A RDF-based Framework for User Profile Creation and Management

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

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

Abstract. The semantic evolution of the Web has an heavy impact on traditional systems, as the ability to use a formal interoperable language simplifies information exchange between different systems. In order to foster information exchange and to easily connect new functionalities to semantic knowledge bases, in order to be able to use and reuse the valuable knowledge embedded in the existing systems, we designed a plugin-based framework, and used it to connect together different tools and systems developed in the LACAM laboratory. Our pilot project includes user profiling abilities coming from two components, namely Profile Extractor (PE) and Item Recommender (ITR), and storage capabilities implemented by a repository tool called RDFCore.

1 Introduction

One of the main points for the Semantic Web to be useful is interoperability; Semantic Web applications should be able to exchange information with (almost) no human intervention needed. By exchanging information, we mean exchanging meaningful information, i.e. two applications A and B should be able to share not only the bare data (which is already doable in a number of ways, one of which is through standards like XML), but the associate meaning, in a reliable way. For this to be possible, the process must be described in an unambiguous way; the easiest solution is then to express the knowledge that the applications want to share in a formal language, with well defined and logically based semantics. With this aim, currently two main languages have been defined by W3C: RDF (Resource Description Framework)[1] and OWL (Web Ontology Language).[2]
The use of the Semantic Web languages (OWL Full/DL/Lite) enables applications to decouple knowledge from application machinery, thus enableng other applications to share the meaning, provided that they can understand the same logic language.

[1] `http://www.w3.org/RDF/`
[2] `http://www.w3.org/2004/OWL/`

With this aim, we designed a framework to ease the realization of semantic applications. The framework is based on a set of interfaces that abstract some common functionalities, built around the concept of *flow* of information.

2 The flow metaphor

By *flow* of information, we mean the transmission of information (expressed as ontological information in SW languages, with DL semantics as background) from a *source* to a consumer for that information (*sink*). Along the path, the information flow can be modified in many ways; we identify two main approaches for information change: *enrichment* and *transformation*.
Enriching an information flow means adding new information to this flow; an example could be the use of inference and deduction rules in order to explicit some implicit knowledge or to add new information coming from background knowledge.
Transforming an information flow involves the rewriting of the information; an example could be the change of background ontology for some data, or the merge of two information flows into a single one, involving the use of ontology alignment techniques.
Another kind of flow modification is the *store* of an information flow for later use; this is the typical job of a persistent storage component.
A sketch of the resulting architecture is in Figure 1.
The kind of components that implement these interfaces can then be summarized as:

- Source plugins: this kind of plugin creates new information (e.g., wrapping an external source of information, such as a database)
- Store plugins: this kind of plugin stores information, enabling both persistent storage and retrieval
- Transformer plugins: this kind of plugin modifies the information it is fed with (e.g., changing class or property definitions)
- Enricher plugins: this kind of plugin differs from the Transformer because it does not modify existing informations, but adds new information (e.g., an external reasoner could be wrapped in this kind of plugin)
- Sink plugin: this kind of plugin does not produce or modify information in the framework (e.g., a visualization plugin or an external application that needs to get information from the framework)

3 Information Flow Language

As already said, the Semantic Web languages for ontology expression are RDF and OWL, with the RDFSchema[3] language as an intermediate level of expressiveness (and of computational complexity).

[3] `http://www.w3.org/TR/rdf-schema/`

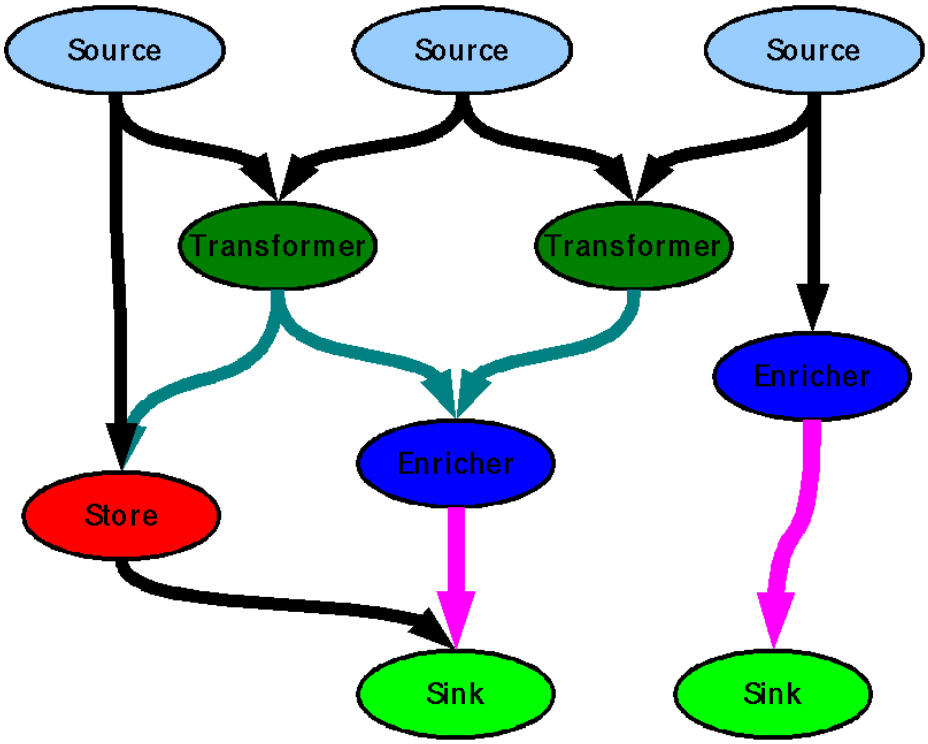

Fig. 1. Architecture Sketch

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). A set of triples `(Subject, Predicate, Object)` is a RDF Model (or Description).
The RDF language is the base for the use of languages with a richer semantic, such as RDFSchema and OWL. OWL includes RDFSchema, in order to reuse the concepts already described there, and is divided into three sublanguages (Lite, DL, Full).
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.
As a consequence, the language for the information flow in our framework is RDF.

4 Framework Test Case

In order to verify the framework in a practical scenario, we used the defined interfaces to wrap up other components developed in the LACAM lab. The components we included so far are:

- *RDFCore*: a component for RDF storage, wrapped as a *store* plugin
- *Profile Extractor*: a component for supervised learning of user classification rules, wrapped up as an *enrichment* plugin
- *ITR (ITem Recommender)*: a component for content based classification, based on naïve Bayes classifiers; from this component, which originally was a Java Web Application, many plugins have been created:
  - an *enrichment* plugin, that encloses the learning abilities of the system, in a way similar to Profile Extractor
  - a *source* plugin, that encapsulates the part of the web application that gains data from users and domain experts
  - a *sink* plugin, that contains the result display part of the system

The resulting istance of the framework is biased towards the user modeling domain, as is easy to see from the description of the systems that foollows; other work in this area has been done, for example UUCM

(Unified User Context Model) [5], 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 components we integrated are not tied to a particular ontology, but can be used with any OWL ontology like UUCM.

4.1 RDFCore

The RDFCore component, presented in [3], 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.
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.
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[4]-exposed services have been wrapped as a Web Service[5] for metadata storage, retrieval and querying.
In Figure 2 there is a small sketch of the system architecture.

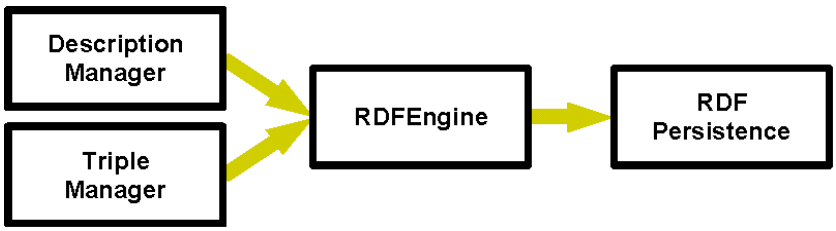

Fig. 2. Architecture of the RDFCore system

[4] http://www.w3.org/2000/xp/Group/
[5] http://www.w3.org/2002/ws/

4.2 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 is actually limited in its scope, since 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.
To build profiles, the PE component uses decision rules induced from training data, 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 = $\{c_1, c_2, \ldots c_n\}$, 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.

4.3 ITem Recommender

ITR (ITem 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 [6], 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 [7].
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 concepts* (BOC) [8]. In this approach each feature corresponds to a single word found in the training set.
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.

4.4 Other Plugins

Other algorithms developed in the LACAM lab have been ported in the framework or are migrating at the time of writing; in particular, the REDD algorithm [4] has been wrapped in a Transformer plugin, enabling the applications that use the framework to apply the redundancy detection algorithm on any RDF model they use.
REDD is based on blank node semantics, and is able to detect redundancies in RDF (and OWL) models, where, for example, multiple blank nodes with no distinguishing features are present in the same model.
This is the case, for example, of a remote store that gets updates from other applications; it is possible that one or more applications send the same information more than once, and, while this is not a problem with RDF ground statements (i.e. statements with no blank nodes), since RDF models are defined as triple sets, blank nodes inserted during the life of the model are not recognized as already present; this increases the size of the models without reason, and could also be regarded as an error.
Another possible application, which is under experimentation at the time of writing, is the use of REDD to detect redundant definitions in ontologies; the ongoing project aims at using the framework in the building of a Protégé [6] plugin.

5 Semantic Evolution

On the side of algorithm evolution, in the ITR component the update to OWL formalism is strictly related to the switching from keyword-based representation of the user profile to user profiles based on concepts

[6] `http://protege.stanford.edu`

(bags-of-concepts, BOC, instead of bag-of-words, BOW). While this shift of representation is natural when considering the new environment, we already demonstrated in [8] that the traditional TF-IDF heuristic gains some percents both in precision and recall from the new representation. Moreover, we are currently doing empirical measures on an evolution of TF-IDF that takes into account the hierarchical relations between concepts, that, informally, redefines the classical definition of TF-IDF, which is based on sheer concept occurrence number, taking into account that a more specific term is also an instance of a more general term, and as consequence, so to speak, each occurrence of the more specific term counts also as an occurrence of the more general term.

6 Acknowledgments

This research was partially funded by the European Commission under the 6^{th} 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. 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.
4. Floriana Esposito, Luigi Iannone, Ignazio Palmisano, Domenico Redavid, and Giovanni Semeraro. REDD: An Algorithm for Redundancy Detection in RDF Models. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *The Semantic Web: Research and Applications, Second European Semantic Web Conference*, volume 3532 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 2005.
5. 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.

6. G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
7. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), 2002.
8. Giovanni Semeraro, Marco Degemmis, Pasquale Lops, and Ignazio Palmisano. WordNet-based User Profiles for Semantic Personalization. In P. Brusilovsky, C. Callaway, and A. Nurnberger, editors, *Proceedings of the Workshop on New Technologies for Personalized Information Access (PIA 2005), part of the 10th Int. Conf. on User Modeling (UM'05), Edinburgh, UK, 2005.*, pages 74–83, 2005.