_{inst})$*, where:*

- T*, represents the PIT that is instantiated by the PI.*
- $Reference$*, is URI which is an identifier used to represent the Personal Information instance in the system.*
- $\mathcal{P}_{inst}$*, is a set of tuples* $(property, value, context)$ *which instantiates the PIT "T" properties.*

We consider that the context values are declared in a given context ontology. In this work we will focus our study on the geographical context of personal information. We consider that the geographical ontology is composed of a set of classes that are organized in a hierarchy thanks to the *partOf* relation. We assume that one class cannot be *partOf* more than one class. We assume also the declaration of disjointness constraints and equivalence constraints between the geographical context classes. In order to represent in RDF the context values associated to the PI instances (***e.g.*** a person leaving partly in France and in UK may have two Social welfare numbers (SSN)), we need to describe RDF statement, representing PI, using an other RDF statement. This is possible thanks to the reification mechanism. Then, Personal information are represented in RDF as a set of statements. Each statement instantiates one PIT property by assigning to the statement a URI *uriRef* as follows:

```
<uriRef rdf:type rdf:Statement>  <uriRef rdf:subject subj>
<uriRef rdf:predicate property> <uriRef rdf:object value>
<uriRef :contextGO context>
```

where, *subj* defines the PI URI, *property* expresses the URI of the property, *value* represents the value given to the instance of property and *context* is the value of the context, in our case is the geographical context.
An example of RDF representation of "phoneNumber" property from Friend#001 is done as follows:

<t1 **rdf:type** rdf:Statement>	<t2 **rdf:type** rdf:Statement>
<t1 **rdf:subject** Friend#001>	<t2 **rdf:subject** Friend#001>
<t1 **rdf:predicate** O2:phoneNumber>	<t2 **rdf:predicate** O2:phoneNumber>
<t1 **rdf:object** "phone11">	<t2 **rdf:object** "phone12">
<t1 **:contextGO** France>	<t2 **:contextGO** England>

4 Context-based SPARQL querying

It is important to provide the user a query interface with his PIM in a transparent way. This is possible, thanks to the PIT ontology that is used to describe personal information. The user simply expresses his queries using the vocabulary defined

in the PIT ontology. Furthermore, in order to exploit the contextual values, the user may express his queries by specifying the desired context value. We have defined three kinds of contextual-queries:

1. *Strict-context query*: a query where the user asks for answers having the same context than the one given in the query;
2. *Disjoint-context query*: a query where the user asks for answers having a context that is distinct from the one given in the query;
3. *Similar-context query*: a query where the user asks for answers having a context that is similar to the one given in the query.

4.1 Strict Query Execution algorithm (SQE)

We perform our queries by giving as parameters a context and a function between contexts that we seek to have. The function between contexts belongs to this operator set $\{=, \neq, \sim\}$. The algorithm returns all tuples whose valid context of these values in context. The disadvantage of these results is that sometimes the data is represented in a more generic and that the fact that data is not asked in the context when there are other data in a context that is more generic context requested by the query is near (the notion of nearness is processed by the notion of hierarchical similarity detailed in subsection 4.3). $\|q^{rc}\| = \bigcup_{c' \in Valid(\mathcal{O},r,c)} (\|f_1(q,c')\|^{\text{CSPARQL}}) \times \delta_1(r,c,c')$ where:

- $Valid(\mathcal{O}, r, c)$ is a function that determines the list of contexts similar to a given context c according to the function r based on an ontology $\mathcal{O}$.
- $\delta_1(r, c, c')$ relevance degree that allows to determine the relevance of the returned value according to the query context.(if $c = c'$ then $\delta_1 = 1$, if $c \neq c'$ then $\delta_1 = 1$, if $c \sim c'$ then $\delta_1 = \Re(c, c')$, $\Re$ is a function using similarity measure that compute relevance degree (see subsection 4.3)).
- f_1: takes as parameter a user query "q" and a context "c'" and returns all answers where values have the context "c'".

4.2 Flexible Query Execution algorithm (FQE)

Our solution consist to return to the same tuple of values having a similar context to context requested by the query and to evaluate these values by a degree of relevance calculated by reference to the context of the query and the ontology of context. For example, if the user requests a phone number and an address of a contact that can be valid in *France*, and we assume that the data of the user's contacts are valid in different contexts: taking the example of a contact where his phone number is given in *Europe* and the address is given in *Paris*. Using Algorithm 1, this tuple will not be returned by the query. Indeed, in algorithm 1 we consider two constraints *(i)* the contexts of values belonging to the same tuple must be the same *(ii)* the context of the tuple must verify the function $Valid(\mathcal{O}, r, c)$. In Algorithm 2, we remove the first constraint to allow obtaining more flexible tuples, which can be composed by several values in different contexts. The algorithm execution of this query is given as follows: $\|q^{rc}\|^{\mathcal{O}} = h_1(g_1(\|f_2(q)\|^{\text{CSPARQL}}, \mathcal{O}, r, c), \epsilon)$

- f_2: function takes as parameter only user query and returns all answers satisfying query.

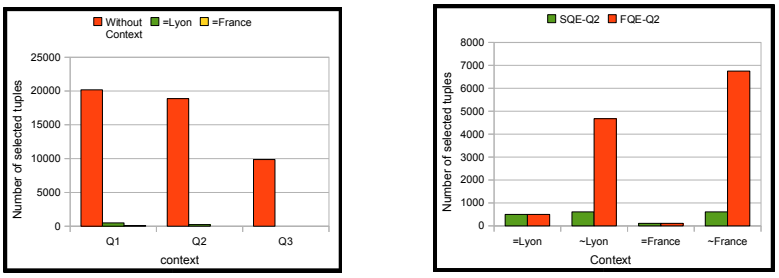
- g_1: is a function allowing to select answers resulting from f_1, according to context "c" and satisfying "r" function and compute relevance score for each answer.
- h_1: is the final selection where the function select answer that have a relevance degree greater than ϵ.

4.3 Relevance degree computation between contexts

In case where there are no answers for the user query, because of the absence of user context in the personal data, the user query is rewritten into a set of queries where the user context c is replaced by all the contexts c' that are similar to. To compute this similarity between two contexts, we exploit the hierarchical structure of our context ontology. We are choose in this work to use Wu and Palmer measure [6], but we can also use other one. The similarity computation principle of this measure is: Given an ontology $\mathcal{O}$ composed of a set of nodes and a root node R. A and B represent two ontology elements of which the we aim to compute the similarity. The similarity score is computed as function of the distances: (i) $N1$ respectively $N2$ which represents the distance of A from their LCS (Least Common Subsumer), respectively, the distance of B from their LCS, and (ii) the distance N which computes the distance between the LCS of A and B and the hierarchy root R. The similarity measure of Wu and Palmer is computed by the following formula: $Sim_{WP}(A,B) = \frac{2\times N}{N1+N2+2\times N}$. The relevance degree $\delta(A,B)$ between two classes A and B is obtained as follows: $\delta(A,B) = 0\ if\ \forall A \not\equiv B,\ 1\ if\ \forall A \equiv B\ \vee\ (B \leq A) \vee\ (A \leq B)\ or\ Sim_{WP}(A,B)$

5 Experimentation

To evaluate the efficiency of our approach, we have performed our system on real datasets. Our dataset is composed of a set of user contacts which is described by name, address, birth date, phone number, ..., and each one of those properties is validate in different geographical contexts, the size of our test file is 22177 tuples. The aim of our first test is to present the usability of context and the importance of the different function that we have defined to execute queries ($Equal$, $Similar$, $Disjoint$). Figure 3(a) presents queries results, we have execute query in three cases: (i) without context *Q1*,(ii) context is "$= Lyon$" *Q2* and (ii) context is "$= France$" *Q3*. We observe that *Q1* returns a lot of tuples unlike in the case of *Q2* and *Q3*, we observe that using context decrease the number of selected tuples because context is considered as filter applied to the query. The second test consists on comparing SQE and FQE algorithms. We have applied the two algorithm on queries with two different contexts and using equal and similar function between context. The first query returns about 12 answers and the second about 7000 answers. The number of answers in each case is shown in figure 3(b). In figure 3(b), we observe that when the context function is equality we found the same number of selected tuple using SQE and FQE and in the case of similarity context function, we obtain a large number of answers for FQE comparing tothe case of SQE algorithm. This is due to the fact that SQE



(a) (b)

Fig. 3: (a) Results with and without context (b) SQE vs FQE: number of answers

algorithm require the same context for all values of the same answer, while for *FQE* we can have different context on the same answer. The conclusion of this experimentations is the use of one or other of the algorithm depends on the user needs and on the nature of its application.

6 Conclusion and future work

In this article, we have presented our personal information management system which is context-aware and ontology-based. Our meta model is defined to deal with several problems: (*i*) heterogeneity, (*ii*) redundancy and (*iii*) dispersion. It exploits ontologies to deal with semantic heterogeneity and to take benefits from the intensive work on knowledge representation and reasoning in this domain. The proposed contextual querying allows the system to give the user more relevant answers. Our model applies to multiple-user application, for example help the user to fulfill his form of an automatic and fast way using there stored data. In the future, we plan to continue the work over personal information querying, the automatic creation of e-governance process described in the form of workflows using contextual queries.

References

1. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
2. Bush, V.: As we may think. The Atlantic Monthly 176(1), 101–108 (1945)
3. Dong, X., Halevy, A.: A platform for personal information management and integration. In: Proceedings of VLDB 2005 PhD Workshop. p. 26. Citeseer (2005)
4. Leo S., Ludger V. E., A.D.: Pimo - a framework for representing personal information models. Proceedings of I-Semantics pp. 270–277 (2007)
5. Martinenghi, D., Torlone, R.: Querying context-aware databases. Flexible Query Answering Systems 5822, 76–87 (2009)
6. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. 32nd Annual Meeting of the Association for Computational Linguistics pp. 133–138 (1994)