

# A Framework for RDF User Profile Management

Ignazio Palmisano, Domenico Redavid, Giovanni Semeraro, Marco Degemmis,  
Pasquale Lops, and Oriana Licchelli

Dipartimento di Informatica, Università degli Studi di Bari  
Campus, Via Orabona 4, 70125 Bari, Italy  
email: {*palmisano, redavid, semeraro, degemmis, lops, licchelli*}@di.uniba.it

**Abstract.** The semantic evolution of the Web has an heavy impact on traditional techniques for user profiling, starting from the different representation of profiles in the new environment. From the Semantic Web perspective, user profiles should no longer use proprietary formats for profile encoding; instead, the data stored in them should be transferred to interoperable formats, possibly in languages with a well defined semantic. The main languages for these tasks are RDF (Resource Description Framework) and OWL (Web Ontology Language). User profiling applications can leverage the added power of these languages in order to develop interoperable and semantically rich systems. We present a framework to manage user profiles represented in RDF, following OWL ontologies, backed up by a component for generic RDF storage with multiuser support.

## 1 Motivation

The Semantic Web, the well known evolution of the Web foreseen by Sir Tim Berners-Lee [3], offers a new perspective to evaluate current state-of-the-art personalization systems. The Semantic Web initiative aims to build up a Web made of machine understandable informations; to do so, it is necessary to find a way of representing information that gives full access to machines, i.e. a representation language in which all concepts and all relations are explicit and well defined. With this aim, currently two main languages have been defined by W3C: RDF (Resource Description Framework)<sup>1</sup> and OWL (Web Ontology Language).<sup>2</sup>

RDF is primarily focused on the concepts of resource and property: a resource is an identifiable entity, e.g. a human being, a web site, or a building, while a property is a relation between two resources or between a resource and a literal value (e.g. a human being is related to his name). It is possible to refer to named and unnamed resources; a named resource is identified by its name, which is an URI; an unnamed resource has no persistent identifiers, and can be accessed or retrieved by means of the related properties. This kind of resource is called blank node.

---

<sup>1</sup> <http://www.w3.org/RDF/>

<sup>2</sup> <http://www.w3.org/2004/OWL/>

Every RDF construct is built up of triples. Each triple is of the type (Subject, Predicate, Object) where Subject and Predicate are resources (in particular, Predicate identifies a relation and has to be a named resource), while Object can be a resource (named or unnamed) or a literal value. A literal value can include a language tag, to identify the language in which it is expressed, or a type tag, that identifies with an URI the type of the literal (e.g. the string “1” can represent a string or an integer; the type tag disambiguates this situation). A set of triples is a RDF Model (or Description).

The RDF language is the base for the use of languages with a richer semantic; in particular, the most widespread languages for RDF are RDFSchema and OWL. OWL includes RDFSchema, in order to reuse the concepts already described there, and is divided into three sublanguages (Lite, DL, Full), with different constraints and different computational complexity of the related reasoners.

While in RDFS the main relation is inheritance, i.e. the definition of subclass/superclass relations between resources and subproperty/superproperty relations for properties, OWL introduces a more complex semantic, e.g. restrictions on properties (it is possible to define cardinalities and data ranges for properties); the main advantage of this language, however, is the well defined semantic of the defined relations; this enables the construction of automatic reasoners that are not limited to a particular domain or to a particular implementation. Since OWL ontologies are expressed in RDF, there is no need for a separate storage layer for OWL data; and, since RDF is an abstract specification that can have different representations (see RDF/XML, Notation3, N-Triples, Turtle), it is possible to exchange RDF data between application without imposing an a priori representation.

From this brief description, it is easy to understand that any proprietary format for user models can be translated into RDF without losing information. However, it is obvious that this is not a good reason to modify working software systems. The added value from the use of RDF as communication language, and OWL as language to describe ontologies representing the user model, lies in the ability to use the available reasoners in order to draw inferences from the data. Reasoners are able not only to use OWL semantics [8] to draw assertions, but can use any ontology that refers to the domain of the data. The use case that arise is as follows:

- Data and knowledge about a user or a group of users is collected from the available sources (e.g. web logs, database transactions, direct input from the user)
- Data and knowledge are expressed in a RDF model, according to a predefined ontology for the application domain
- The reasoner can now run on the RDF data, referring to the available ontology
- If a new ontology is developed, providing more ontological knowledge on the application domain, the new ontology can be used in conjunction with the

previous ontology and the original data, and the reasoner can be run again to draw more assertions

For the last step to be feasible, it is often necessary to first align the ontologies [7] [10], in order to detect correspondences between classes and relations that can be used to relate the ontologies in use. This step is necessary, because any difference in the ontology design can hamper the process, and thus not lead to improvements in the learning task. As an example, let us suppose to have a reference ontology for our data that has only one class to describe books (the class *Book* itself). In this ontology, it is not possible to discriminate between horror novels and conference proceedings. When an ontology with more detailed knowledge has to be used, it is necessary, for this ontology to be directly usable on the data, that the *Book* class is defined in both ontologies. However, this is not granted, since the new ontology could use the class *Publication* instead. Many approaches to solve this problem are already in literature [4]; however, in this work we will consider the ontologies to be compatible with each other, since this does not change our approach in a significant way.

## 2 Related Work

The idea of integration of multiple user profile data into a single model is not new; many similar approaches have already been studied, e.g. the use of standardized XML schemata to provide interoperable user models (see [9]); the idea is that a set of standardized tags is used to describe the user models. However, the limitation to XML Schema hampers extendibility, and does not make the semantic of underlying data explicit; as a consequence, it is not possible to use a general purpose reasoner on these data.

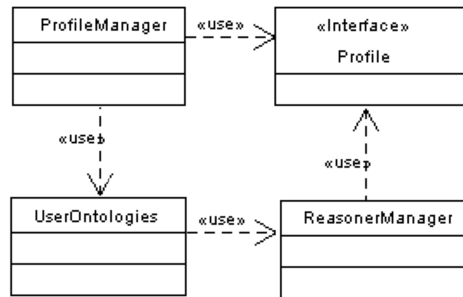
The most recent approach based on RDF as representation is UUCM (Unified User Context Model) [11], which is based on an extensible representation for models. UUCM provides a simple schema to describe different dimensions of user models; each dimension can be described through values that can be either simple types (such as strings, numbers, dates) or be typed. In this last case, the type of the value is expressed as classes defined in OWL language.

The system we are going to present is not tied to a particular ontology, but can be used with any OWL ontology like UUCM.

## 3 The ProfileManager System

### 3.1 System Architecture

ProfileManager is a framework devoted to manage user profiles represented as RDF models; the two main modules of the system are *ProfileManager* itself and *ReasonerManager*; these are interfaces that can be implemented independently. In this way, using the well known *Strategy* pattern [6], it is possible to use different implementation strategies (e.g. local storage or remote storage).



**Fig. 1.** Sketch of the system architecture

ProfileManager does not implement directly strategies for user profiling or machine learning algorithms; instead, it aims at building an abstraction layer to simplify the integration of different components that can contribute to the user profiling task. The actual implementation integrates two components for user profiling and a component for RDF management, that will be presented in the following. In Figure 1 a sketch of the architecture of the system is presented.

*ProfileManager* enables to abstract from all the operations that are needed to create a *Profile* object. *Profile* is an interface specifying the behavior of what we intend to be a user profile, i.e.:

- a RDF model containing user data (collected from different sources and translated to RDF)
- a set of ontologies referred in the user data
- a RDF model containing profile data (i.e. data about the user profile, created by the applications that will use the framework - see Section 3.2)
- a set of ontologies referred in the profile data
- an identifier of the referred user

The *Profile* can be used to modify the user data or the profile data, as well as the referred ontologies, and then can be saved back to the persistent storage.

The ReasonerManager interface enables an application to use an external reasoner to draw every assertion that is implicitly present in the data. The actual implementation works with a DIG<sup>3</sup> reasoner (currently RACER<sup>4</sup>). The use of RACER enables the ReasonerManager to cope with OWL DL semantics and ontologies; the limitation to this subset of OWL depends on the undecidability of OWL Full; in fact, there are some ongoing efforts to build OWL Full reasoners<sup>5</sup>, but the computational complexity of even OWL DL is too high for the use we foresee for our framework.

<sup>3</sup> <http://sourceforge.net/projects/dig>

<sup>4</sup> <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>

<sup>5</sup> <http://www.w3.org/2004/OWL/>, Tools, Projects and Applications Section

### 3.2 Integrated Components

The ProfileManager framework integrates many other components developed in the LACAM lab<sup>6</sup>; they are:

- RDFCore: a component for RDF storage
- Profile Extractor: a component for supervised learning of user classification rules
- ITR (ITem Recommender): a component for content based classification, based on naïve Bayes classifiers

While ProfileManager implementation is based on RDFCore (i.e. the implementation relies on the services of RDFCore), both the Profile Extractor component and the ITR component use the ProfileManager interface as repository for user and profile data. They therefore act as external applications using the framework to ease the managing of RDF profiles.

In Figure 2, a sketch of the framework and how it is related to the other components is presented.

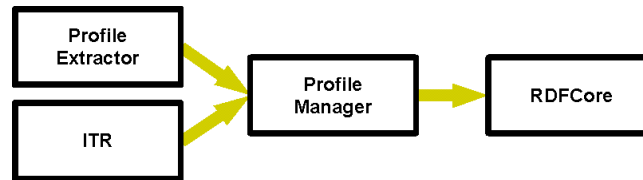


Fig. 2. Whole framework sketch

### 3.3 RDFCore

The RDFCore component, presented in [5], is a component used for RDF descriptions storage and retrieval, including multiuser support and extensible support for query languages.

The main modules of RDFCore are *DescriptionManager* and *TripleManager*. The first one gives access to Creation, Retrieval, Updating and Deletion (CRUD) operations on RDF models seen as a whole, while the second component enables the same operations at the single assertion level. Both modules use the Jena Semantic Web Toolkit[2] API to work with RDF models.

RDFCore has been adopted in the VIKEF Project as the basic component for RDF metadata storage in the VIKE (Virtual Information and Knowledge Environment) Framework, where its SOAP<sup>7</sup>-exposed services have been wrapped as a Web Service<sup>8</sup> for metadata storage, retrieval and querying.

<sup>6</sup> <http://lacam.di.uniba.it:8000>

<sup>7</sup> <http://www.w3.org/2000/xp/Group/>

<sup>8</sup> <http://www.w3.org/2002/ws/>

RDFCore also has extensible support for different solutions for physical persistence. At the time of writing, there are four implementations of *RDFEngineInterface* (the basic interface to be implemented by plugins), two based on the already mentioned Jena Toolkit, one with MySQL RDBMS<sup>9</sup> as persistent storage, called *RDFEngineJENA*, and the other one using Microsoft SQL Server<sup>10</sup>, using the resources by Erik Barke<sup>11</sup>, called *RDFEngineMsSQL*. The third implementation is based on simple RDF/XML files, and is called *RDFEnginePlain*. The fourth implementation, called *RDFEngineREDD*, is the one in which we implemented the REDD algorithm natively in the storage level. It uses Oracle<sup>12</sup> as RDBMS.

The component also offers multiuser support; users can choose whether some of the models they own should be private, publicly readable or writable, and can restrict access to single users or groups of users. This support is useful when designing cooperative applications, thus enabling geographically dispersed teams to work together easily.

In Figure 3 there is a small sketch of the system architecture.

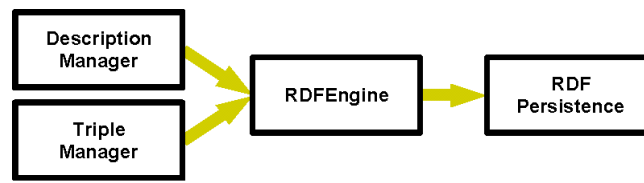


Fig. 3. Architecture of the RDFCore system

### 3.4 Profile Extractor

The Profile Extractor (PE) [1] is a module that classifies users using supervised learning techniques. It can be used to discover users' preferences by analyzing data relative to user interaction or other data that are gathered from different data sources, such as data warehouse or transactions, in order to infer rules describing the user behavior. More in detail, these data are represented in RDF and refer to a simple ontology designed to be used as UUCM value type. The ontology, part of which is depicted in Figure 4, is actually limited in its scope, since, as we will see, the PE component is limited to the use of zero order data (vectors of attribute/value pairs), and cannot exploit relational knowledge available in the input data.

<sup>9</sup> <http://dev.mysql.com/doc/mysql/en/index.html>

<sup>10</sup> [www.microsoft.com/sql/](http://www.microsoft.com/sql/)

<sup>11</sup> <http://www.ur.se/jena/Jena2MsSql.zip>

<sup>12</sup> more specifically Oracle 9.2.0.1.0 also known as Oracle 9i Release 2 <http://otn.oracle.com/documentation/oracle9i.html>

To build profiles, the PE component uses decision rules induced from training data. At the actual stage of development, the information that PE stores into profiles consists of information on the set of categories that the user belongs to; these categories are defined in the learning problem for the component.

The rules are inferred through the use of well-known Machine Learning techniques, such as partition trees. In order for the rules to be inferred in an efficient way, and to maximize the predictive power of the inferred rules, it is necessary to establish what features and attributes, in the available data, are useful to accomplish the learning task, and what data, on the other hand, would not increase the predictive power or could waste computation time. The other main problem concerns the definition of meaningful classes to learn, which are to be defined before the learning task starts.

The problem of learning user preferences can be cast to the problem of inducing general concepts from examples labeled as members (or non-members) of the concepts. In this context, given for example a finite set of categories of interest  $C = \{c_1, c_2, \dots, c_n\}$ , the task may consist in “learning the target concept  $T_i$  users interested in the category  $c_i$ ”. In the training phase, the users are positive examples for the categories they like/are interested, and negative examples for the categories they don’t like/have interest. We chose an operational description of the target concept  $T_i$ , using a collection of rules that match against the features describing a user in order to decide if he/she is a member of  $T_i$ . Hence, the problem is reduced to the combination of a number of binary classifiers, in this specific context. For particular classes, where the expected value is not binary (like/dislike), but has more possible values (likes much/enough/little/nothing), the solution is still valid, but the classifier will not be binary; this could result in a small increase in the required computational time. Transactional data or other data related to users have to be arranged into a set of unclassified instances; the training set must be labeled by a domain expert (acting as an oracle) in order to label a specific instance as a positive or negative example. Then, the training instances are processed by the Profile Extractor, which induces a classification rule set for each target concept. Rules are used to classify each user, as part of the target concept on the basis of the transactional data.

The Profile Extractor system is divided into four main modules:

- Learning task service: responsible for the inference of rule sets from labeled data, for classification of unlabeled data, and for validation of the obtained results (the test phase). It is organized as a library, and provide SOAP services for remote invocation. Inputs and outputs of the system are RDF data. The system is designed to be loosely coupled with the specific ontology used to describe the data, so that the ontology can be easily updated.
- Rules Manager: responsible for rule sets managing; it is a Web Application for remote managing of the sets of rules, enabling version check and creation/deletion of rules. It uses the services of the Learning task service.
- Labeling Manager: responsible for the creation of labeled instances of the data; it is a Web Application that provides the Learning task service with labeled examples for its training phase.

- Profile Manager: responsible for the profiling task, i.e. the use of the classification rules to build profiles for users not labeled by the domain expert.

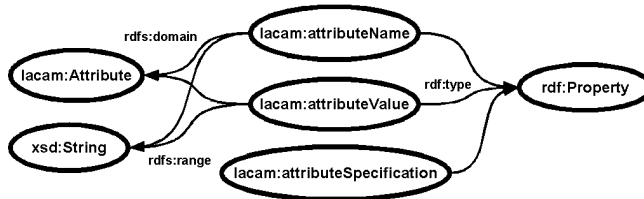


Fig. 4. Sketch of the PE input ontology

### 3.5 IItem Recommender

ITR (IItem Recommender) implements a probabilistic learning algorithm, the naïve Bayes classifier, relying on a content-based approach. The prototype is able to classify documents as interesting or uninteresting for a particular user, on the ground of the textual content of the documents. This approach is analog to the relevance feedback in Information Retrieval [12], which adapts the query vector by iteratively absorbing users relevance judgments on newly returned documents. In the Information Filtering paradigm, the tuned query vector is actually a profile model, specifying both keywords and their informative power. Based on the constructed user profile, a new item relevance is measured by computing a similarity measure between the query vector and the item’s feature vector. Learning a user profile generally involves the application of Machine Learning techniques to generate a predictive model based on information that has been previously labeled by the user. To learn user profiles, ITR casts the problem as a Text Categorization (TC) problem. The techniques used are those that are well-suited for text categorization [13].

The task is to approximate the unknown target function  $\Phi : D \times C \rightarrow \{True, False\}$ , where  $D$  is a set of documents and  $C$  a set of categories. We consider the problem of learning user profiles as a binary TC task: each document has to be classified as interesting or not w.r.t. the user preferences. Therefore, the set of categories is restricted to  $c_+$ , that represents the positive class (user-likes), and  $c_-$  the negative one (user-dislikes).

ITR representation is based on *bag of words* (BOW). In this approach each feature corresponds to a single word found in the training set.

Naïve Bayes is a probabilistic approach to inductive learning, and belongs to the general class of Bayesian classifiers. The learned probabilistic model estimates the a posteriori probability,  $Pr(c|d)$ , of document  $d$  belonging to class  $c$ . This estimation is based on the a priori probability,  $Pr(c)$ , i.e. the probability of observing a document in class  $c$ ,  $Pr(d|c)$ , that is the probability of observing

the document  $d$  given  $c$  and,  $Pr(d)$ , the probability of observing the instance  $d$  at all. Using these probabilities, Bayesian classifiers apply Bayes theorem to calculate  $Pr(c|d)$ .

As working model of the naïve Bayes classifier, ITR uses the multinomial event model. The way the multinomial event model uses its document vectors to calculate  $Pr(c_j|d_i)$  is as follows:

$$P(c_j|d_i) = P(c_j) \prod_{w \in V_{d_i}} P(t_k|c)^{N(d_i, t_k)}$$

where  $N(d_i, t_k)$  is defined as the number of times word or token  $t_k$  appeared in document  $d_i$ . A key step in implementing naïve Bayes is estimating the word probabilities  $P(t_k|c_j)$ . ITR use Witten-Bell smoothing [14] that sets  $P(t_k|c_j)$  as follows:

$$P(t_k|c_j) = \begin{cases} \frac{N(t_k, c_j)}{V_{c_j} + \sum_i N(t_i, c_j)} & \text{if } N(t_k, c_j) \neq 0 \\ \frac{1}{V - V_{c_j}} & \text{if } N(t_k, c_j) = 0 \end{cases}$$

where  $N(t_k, c_j)$  is the count of the number of times word  $t_k$  occurs in the training data for class  $c_j$ ,  $V_{c_j}$  is the total number of unique words in class  $c_j$ , and  $V$  is the total number of unique words across all classes.

The final outcome of the learning process is a probabilistic model used to classify a new instance in the class  $c_+$  or  $c_-$ . The model can be used to build a personal profile that includes those words that turn out to be most indicative of the user's preferences.

## 4 Conclusions and Future Work

We presented a framework based on Semantic Web technologies that can be used to abstract from data managing issues for user profiling applications. Not enough work has been done on exploiting the features coming from the use of Semantic Web technologies to enhance the quality of profiling already carried out by the integrated systems. The future developments of the framework will mainly aim to update the techniques used in the integrated components in order to use the richer knowledge that can be encoded in the input data. As an example, at the time of writing we are examining the possibility of using hierarchical information between resources to find better discrimination criteria for partition tree classifiers, as used in the Profile Extractor component.

## 5 Acknowledgments

This research was partially funded by the European Commission under the 6<sup>th</sup> Framework Programme IST Integrated Project VIKEF - Virtual Information and Knowledge Environment Framework (Contract no. 507173, Priority 2.3.1.7 Semantic-based Knowledge Systems; more information at <http://www.vikef.net>).

## References

1. F. Abbattista, M. Degemmis, O. Licchelli, P. Lops, G. Semeraro, and F. Zambetta. Agents, Personalisation and Intelligent Applications. In R. Corchuelo, A. Ruiz Cortés, and R. Wrembel, editors, *Technologies Supporting Business Solutions, Part IV: Data Analysis and Knowledge Discovery, Chapter 7*, pages 141–158. Nova Sciences Books and Journals, 2003.
2. B. McBride. JENA: A Semantic Web toolkit. *IEEE Internet Computing*, 6:55–59, Nov-Dec 2002.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
4. M. Ehrig and Y. Sure. Ontology mapping - an integrated approach. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, volume 3053 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2004.
5. F. Esposito, L. Iannone, I. Palmisano, and G. Semeraro. RDF Core: a Component for Effective Management of RDF Models. In Isabel F. Cruz, Vipul Kashyap, Stefan Decker, and Rainer Eckstein, editors, *Proceedings of SWDB'03, The first International Workshop on Semantic Web and Databases, Co-located with VLDB 2003, Humboldt-Universität, Berlin, Germany, September 7-8, 2003*, 2003.
6. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley, 1st edition, 1995.
7. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, volume 3053 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2004.
8. P. Hayes. RDF semantics, 2004. W3C Recommendation 10 February 2004 <http://www.w3.org/TR/rdf-mt/>.
9. D. Heckmann and A. Krüger. A user modeling markup language (userml) for ubiquitous computing. In *User Modeling*, pages 393–397, 2003.
10. Y. Kalfoglou and W. Marco Schorlemmer. Formal support for representing and automating semantic interoperability. In Christoph Bussler, John Davies, Dieter Fensel, and Rudi Studer, editors, *The Semantic Web: Research and Applications, First European Semantic Web Symposium, ESWS 2004, Heraklion, Crete, Greece, May 10-12, 2004, Proceedings*, volume 3053 of *Lecture Notes in Computer Science*, pages 45–60. Springer, 2004.
11. C. Niederée, A. Stewart, B. Mehta, and M. Hemmje. A Multi-Dimensional, Unified User Model for Cross-System Personalization. In Liliana Ardissono and Giovanni Semeraro, editors, *Proceedings of the AVI 2004 Workshop On Environments For Personalized Information Access*, pages 34–54, 2004.
12. G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
13. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 2002.
14. I.H. Witten and T.C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4), 1991.