OntoPIM: How to Rely on a Personal Ontology for Personal Information Management

Vivi Katifori[2], Antonella Poggi[1], Monica Scannapieco[1], Tiziana Catarci[1], and Yannis Ioannidis[2]

[1] Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "La Sapienza" - Via Salaria 113, I-00198 Roma, Italy
surname@dis.uniroma1.it

[2] Department of Informatics and Telecommunications
University of Athens - 157-84, Ilisia, Athens, Hellas, Greece
vivi@mm.di.uoa.gr, yannis@di.uoa.gr

1 Introduction

Nowadays, our personal computer contains a huge amount of information, that is stored in several different formats, including emails, pictures, text documents, media file, address books, etc. When we need to look for some information, one possibility is to use a keyword-based search tool, such as Google Desktop [1]. We then get several links to documents, mails, databases, etc. that relate to our search but are often too scattered in order to let us easily obtain the information we are looking for, even if this information is actually contained in our desktop.

In this paper, we propose a framework for Personal Information Management (PIM), called *OntoPIM*, that relies on the use of a *Personal Ontology*, that describes user's domain of interest in terms of objects, classes and relations. The ontology is personal in the sense that it reflects the user view of her domain(s) of interest. It is used to assign a semantics to the information contained in the user desktop, as well as to query the system in order to obtain a certain information. Then, by relying on the Personal Ontology, our framework overcomes the limitations of desktop search tools available nowadays. In particular, by the use of the *Semantic Save*, it provides the user the possibility to store any object of interest according to its semantics, i.e. to relate it to the concepts of the Personal Ontology, where an object may be a mail, a document, a picture, or any other type of data. Then, the user is able to query the Personal Ontology, whereas the system carries out the task of suitably processing the query, accessing the different pieces of information involved in the query, and assembling the data into the final answer.

The main contributions of this work are therefore (i) the framework definition for Personal Information Management using a Personal Ontology, and (ii) the architecture for the system, that encompasses heterogeneous data wrapping, data integration and personalization tools. This work is part of a wider project called TIM - Task-centered Information Management - under development in the frame of the DELOS NoE [2]. TIM has the two main goals of (i) classifying personal information by means of a user-tailored ontology, and (ii) allowing task-oriented interaction with one's own PC. In this paper, we focus on the first goal.

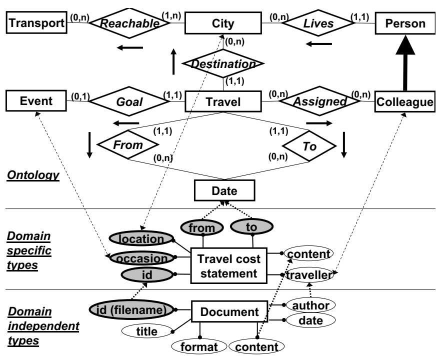
The paper is organised as follows. In Section 2, we illustrate the use of the Semantic Save. In Section 3, we discuss the architecture of the system. Then, in Section 4, we present the formal framework underlying the OntoPIM system.

2 Semantic Save

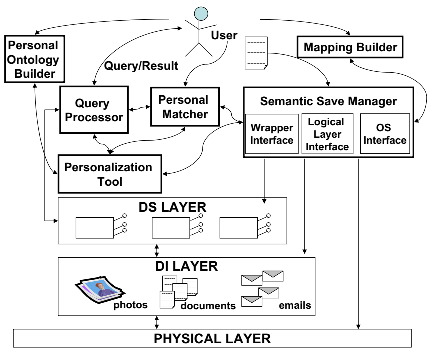
In this section, we illustrate how the OntoPIM *Semantic Save* works. Suppose that we have filled our last travel cost statement. We then proceed as follows.

- First, we indicate that we are saving an object of type **document**. The system extracts from the document a set of metadata, e.g. the **author** and the **date**. The objects that are created in this step are called *domain independent (DI) objects*, since they may exist in every domain and have always the same set of attributes.
- Second, we specialize the type of the data with respect to a particular domain. In our scenario, we indicate that the document we are saving is an object of type **travel cost statement (TCS)**, that is one of the *domain specific (DS) types* that are associated with the **business domain**. Thus, a new DS object of type **TCS** is created, whose part of the attributes is automatically mapped from part of the attributes of the DI object. This is the case of the attribute **traveller** in our example. We then may be asked to enter some other attributes associated with the **travel cost statement** DS type, as for example the **location** and the **occasion** of the travel.
- Finally, the system maps the attributes of interest of the newly created object of type **TCS** to concepts of the Personal Ontology. Note that this step is performed automatically, thanks to a set of rules, called *mappings*, that characterize each DS type and are specified when the DS type is newly created. The semantics of these mappings is that each attribute value becomes a *representation* of an instance of the concept to which the attribute is mapped. In our scenario, OntoPIM maps the attribute **traveller** of the travel cost statement to the concept **colleague**. Similarly, it maps the **location** and the **occasion** respectively to the concepts **city** and **event**.

The result of the performed Semantic Save is graphically represented in Fig. 1(a), where the ontology is represented in the flavor of a simplified Entity-Relationship model.



(a) Semantic Save result



(b) OntoPIM architecture

Fig. 1.

3 The OntoPIM Architecture

The OntoPIM architecture is shown in Figure 1(b). Note that all the modules interact with three diffent data layers that, starting from the bottom, are: (i) the physical layer, storing files or relational tables or any other physical objects that can be stored on a PC;

(ii) the first wrapper layer (DI Layer) representing domain independent (DI) objects from the physical layer, such as emails, documents, photos etc., and (iii) the second wrapper layer (DS Layer) representing domain specific (DS) objects that correspond to domain specific types, such as the travel cost statement of the running example.

In what follows we describe the main OntoPIM modules.

– The user interacts with the **Personal Ontology Builder (POB)** in order to build her own Personal Ontology. Such representation is intended to be completely independent of the physical representation of information.
– The **Personalization Tool (PT)** interacts with the POB, to automate the creation and the modification of the ontology on the basis of an appropriate user profile. Moreover, the PT is responsible for automating the Semantic Save function to some extent, proposing itself possible concepts to be associated with the document, completing queries with things implied by the user, etc.
– The **Mapping Builder (MB)** allows the user to create and modify her DS types. By interacting with the user, it establishes the correspondence between DS objects of the DS Layer and concepts of the Personal Ontology. This specification is then translated into the set of rules that constitutes the set of *mappings* that will be formally introduced in the next section.
– The **Semantic Save Manager (SSM)** takes as input a physical object **o** and uses the mapping created by the MB module to perform the Semantic Save by: (i) invoking the operating system in order to save **o** in the OS file system, (ii) creating the DI abstraction of **o** and (iii) linking it to the corresponding wrapper.
– The **Personal Matcher (PM)** performs instance matching. It is responsible for identifying attribute values of different DS objects as representing the same real world entity. It produces as output the set of *matching rules* that describe how to perform the matching. These rules will be formally presented in the next section.
– The **Query Processor (QP)** is responsible to process and answer the queries posed by the user over the Personal Ontology. More specifically, the QP exploits the abstraction created by the SSM, the mapping created by the MB and the rules produced by PM, in order to rewrite the query in terms of queries to wrappers, that retrieve the actual data from the physical layer.

4 Formal Framework

In this section, we introduce the formal framework underlying the OntoPIM system, that encompasses two main functions that are the Semantic Wrapping and the Semantic Integration. The former aims at overcoming the personal data heterogeneity and its primitive lack of semantics by presenting the information contained in its mails, documents, etc. as data tuples of relations that are meaningful with respect to the user's domain of interest. On the other hand, the Semantic Integration function lets the user query the ontology, that represents its personal, integrated view of its domain of interest, while the system carries out the task of suitably retrieving, reconciling and assembling the actual data. Because of lack of space we will focus here on the more challenging part of the system, i.e. the Semantic Integration. In particular, this makes use of a simple description logic, called DL-Lite [4], to describe the Personal Ontology provided to the user. DL-Lite is tailored to capture basic ontology languages and it is particularly suitable in our context, where the user may want to pose complex queries over a huge amount of data. Thus, in DL-Lite, answering conjunctive queries posed over the Personal Ontology can be done in polynomial time in the size of the personal data. Notably, DL-Lite comes with a system, called QUONTO [3], upon which OntoPIM is built.

Given an appropriate Semantic Wrapping layer that presents user's own data as DS objects, the Semantic Integration part of OntoPIM can be characterized by means of a quadruple $\mathcal{SI} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}, \mathcal{R} \rangle$, such that:

– $\mathcal{O}$ is the Personal Ontology, described by means of a DL-Lite TBox.
– $\mathcal{S}$ is a set of DS types.
– $\mathcal{M}$ is a set of mappings, i.e. a set of rules of the form:

$$R_S(\boldsymbol{v}) \rightarrow conj(\boldsymbol{x}, \boldsymbol{y}), I(\boldsymbol{x}, \boldsymbol{v}),$$

where $R_S \in \mathcal{S}$, $\boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}$ denote variables $v_1, ...v_n, x_1, ...x_n, y_1, ...y_m$, n is the arity of R_S, $m \geq 1$ and $conj(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of atoms of the form $C(z)$ or $R(z_1, z_2)$, where C and R are resp. a basic concept and a role in $\mathcal{O}$, z, z_1, z_2 are variables in $\boldsymbol{x}, \boldsymbol{y}$ and $I(\boldsymbol{x}, \boldsymbol{v})$ is a set of atoms of the form $I(x, v)$ that indicates that v is a representation of the instance x. We call I *Instance relation*.
– $\mathcal{R}$ is a set of rules, called *matching rules*, that specify how to identify and match different representations of the same instance of a given concept. These rules are applied to the set of atoms generated by the mappings.They may have one of the following forms:
  1. $C(x_1) \wedge C(x_2) \wedge I(x_1, v) \wedge I(x_2, v) \rightarrow x_1 = x_2$;
  2. $C(x_1) \wedge C(x_2) \wedge I(x_1, v_1) \wedge I(x_2, v_2) \wedge sim(v_1, v_2) \rightarrow x_1 = x_2$.

  where x, x_1, x_2 are variables in $\boldsymbol{x}$, v, v_1, v_2 are variables denoting data values, C is a basic concept of $\mathcal{O}$, $sim(v_1, v_2)$ is a predicate that checks whether v_1, v_2 are similar according to a certain similarity definition, and $conj(\boldsymbol{x})$ and $I(x_i, v_i)$, are defined as above for $i = 1, 2$.

To illustrate the scenario above, let us come back to the example of the Section 2. We establish a connection between the data of interest contained in each object of type **TCS** and the Personal Ontology graphically represented in Figure 1(a) by means of the following mapping assertion:

$$\begin{aligned}\textbf{TCS}(v_1, v_2, v_3, v_4, v_5, v_6, v_7) \rightarrow\ & \text{Goal}(x_1, x_4), I(x_4, v_4), \text{Destination}(x_1, x_3),\\ & I(x_3, v_3), \text{Assigned}(x_1, x_2), I(x_2, v_2), \text{From}(x_1, x_5),\\ & I(x_5, v_5), \text{To}(y, x_6), I(x_6, v_6).\end{aligned}$$

Then for each concept of $\mathcal{O}$ we define a matching rule of type 1. We also define the following matching rule of type 2 stating that two dates that are expressed in a different format represent the same instance of the concept Date:

$$\text{Date}(x_1), I(x_1, v_1), \text{Date}(x_2), I(x_2, v_2), \text{sameDate}(v_1, v_2) \rightarrow x_1 = x_2,$$

where we assume that the system is able to evaluate the predicate $sameDate(x_1, x_2)$.

Now, suppose that we are saving a travel cost statement concerning the travel that Mr. Cabernet made to participate to the World Wine Event (WWE) in Bordeaux from the 1/09/2003 to the 5/09/2003. The **TCS** mapping generates the following set of facts, that constitutes a portion of the DL-Lite ABox:

$$\begin{aligned}&\text{Travel}(x_1), \text{Event}(x_2), \text{Goal}(x_1, x_2), \text{City}(x_3), \text{Destination}(x_1, x_3),\\ &\text{Colleague}(x_4), \text{Assigned}(x_1, x_4), \text{Date}(x_5), \text{From}(x_1, x_5), \text{Date}(x_6), \text{To}(x_1, x_6).\end{aligned}$$

Moreover, the mapping generates the following portion of the Instance relation I:

Constant	Representation	Constant	Representation	Constant	Representation
x_2	WWE	x_4	Mr.Cabernet	x_6	05/09/03
x_3	Bordeaux	x_5	01/09/03		

Then, given the DL-Lite TBox expressed by means of the Personal Ontology $\mathcal{O}$, the DL-Lite ABox obtained above, the Instance relation I and the matching rules $\mathcal{R}$, the system can answer any conjunctive query over $\mathcal{O}$ and, for every constant x_i possibly returned, it proposes the set of corresponding representations, according to the computed extension of the relation I. Note that x_1 has not any representation. This is not surprising since the instances of the concept Travel would never be mapped to any attribute value. Similarly, it would not make sense to ask for an instance of the concept Travel.

5 Conclusion

We have presented a novel approach to Personal Information Management that takes advantage of the use of a Personal Ontology to store the data of one's desktop and to provide the user for an intelligent and efficient way of querying such data. We have proposed a framework that (i) overcomes data heterogeneity and lack of semantics by the use of a Semantic Wrapping function, (ii) integrates data and makes it accessible through a unified, user's conceptual view, by the use of a Semantic Integration function. Finally, we have presented the architecture of the system.

Currently, we are facing the *instance matching* problem by incorporating in the framework a set of rules responsible for detecting different representations of the same instance. In the future, we plan to investigate how to produce this set of rules. Moreover, note that once the matching rules have been applied, we actually keep all different representations of the same instance. However, sometimes we may want to correct some them. Suppose for example that in our domain view, a colleague has a unique email address. On one hand, two different email addresses may be associated to the same colleague because of spelling errors, in which case we would like to keep only the correct address. On the other hand, whenever a colleague moves, we may want to update his address while keeping the old one, in order to be able to retrieve, for example, an email that he sent us before moving. We plan to work on this in the future. Finally, we have discussed how personalization would come into play in order to help the user expressing queries, saving documents, etc. This will be the object of future deeper joint research activities. Moreover, we aim at studying how to rely on OntoPIM in order to develop a task-centered tool that would for example automatically fill a travel cost statement thanks to the data in our desktop.

References

1. Google Desktop, `http://desktop.google.com/`.
2. DELOS NoE, `http://http://delos-noe.iei.pi.cnr.it/`.
3. A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QuOnto: Querying Ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, 2005.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI'05)*, 2005.